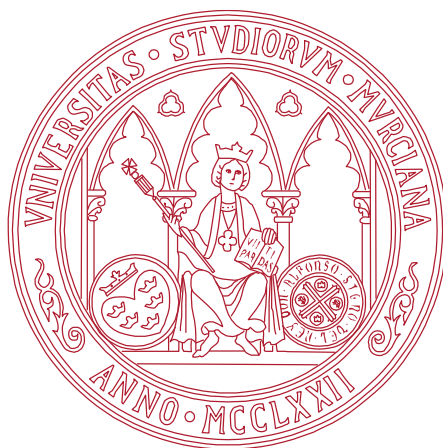




UNIVERSIDAD DE MURCIA

**DEPARTAMENTO DE INFORMÁTICA Y
SISTEMAS**

TESIS DOCTORAL

**Un proceso de Ingeniería de Requisitos
dirigido por modelos
centrado en reutilización**

**D^a Begoña Moros Valle
2012**

UNIVERSIDAD DE MURCIA
Facultad de Informática
Departamento de Informática y Sistemas



TESIS DOCTORAL

**Un proceso de Ingeniería de Requisitos dirigido por
modelos centrado en reutilización**

Doctoranda
Begoña Moros Valle

Director
Dr. D. José Ambrosio Toval Álvarez

2012

*Para Jacobo y Lola
por todo el tiempo que les he robado.*

*Para mis padres
por su generosidad y apoyo incondicional.*

*A mi abuela
ejemplo de vida.*

AGRADECIMIENTOS

Ha sido tan largo el camino que me ha llevado hasta aquí, que han sido muchas las personas que me han acompañado en todo o parte de él y que han contribuido, directa o indirectamente a que este trabajo sea lo que es. Espero que no se me olvide nadie, si así ocurriera, mis disculpas anticipadas por la omisión, creedme, no deliberada.

En primer lugar, es justo reconocer la labor de mi director de tesis, Ambrosio Toval, como I.P. en los proyectos de investigación de los que se ha ido nutriendo esta tesis. Sus revisiones y consejos han contribuido, sin duda, a mejorar la calidad de este trabajo. Agradezco su confianza, durante estos años, en mi capacidad para sacar adelante los resultados de nuestras investigaciones.

También quería expresar mi más sincero agradecimiento a Cristina Vicente-Chicote, ella fue realmente la que me impulsó y me sostuvo en uno de esos baches del camino en el que pierdes el rumbo y la perspectiva. Pasamos horas discutiendo sobre el “metamundo” y sobre lo “divino y lo humano”. Espero que en algún momento veas los frutos finales de nuestro trabajo y te sientas orgullosa por lo que de tí hay en ellos.

En la ruta de ascenso hacia el “metamundo” también me ha acompañado Diego Alonso, en el que he encontrado además de un guía aventajado, un amigo y espero que colaborador a partir de ahora. También de la UPCT, agradezco a Paqui y a Pedro su disposición y ayuda durante la elaboración del caso de estudio.

Una vez alcanzada la cima del “metamundo”, en el camino hacia las transformaciones, me resultó imprescindible la ayuda de Jesús Sánchez, el padre de RubyTL. Agradezco enormemente su disposición incondicional, incluso durante el mes de agosto, y su rapidez en responder a todas mis preguntas de principiante.

Durante estos años he ido compartiendo vivencias y experiencias con los miembros del grupo de investigación: Joaquín Nicolás, José Sáez, Joaquín Lasheras, Fernando Molina, Miguel Ángel Martínez y Fran J. Lucas. Algunos de ellos se adelantaron en el camino, llegaron antes a la meta y ya son doctores. Otros, con un trabajo brillante, se quedaron atascados. Espero que algún día recobren las fuerzas para continuar este camino sinuoso pero reconfortante al final. Me gustaría dedicar una mención especial a Joaquín Nicolás, con el que he trabajado codo con codo en la definición del método SIREN, germen de esta tesis doctoral.

Además del apoyo científico, igual o casi más importante es sentirte arropado o apoyado a nivel humano. En este plano, agradezco a mis compañeros de “pasillo4”: Joaquín, Mariajo, Ginés, Marcos y Quino, los ratos distendidos en nuestros cafés y aperitivos, compartiendo preocupaciones, en estos tiempos convulsos, y también

alegrías, como los nacimientos de nuestras criaturas, que acaparan el centro de nuestras conversaciones.

A mis amigos, que durante tanto tiempo les he hablado de la tesis y que parecía más una excusa para ausentarme de los planes de grupo que un trabajo real. Esta es la prueba de que siempre ha estado ahí, aunque no se haya materializado hasta este momento. Gracias a todos por “aguantarme”.

Por último, también es justo reconocer que todo esto no habría sido posible sin el apoyo de mi familia. De Jacobo, por ocuparse de todo mientras yo estaba inmersa en mis modelos y transformaciones; de Lola, por alegrarme todos los “tiempos muertos” con su inteligencia, vivacidad y frescura; de mis padres y hermanos, por cubrirme en mis ausencias y animarme a seguir hacia delante.

Gracias, gracias, gracias a todos!!

Begoña Moros Valle

*Cuando quieres realmente una cosa, todo el Universo conspira
para ayudarte a conseguirla.*

(Paulo Coelho)

Se quiere más aquello que se ha conseguido con muchas fatigas.

(Aristóteles)

ÍNDICE DE CONTENIDOS

ÍNDICE DE CONTENIDOS	XI
ÍNDICE DE FIGURAS.....	XV
ÍNDICE DE TABLAS	XIX
RESUMEN.....	XXI
ABSTRACT.....	XXIII
CAPÍTULO 1 INTRODUCCIÓN Y ANÁLISIS DE OBJETIVOS	1
1.1. INTRODUCCIÓN	1
1.1.1. <i>La importancia de la Ingeniería de Requisitos en el proceso de desarrollo del software</i>	2
1.1.2. <i>Ingeniería de Requisitos y reutilización</i>	9
1.1.3. <i>Ingeniería de Requisitos dirigida por modelos</i>	13
1.2. HIPÓTESIS Y OBJETIVOS.....	24
1.3. MARCO DE LA TESIS	25
1.3.1. <i>Grupo de investigación</i>	25
1.3.2. <i>Proyectos de investigación</i>	25
1.3.3. <i>Relación con empresas</i>	30
1.3.4. <i>Redes temáticas</i>	31
1.4. ORGANIZACIÓN DE LA TESIS	32
CAPÍTULO 2 ESTRATEGIA DE INVESTIGACIÓN	35
2.1. INTRODUCCIÓN	35
2.2. ESTRATEGIA DE INVESTIGACIÓN EN EL DESARROLLO DE LA TESIS.....	36
2.3. MÉTODO DE INVESTIGACIÓN CONSTRUCTIVO.....	37
2.4. REVISIÓN SISTEMÁTICA DE LA LITERATURA	39
2.5. MÉTODOS EMPÍRICOS PARA LA EVALUACIÓN DE RESULTADOS.....	42
2.5.1. <i>Caso de estudio</i>	43
2.5.2. <i>Encuesta</i>	45
CAPÍTULO 3 ANÁLISIS DEL ESTADO DEL ARTE	49
3.1. INTRODUCCIÓN	49

3.1.1.	<i>Uso de técnicas de Ingeniería de Requisitos en el DSDM</i>	50
3.1.2.	<i>Integración de los requisitos y modelos en el DSDM</i>	51
3.1.3.	<i>Enfoque de transformación entre requisitos de usuario y modelos de análisis</i>	52
3.2.	RSL METAMODELOS DE REQUISITOS.....	53
3.2.1.	<i>Marco de comparación de los metamodelos de requisitos</i>	54
3.2.2.	<i>Diseño de la revisión sistemática en relación a los metamodelos de requisitos</i>	57
3.2.3.	<i>Resultados</i>	61
3.2.4.	<i>Discusión</i>	70
3.2.5.	<i>Conclusiones</i>	80
3.3.	RSL TRANSFORMACIONES DE MODELOS DE REQUISITOS A OTROS MODELOS DEL DESARROLLO.....	81
3.3.1.	<i>Diseño de la revisión sistemática en relación a las transformaciones a partir de modelos de requisitos</i>	81
3.3.2.	<i>Resultados</i>	86
3.3.3.	<i>Discusión</i>	96
3.3.4.	<i>Conclusiones</i>	105
CAPÍTULO 4 SIREN: UN MÉTODO PRÁCTICO DE INGENIERÍA DE REQUISITOS BASADO EN REUTILIZACIÓN		107
4.1.	INTRODUCCIÓN.....	107
4.2.	MODELO DE PROCESO DE SIREN.....	109
4.2.1.	<i>Un modelo de proceso básico de IR</i>	109
4.2.2.	<i>El modelo de proceso de SIREN</i>	110
4.2.3.	<i>Modelo de roles de SIREN</i>	112
4.2.4.	<i>Tareas de un modelo de proceso general de IR con SIREN</i>	112
4.3.	GUÍAS DE SIREN.....	118
4.3.1.	<i>Guías para la definición de requisitos</i>	118
4.3.2.	<i>Guías para la definición de atributos</i>	120
4.3.3.	<i>Guías para la definición de requisitos parametrizados</i>	123
4.3.4.	<i>Guías para la definición de trazas entre requisitos</i>	125
4.3.5.	<i>Guías para la organización de los catálogos</i>	128
4.3.6.	<i>Guías para la reutilización de requisitos</i>	130
4.3.7.	<i>Guías para la mejora del repositorio</i>	131
4.3.8.	<i>Guías para la organización de la documentación de especificación de requisitos</i>	131
4.4.	EXPERIENCIAS EN LA APLICACIÓN DE SIREN.....	132
4.4.1.	<i>SIREN y la seguridad</i>	132
4.4.2.	<i>SIREN y los sistemas teleoperados para el mantenimiento de cascos de buques</i>	140
4.4.3.	<i>SIREN en casos de estudios reales: Proyecto GARTIC</i>	142
4.5.	OCHO CUESTIONES CLAVE EN REUTILIZACIÓN DE REQUISITOS.....	144
4.5.1.	<i>Organización de los requisitos reutilizables (K1)</i>	145
4.5.2.	<i>Motor de búsqueda de requisitos reutilizables (K2)</i>	146
4.5.3.	<i>Selección y reutilización de requisitos con distintos niveles de granularidad (K3)</i>	146
4.5.4.	<i>Reutilización de atributos de los requisitos (K4)</i>	146
4.5.5.	<i>Reutilización de relaciones de traza (K5)</i>	147
4.5.6.	<i>Gestión de requisitos parametrizados (K6)</i>	147
4.5.7.	<i>Mejora del repositorio (K7)</i>	148
4.5.8.	<i>Soporte CARE para la reutilización (K8)</i>	148

4.6.	HERRAMIENTA DE SOPORTE: SIRENTOOL.....	148
4.6.1.	<i>La reutilización en las herramientas CARE comerciales.....</i>	149
4.6.2.	<i>SirenTool.....</i>	150
4.7.	SIREN _{MD} : UN PROCESO DE INGENIERIA DE REQUISITOS DIRIGIDO POR MODELOS.....	151
4.8.	CONCLUSIONES Y APORTACIONES AL ESTADO DEL ARTE.....	154
CAPÍTULO 5 REMM: UN METAMODELO PARA INGENIERÍA DE REQUISITOS.....		157
5.1.	INTRODUCCIÓN.....	157
5.2.	UN LENGUAJE PARA EL MODELADO DE REQUISITOS.....	159
5.3.	CONCEPTOS BÁSICOS Y RELACIONES DE TRAZA EN REMM.....	161
5.3.1.	<i>Conceptos básicos en REMM.....</i>	161
5.3.2.	<i>Trazabilidad extra-requisitos en REMM.....</i>	168
5.3.3.	<i>Trazabilidad inter-requisitos en REMM.....</i>	169
5.4.	METAMODELADO DE LA VARIABILIDAD PARA DAR SOPORTE A LA REUTILIZACIÓN DE REQUISITOS.....	172
5.5.	MODELOS BIEN FORMADOS. RESTRICCIONES.....	175
5.5.1.	<i>Gramática del lenguaje para la definición de condiciones.....</i>	178
5.6.	FORMALIZACIÓN DE SIREN EN REMM.....	179
5.7.	CONCLUSIONES Y APORTACIONES AL ESTADO DEL ARTE.....	179
CAPÍTULO 6 RETOCO: UN PROCESO COMPLETO DE DSDM, DESDE LOS REQUISITOS HASTA EL CÓDIGO.....		183
6.1.	INTRODUCCIÓN.....	183
6.2.	ReToCo: UN PROCESO DSDM DESDE LOS REQUISITOS HASTA EL CÓDIGO.....	186
6.2.1.	<i>Proceso de Ingeniería del Dominio.....</i>	188
6.2.2.	<i>Proceso de Ingeniería de la Aplicación.....</i>	192
6.2.3.	<i>Criterios de calidad.....</i>	198
6.2.4.	<i>Discusión: transformación vertical u horizontal.....</i>	199
6.3.	TRANSFORMACIONES DE MODELOS EN EL PROCESO RETOCO.....	200
6.3.1.	<i>Transformaciones modelo a modelo a partir de modelos de requisitos.....</i>	200
6.3.2.	<i>Transformaciones modelo a texto a partir de los modelos de requisitos.....</i>	213
6.4.	CONCLUSIONES Y APORTACIONES AL ESTADO DEL ARTE.....	219
CAPÍTULO 7 REMM-STUDIO*: UN ENTORNO INTEGRADO DE INGENIERÍA DE REQUISITOS DIRIGIDO POR MODELOS Y BASADO EN REUTILIZACIÓN.....		221
7.1.	INTRODUCCIÓN.....	221
7.1.1.	<i>Herramientas de gestión de requisitos.....</i>	222
7.1.2.	<i>Herramientas de soporte al DSDM.....</i>	223
7.1.3.	<i>Objetivo y organización del capítulo.....</i>	227
7.2.	ARQUITECTURA DE LA HERRAMIENTA.....	228
7.3.	EDITORES DE MODELOS.....	232
7.3.1.	<i>REMMextra: editor de los artefactos de IR relacionados con los requisitos.....</i>	232
7.3.2.	<i>REMMf: editor de modelos de requisitos para reutilización.....</i>	234
7.3.3.	<i>REMMb: editor de modelos de requisitos con reutilización.....</i>	236
7.3.4.	<i>Trazabilidad entre requisitos en REMMb y REMMf.....</i>	241
7.4.	HERRAMIENTA PARA VALIDAR LOS MODELOS.....	243
7.5.	EJECUCIÓN DE LAS TRANSFORMACIONES DE MODELOS.....	244
7.6.	CONCLUSIONES Y APORTACIONES AL ESTADO DEL ARTE.....	246
CAPÍTULO 8 CASO DE ESTUDIO: SISTEMAS DOMÓTICOS.....		249
8.1.	INTRODUCCIÓN.....	249
8.2.	HABITATION: UN DSL PARA LOS SISTEMAS DOMÓTICOS.....	251
8.2.1.	<i>Vista Catálogo del DSL Habitation.....</i>	252

8.2.2	<i>Vista Aplicación del DSL Habitation.....</i>	254
8.3.	PROCESO DE DESARROLLO DIRIGIDO POR MODELOS PARA LOS SISTEMAS DOMÓTICOS ..	255
8.3.1	<i>Aportaciones de esta tesis doctoral al proceso definido.....</i>	256
8.4.	HAREL: UN CATÁLOGO DE REQUISITOS REUTILIZABLES PARA LOS SISTEMAS DOMÓTICOS	258
8.4.1	<i>Requisitos del sistema de iluminación.....</i>	259
8.4.2	<i>Requisitos de motorización: persianas y toldos.....</i>	259
8.4.3	<i>Requisitos del sistema de climatización</i>	261
8.4.4	<i>Requisitos del sistema de seguridad</i>	262
8.5.	ESPECIFICACIÓN DE REQUISITOS PARA UNA NUEVA APLICACIÓN	262
8.6.	MODELOS DOMÓTICOS GENERADOS AUTOMÁTICAMENTE A PARTIR DE LOS REQUISITOS... ..	263
8.6.1	<i>Modelos domóticos para el sistema de iluminación.....</i>	265
8.6.2	<i>Modelos domóticos para el sistema de motorización: toldos y persianas.....</i>	267
8.6.3	<i>Modelos domóticos para el sistema de climatización</i>	269
8.6.4	<i>Modelos domóticos para el sistema de seguridad</i>	271
8.7.	GENERACIÓN DE CÓDIGO A PARTIR DE LOS MODELOS DE DOMÓTICA	272
8.8.	ENCUESTA DE VALIDACIÓN	274
8.8.1	<i>Limitaciones del estudio.....</i>	283
8.9.	CONCLUSIONES Y APORTACIONES AL ESTADO DEL ARTE.....	284
CAPÍTULO 9 ANÁLISIS DE RESULTADOS, CONCLUSIONES Y VÍAS FUTURAS.....		287
9.1.	INTRODUCCIÓN.....	287
9.2.	ANÁLISIS DE LA CONSECUCIÓN DE OBJETIVOS	287
9.3.	CONCLUSIONES FINALES.....	290
9.4.	CONTRASTE DE RESULTADOS DE LA INVESTIGACIÓN.....	291
9.4.1	<i>Artículos en revistas internacionales indexadas en JCR.....</i>	292
9.4.2	<i>Artículos en revistas no indexadas</i>	292
9.4.3	<i>Capítulos de libro</i>	292
9.4.4	<i>Congresos y talleres internacionales.....</i>	293
9.4.5	<i>Congresos, talleres y reuniones científicas nacionales.....</i>	293
9.4.6	<i>Proyectos fin de carrera.....</i>	294
9.4.7	<i>Informes técnicos.....</i>	295
9.4.8	<i>Registros de la propiedad intelectual.....</i>	295
9.4.9	<i>Citas a los trabajos relacionados con la tesis doctoral.....</i>	295
9.5.	LINEAS DE TRABAJO FUTURO	302
BIBLIOGRAFÍA Y REFERENCIAS		305
ANEXO I. TRABAJOS EXCLUIDOS EN LA RSL SOBRE METAMODELOS DE REQUISITOS		327

ÍNDICE DE FIGURAS

FIGURA 1 EJEMPLO DE REQUISITO PARAMETRIZADO.	5
FIGURA 2 SEPARACIÓN DE INTERESES EN EL ANÁLISIS DE REQUISITOS [61].	8
FIGURA 3 RESUMEN DE LAS TÉCNICAS DEL ENFOQUE AORE [62].	8
FIGURA 4 VARIABILIDAD EN REQUISITOS TEXTUALES [276].	11
FIGURA 5 ÁRBOL DE CARACTERÍSTICAS DE UNA SPL DE SEGURIDAD DOMÉSTICA [276].	12
FIGURA 6 MODELADO CON UML VS. DSL [163].	13
FIGURA 7 ESQUEMA DEL PROCESO DSDM [FIGURA TOMADA DE [374]].	14
FIGURA 8 LA PROPUESTA EN EL MARCO DE MDA.	17
FIGURA 9 MODOS DE TRAZABILIDAD [FIGURA TOMADA DE [269]].	20
FIGURA 10 CARACTERÍSTICAS DEL ENFOQUE DE INVESTIGACIÓN CONSTRUCTIVO [197].	38
FIGURA 11 PROCESO DE INVESTIGACIÓN CONSTRUCTIVO [197].	38
FIGURA 12 LENGUAJES DE TRANSFORMACIÓN DE MODELOS UTILIZADOS EN CADA UNO DE LOS ENFOQUES DE DSDM.	94
FIGURA 13 CICLOS CON Y PARA REUTILIZACIÓN.	108
FIGURA 14 MODELO EN ESPIRAL DE UN PROCESO GENERAL DE IR.	110
FIGURA 15 RELACIÓN DE LAS ACTIVIDADES DE SIREN CON UN MODELO DE PROCESO GENERAL DE IR.	111
FIGURA 16 MODELO DE PROCESO DE SIREN CON LA NOTACIÓN SPEM.	111
FIGURA 17 DIAGRAMA DE ESTADOS DE UN REQUISITO.	122
FIGURA 18 UNA POSIBLE JERARQUÍA DE DOCUMENTOS DE REQUISITOS PARA SIREN.	128
FIGURA 19 JERARQUÍA DE ACTIVOS DE MAGERIT 1.0.	137
FIGURA 20 UN EXTRACTO DE LA CAPA 2 DE SISTEMAS DE INFORMACIÓN DE MAGERIT 1.0.	138
FIGURA 21 EJEMPLOS DE REQUISITOS DE SEGURIDAD DEL SISTEMA Y DEL SOFTWARE.	139
FIGURA 22 SIRENA COMO <i>PLUG-IN</i> DE REQUISITEPRO.	150
FIGURA 23 INTERFAZ PARA LA REUTILIZACIÓN DE REQUISITOS DE SIRENTOOL 2.0.	151
FIGURA 24 MODELO DE PROCESO PARA SIREN _{MD}	152
FIGURA 25 CONCEPTOS BÁSICOS EN LA INGENIERÍA DE MODELOS [33].	159
FIGURA 26 MODELO, METAMODELO Y META-LENGUAJE [170].	159
FIGURA 27 PROPUESTA DE MODELADO DE REQUISITOS.	160
FIGURA 28 RELACIÓN ENTRE NECESIDADES, CARACTERÍSTICAS Y REQUISITOS SOFTWARE SEGÚN LEFFINGWELL [186].	163
FIGURA 29 RELACIÓN ENTRE LOS NIVELES DE ABSTRACCIÓN DE LOS REQUISITOS EN REMM.	164
FIGURA 30 MODELO DE CALIDAD DEL ESTÁNDAR ISO 9126.	164
FIGURA 31 DEFINICIÓN DE NUEVOS ATRIBUTOS EN REMM.	166
FIGURA 32 TRAZABILIDAD EXTRA-REQUISITOS EN REMM.	168
FIGURA 33 RELACIONES DE TRAZA INTER-REQUISITOS EN REMM.	170
FIGURA 34 REQUISITOS REUTILIZABLES Y REQUISITOS DE PRODUCTO EN REMM.	173
FIGURA 35 REQUISITOS PARAMETRIZADOS.	174

FIGURA 36 PROCESO DE TRANSFORMACIÓN DE MODELOS.....	184
FIGURA 37 MARCO DE TRANSFORMACIÓN DE MODELOS. ÉSTA FIGURA ES UNA GENERALIZACIÓN AL DSDM DE LA PROPUESTA DEL MARCO DE MDA [170].....	184
FIGURA 38 VISIÓN GENERAL DEL ENFOQUE COMPLETO DE DSDM: ReToCo.	187
FIGURA 39 PROCESO DE INGENIERÍA DEL DOMINIO DE ReToCo.	188
FIGURA 40 REQUISITO REUTILIZABLE PARAMETRIZADO PARA LOS SISTEMAS DOMÓTICOS.....	189
FIGURA 41 MODELOS DE ENTRADA Y SALIDA EN LA TRANSFORMACIÓN M2M.	190
FIGURA 42 PROCESO DE INGENIERÍA DEL DOMINIO PARA LOS SISTEMAS DOMÓTICOS.	191
FIGURA 43 PROCESO DE INGENIERÍA DE LA APLICACIÓN DE ReToCo.	193
FIGURA 44 INSTANCIACIÓN DEL REQUISITO DE ILUMINACIÓN DE LA FIGURA 40.....	194
FIGURA 45 TRANSFORMACIÓN M2M DESDE LOS MODELOS DE REQUISITOS.	201
FIGURA 46 RELACIONES ENTRE LOS PARÁMETROS DEL REQUISITO DE ILUMINACIÓN Y LAS UNIDADES FUNCIONALES DE LOS SISTEMAS DOMÓTICOS.....	203
FIGURA 47 REGLA DE TRANSFORMACIÓN REQUIREMENT2DIMMABLELIGHT (1/3).....	204
FIGURA 48 REGLA DE TRANSFORMACIÓN REQUIREMENT2DIMMABLELIGHT (2/3).	205
FIGURA 49 REGLA DE TRANSFORMACIÓN REQUIREMENT2DIMMABLELIGHT (3/3).	206
FIGURA 50 MODELO DE LA APLICACIÓN DOMÓTICA QUE SE CORRESPONDE CON EL REQUISITO DE ILUMINACIÓN.....	207
FIGURA 51 RUBYTL TOP RULE.	208
FIGURA 52 METAMODELO PARA REPRESENTAR LAS TRAZAS HACIA DELANTE (TRACEMM).	209
FIGURA 53 CREACIÓN DEL MODELO DE TRAZA EN LA DEFINICIÓN DE TRANSFORMACIÓN REMM2HABITATION.....	210
FIGURA 54 DEFINICIÓN DE TRANSFORMACIÓN PARA LA FUSIÓN DE MODELOS.	211
FIGURA 55 EXTRACTO DE LA DEFINICIÓN DE TRANSFORMACIÓN TRACE2REMM (1/2).....	211
FIGURA 56 EXTRACTO DE LA DEFINICIÓN DE TRANSFORMACIÓN TRACE2REMM (2/2).....	212
FIGURA 57 TRANSFORMACIÓN M2T A PARTIR DE MODELOS DE REQUISITOS PARA REUTILIZACIÓN.	214
FIGURA 58 TRANSFORMACIÓN M2T A PARTIR DE MODELOS DE REQUISITOS CONREUTILIZACIÓN.	216
FIGURA 59 EJEMPLO DE FICHEROS GENERADOS PARA UN MODELO DE REQUISITOS PARA REUTILIZACIÓN.....	218
FIGURA 60 INFORME DE TRAZABILIDAD GENERADO AUTOMÁTICAMENTE MEDIANTE UNA TRANSFORMACIÓN M2T.....	219
FIGURA 61 FUNCIONALIDAD DE UNA HERRAMIENTA QUE SOPORTE EL DSDM [170].....	224
FIGURA 62 FLUJO DE TRABAJO DE UN PROYECTO EMF.....	230
FIGURA 63 MARCO PARA LA GENERACIÓN DEL EDITOR DE MODELOS DE REQUISITOS PARA REUTILIZACIÓN.....	230
FIGURA 64 CONEXIÓN DE <i>GMF TOOLING</i> Y <i>GMF RUNTIME</i> [338].	231
FIGURA 65 ARQUITECTURA DE REMM-STUDIO+.	232
FIGURA 66 REMMEXTRA: EDITOR DE ARTEFACTOS DE IR RELACIONADOS CON LOS REQUISITOS.....	233
FIGURA 67 REMMF, EDITOR DE MODELOS PARA REUTILIZACIÓN.....	234
FIGURA 68 PROPIEDADES DE UN REQUISITO SINCRONIZADAS CON EL EDITOR REMMEXTRA.	235
FIGURA 69 CATÁLOGO HOLIS EN EL REPOSITORIO.	236
FIGURA 70 CATÁLOGO NORMATIVO DEL REPOSITORIO.	237
FIGURA 71 REMMB, EDITOR DE MODELOS DE REQUISITOS CONREUTILIZACIÓN.....	237
FIGURA 72 BÚSQUEDA DE REQUISITOS EN FUNCIÓN DEL VALOR DE SUS ATRIBUTOS.....	239
FIGURA 73 SELECCIÓN DE REQUISITOS PARA SU REUTILIZACIÓN.....	240
FIGURA 74 SELECCIÓN DE UN ÚNICO REQUISITO PARA SU REUTILIZACIÓN.....	240
FIGURA 75 INSTANCIACIÓN DE LOS REQUISITOS PARAMETRIZADOS.....	241
FIGURA 76 INFORME SOBRE CONFLICTOS ENTRE LOS REQUISITOS REUTILIZADOS.....	242
FIGURA 77 ÁRBOL Y MATRIZ DE TRAZA ENTRE-REQUISITOS.....	242
FIGURA 78 ERRORES ENCONTRADOS EN EL MODELO DE REQUISITOS PARA REUTILIZACIÓN.	243
FIGURA 79 ERROR DE VALIDACIÓN DEL MODELO DEBIDO AL VALOR DEL PARÁMETRO.....	244

FIGURA 80 CONFIGURACIÓN DE LAS PREFERENCIAS DEL PLUG-IN MOFScripT.....	244
FIGURA 81 EJECUCIÓN DE LA TRANSFORMACIÓN MODELO A TEXTO.....	245
FIGURA 82 RESULTADO DE LA EJECUCIÓN DE LA TRANSFORMACIÓN MODELO A TEXTO.....	246
FIGURA 83 METAMODELO HABITATION. UN DSL PARA SISTEMAS DOMÓTICOS	253
FIGURA 84 FRAGMENTO DEL CATÁLOGO DE UNIDADES FUNCIONALES [296].....	254
FIGURA 85 VISIÓN GENERAL DE LA METODOLOGÍA DE DSDM PARA SISTEMAS DOMÓTICOS [296].....	256
FIGURA 86 GENERACIÓN DE MODELOS A PARTIR DE LOS REQUISITOS [301].....	256
FIGURA 87 METAMODELO DE REQUISITOS PROPUESTO POR LA DR. ROSIQUE [296]	257
FIGURA 88 MODELO DE REQUISITOS PARA EL SISTEMA DE ILUMINACIÓN.....	259
FIGURA 89 MODELO DE REQUISITOS PARA LAS PERSIANAS	260
FIGURA 90 MODELO DE REQUISITOS PARA LOS TOLDOS.....	261
FIGURA 91 MODELO DE REQUISITOS PARA EL SISTEMA DE CLIMATIZACIÓN.....	261
FIGURA 92 MODELO DE REQUISITOS PARA EL SISTEMA DE SEGURIDAD	262
FIGURA 93 MODELO DE REQUISITOS PARA UNA NUEVA APLICACIÓN DOMÓTICA	263
FIGURA 94 MODELO PARA LA INSTALACIÓN DE TRES PUNTOS DE LUZ REGULABLES.....	266
FIGURA 95 MODELO GENERADO PARA LA INSTALACIÓN DE TRES PUNTOS DE LUZ FIJOS.....	267
FIGURA 96 UNIDADES FUNCIONALES PARA LOS PUNTOS DE LUZ FLUORESCENTES.....	267
FIGURA 97 MODELO DOMÓTICO PARA LA INSTALACIÓN DE UNA PERSIANA VENECIANA.....	269
FIGURA 98 MODELO DOMÓTICO PARA LA INSTALACIÓN DE UN TOLDO.....	269
FIGURA 99 MODELO PARA LA INSTALACIÓN DE UN <i>FANCOIL</i> COMO DISPOSITIVO DE CLIMATIZACIÓN.	270
FIGURA 100 MODELOS PARA LA INSTALACIÓN DE LAS ALARMAS: HUMO, INCENDIO, INUNDACIÓN E INTRUSIÓN.....	271
FIGURA 101 ESQUEMA DE TRANSFORMACIONES DE MODELOS DESDE LOS REQUISITOS HASTA EL CÓDIGO.	272
FIGURA 102 EJEMPLO DE REGLA DE TRANSFORMACIÓN DEL MODELO DEL DSL AL MODELO DE COMPONENTES.....	273
FIGURA 103 DIAGRAMA DE CAJAS PARA CADA PREGUNTA.....	278
FIGURA 104 FRECUENCIA DE RESPUESTAS.....	278
FIGURA 105 ESTADÍSTICA DESCRIPTIVA POR GRUPOS.....	282
FIGURA 106 FRECUENCIA DE LAS RESPUESTAS POR GRUPOS.....	282

ÍNDICE DE TABLAS

TABLA 1 FICHA DEL PROYECTO MENHIR	26
TABLA 2 FICHA DEL PROYECTO DOLMEN	27
TABLA 3 FICHA DEL PROYECTO PROFIT	28
TABLA 4 FICHA DEL PROYECTO DYNAMICA	28
TABLA 5 FICHA DEL PROYECTO META	29
TABLA 6 FICHA DEL PROYECTO PEGASO	30
TABLA 7 FICHA DEL PROYECTO GARTIC	31
TABLA 8 ESTRATEGIA DE INVESTIGACIÓN DE LA TESIS.....	37
TABLA 9 APLICACIÓN DEL PROCESO DE INVESTIGACIÓN CONSTRUCTIVO A ESTA TESIS.	40
TABLA 10 ETAPAS DEL PROCESO DE UNA RSL	41
TABLA 11 TAXONOMÍA DE RELACIONES DE TRAZA SEGÚN SOMMERVILLE Y SAWYER.	55
TABLA 12 MARCO DE COMPARACIÓN DE LOS METAMODELOS DE REQUISITOS BASADO EN LA TRAZABILIDAD DE REQUISITOS.....	56
TABLA 13 CRITERIO DE CALIDAD PARA EL LUGAR DE PUBLICACIÓN.....	59
TABLA 14 RESULTADOS DE LAS BÚSQUEDAS DE METAMODELOS DE REQUISITOS.	62
TABLA 15 TRABAJOS INCLUIDOS EN LA RSL DE METAMODELOS DE REQUISITOS (1/3).....	65
TABLA 16 TRABAJOS INCLUIDOS EN LA RSL DE METAMODELOS DE REQUISITOS (2/3).....	66
TABLA 17 TRABAJOS INCLUIDOS EN LA RSL DE METAMODELOS DE REQUISITOS (3/3).....	67
TABLA 18 CRITERIOS DE CALIDAD DE LOS TRABAJOS QUE PROPONEN METAMODELOS DE REQUISITOS....	71
TABLA 19 VALORACIÓN DE LOS TRABAJOS SOBRE METAMODELOS DE REQUISITOS.	72
TABLA 20 COMPARATIVA DE LOS METAMODELOS DE REQUISITOS.....	73
TABLA 21 PROPUESTAS DE TRANSFORMACIONES A PARTIR DE LOS REQUISITOS.	79
TABLA 22 RESULTADOS DE LAS BÚSQUEDAS DE TRANSFORMACIONES A PARTIR DE MODELOS DE REQUISITOS	86
TABLA 23 ARTÍCULOS CANDIDATOS NO SELECCIONADOS (1/2)	87
TABLA 24 ARTÍCULOS CANDIDATOS NO SELECCIONADOS (2/2).....	88
TABLA 25 TRABAJOS SELECCIONADOS (1/2)	91
TABLA 26 TRABAJOS SELECCIONADOS (2/2).....	92
TABLA 27 CRITERIOS DE CALIDAD DE LOS TRABAJOS SELECCIONADOS.	95
TABLA 28 EVALUACIÓN DE LA CALIDAD DE CADA TRABAJO SELECCIONADO.....	96
TABLA 29 MODELO DE ROLES DE SIREN	112
TABLA 30 RESUMEN PROYECTO PILOTO GARTIC EMPRESA E1 EN EL DESARROLLO PARA REUTILIZAR.	142
TABLA 31 RESUMEN PROYECTO PILOTO GARTIC EMPRESA E2 EN EL DESARROLLO PARA REUTILIZAR.	143
TABLA 32 RESULTADOS DE LAS EMPRESAS E1 Y E2 EN GARTIC.	144
TABLA 33 OCHO CUESTIONES CLAVE EN REUTILIZACIÓN DE REQUISITOS.	145

TABLA 34 COMPARATIVA DE LAS HERRAMIENTAS CARE COMERCIALES RESPECTO A LAS CUESTIONES CLAVES DE REUTILIZACIÓN.....	149
TABLA 35 DEFINICIÓN DE UN LENGUAJE PARA MODELOS DE REQUISITOS.....	160
TABLA 36 RESTRICCIONES OCL EN EL MODELADO PARA REUTILIZACIÓN.....	176
TABLA 37 RESTRICCIONES OCL EN EL MODELADO CON REUTILIZACIÓN.....	177
TABLA 38 RESTRICCIONES OCL EN REMM.....	177
TABLA 39 IMPLEMENTACIÓN DE SIREN EN REMM.....	180
TABLA 40 TRAZABILIDAD PRE- Y POST-RS EN REMM.....	181
TABLA 41 CORRESPONDENCIA ENTRE LA PLANTILLA DEL IEEE-830 Y REMM.....	217
TABLA 42 RESUMEN DE TRANSFORMACIONES DE MODELOS A PARTIR DE MODELOS DE REQUISITOS....	220
TABLA 43 HERRAMIENTAS PARA EL DSDM.....	225
TABLA 44 GESTIÓN DE REQUISITOS EN LAS HERRAMIENTAS PARA EL DESARROLLO DE APLICACIONES SIGUIENDO EL ENFOQUE DE DSDM.....	227
TABLA 45 CUESTIONARIO.....	277
TABLA 46 RESPUESTAS AL CUESTIONARIO.....	277
TABLA 47 ESTADÍSTICA DESCRIPTIVA PARA LOS DATOS DE LA TABLA 46.....	278
TABLA 48 FRECUENCIA DE RESPUESTA DE CADA PREGUNTA.....	279
TABLA 49 NÚMERO DE PUBLICACIONES Y TRABAJOS REALIZADOS POR TIPOS.....	291
TABLA 50 ARTÍCULOS EN REVISTAS INDEXADAS EN JCR.....	292
TABLA 51 ARTÍCULOS EN REVISTAS NO INDEXADAS.....	292
TABLA 52 CAPÍTULOS DE LIBRO.....	292
TABLA 53 CONGRESOS Y TALLERES INTERNACIONALES.....	293
TABLA 54 CONGRESOS, TALLERES Y REUNIONES CIENTÍFICAS NACIONALES. (1/2).....	293
TABLA 55 CONGRESOS, TALLERES Y REUNIONES CIENTÍFICAS NACIONALES. (2/2).....	294
TABLA 56 PROYECTOS FINAL DE CARRERA.....	294
TABLA 57 INFORMES TÉCNICOS.....	295
TABLA 58 TRABAJOS INCLUIDOS EN EL REGISTRO DE LA PROPIEDAD INTELECTUAL.....	295
TABLA 59 TRABAJOS QUE REFERENCIAN A SIREN (1/5).....	296
TABLA 60 TRABAJOS QUE REFERENCIAN A SIREN (2/5).....	297
TABLA 61 TRABAJOS QUE REFERENCIAN A SIREN (3/5).....	298
TABLA 62 TRABAJOS QUE REFERENCIAN A SIREN (4/5).....	299
TABLA 63 TRABAJOS QUE REFERENCIAN A SIREN (5/5).....	300
TABLA 64 TRABAJOS QUE REFERENCIAN A REMM (1/3).....	300
TABLA 65 TRABAJOS QUE REFERENCIAN A REMM (2/3).....	301
TABLA 66 TRABAJOS QUE REFERENCIAN A REMM (3/3).....	302

RESUMEN

En esta tesis doctoral se presenta una propuesta de Ingeniería de Requisitos dirigida por modelos. Para ello, se ha definido: 1) un metamodelo de requisitos, que permite representar una especificación textual de requisitos; 2) un proceso de Ingeniería de Requisitos basado en reutilización, para guiar en la definición de modelos de requisitos *para* y *con* reutilización, y 3) un proceso para generar, los modelos de diseño de una aplicación, conformes a un DSL, a partir de los modelos de requisitos. Siguiendo el enfoque de modelado específico de dominio, a partir del modelo de la aplicación, se puede generar el código del sistema especificado. Este proceso general, se ha particularizado al dominio de los sistemas domóticos, para el que se ha definido un catálogo de requisitos reutilizables y las transformaciones, modelo a modelo, que permiten generar los modelos de diseño de una aplicación domótica, a partir del modelo de requisitos.

Palabras clave:

Ingeniería de requisitos, reutilización de requisitos, metamodelo de requisitos, transformaciones de modelos, trazabilidad de requisitos, desarrollo de software dirigido por modelos, ingeniería de requisitos dirigida por modelos, sistemas domóticos.

ABSTRACT

In this doctoral thesis a model driven Requirements Engineering approach is presented. The proposal involves the definition of: 1) a requirements metamodel, which allows analysts to represent a textual requirements specification, 2) a Requirements Engineering process based on reuse, which leads engineers through the definition of both requirements models *for* and *with* reuse, and 3) a development process to generate design application models, conforming a DSL, from requirements models. By following the domain specific modeling approach, the application code can be generated from these application models. This general process has been particularized to the home automation domain, for which a reusable requirements catalogue and transformation rules (from requirements models to home automation models) have been defined.

Keywords:

Requirements Engineering, requirements reuse, requirements metamodel, model transformation, requirements traceability, model driven software development, model driven requirements engineering, home automation systems.

Capítulo 1

Introducción y análisis de objetivos

1.1. Introducción

El objetivo del desarrollo del software es producir software de calidad—de acuerdo a los plazos y al presupuesto establecidos—que cumpla las necesidades reales de los clientes. El éxito de los proyectos depende en gran medida de que la gestión de requisitos se aborde de forma efectiva puesto que los errores en esta etapa del desarrollo son los más frecuentes y los más costosos de arreglar [186]. Uno de los estudios más referenciados, la serie de informes CHAOS llevado a cabo por el Standish Group, revelan que los fallos en el software se deben principalmente a la naturaleza cambiante de los requisitos y especificaciones, y a la incompletitud de los mismos.

Otros estudios [68, 328] han demostrado que la aplicación de un proceso de requisitos efectivo reporta beneficios sustanciales a las organizaciones de desarrollo de software. Hay dos estrategias que deberían tenerse en cuenta especialmente puesto que pueden contribuir a mejorar la productividad del desarrollo y la calidad de los productos software obtenidos, estas son: la *reutilización de requisitos* [55] y la *trazabilidad de requisitos* [76, 185]. Por un lado, es reconocido que cuanto más alto es el nivel en el que se lleva a cabo la reutilización, mayores son sus beneficios [55, 319]. Además, en un estudio sobre las tendencias presentes y futuras de la investigación en Ingeniería de Requisitos (IR) [60] se ha señalado la reutilización de requisitos como una de las necesidades más apremiantes y uno de los mayores desafíos, cuya solución posiblemente tenga el mayor impacto en la investigación y la práctica de la Ingeniería del Software. Por otro lado, la trazabilidad de requisitos tiene que ver con la habilidad de describir y seguir la vida de un requisito tanto hacia delante como hacia atrás [111]. La importancia de la trazabilidad de requisitos podemos inferirla por la necesidad de alinear las prácticas de desarrollo de software con los estándares de calidad como CMMI [309] que incluyen la trazabilidad como un elemento que es necesario gestionar en el proceso de desarrollo. Cabe señalar que una gestión adecuada de la trazabilidad de los requisitos beneficiará la reutilización de los mismos puesto que se pueden identificar paquetes de requisitos que están relacionados [360].

Por otro lado, actualmente la comunidad de Ingeniería del Software está prestando especial atención a los modelos como eje central del proceso de desarrollo de software. El

Desarrollo de Software Dirigido por Modelos (DSDM) ha surgido como un enfoque prometedor para abordar la complejidad inherente a los modernos sistemas basados en el software elevando el nivel de abstracción en el cual se concibe, implementa y desarrolla el mismo [307]. El modelado y las transformaciones de modelos constituyen las piezas claves del DSDM.

Recientemente se ha reconocido que la comunidad de DSDM no ha prestado especial atención a la investigación relacionada con los requisitos y que existen retos importantes en relación con el modelado de requisitos que deberían abordarse como son la integración de las especificaciones de requisitos en el modelado tradicional y la gestión de la trazabilidad entre los requisitos y los modelos de diseño [335]. Con esta tesis doctoral se trata de contribuir en esta línea, proporcionando las herramientas que permitan la integración de los requisitos textuales, que es el formato preferido en la industria [63], en el DSDM utilizando las herramientas propias de este enfoque: modelado y transformaciones de modelos. Por tanto, los requisitos serán modelados de manera que puedan trazarse hacia el resto de artefactos del desarrollo cumpliendo con las demandas de los estándares de calidad y prestando especial atención a la reutilización de requisitos con el objeto de mejorar la calidad y productividad del proceso de desarrollo.

El interés de este trabajo se justifica en la importancia de la IR en el proceso de desarrollo de software utilizando un lenguaje natural controlado como formato de especificación de los requisitos, los beneficios que aporta la reutilización de requisitos para mejorar la productividad del proceso de IR y la necesidad de integrar las especificaciones de requisitos en el DSDM adoptando un enfoque de IR dirigido por modelos. A continuación justificamos con más detalle cada una de estas líneas con el fin de establecer el marco necesario para la definición de los objetivos de esta tesis doctoral.

1.1.1. La importancia de la Ingeniería de Requisitos en el proceso de desarrollo del software

Según Pressman [279] uno de los retos de la Ingeniería del Software es saber cómo asegurar que se ha especificado un sistema que cumple correctamente las necesidades y expectativas del cliente. Si bien, de acuerdo con este autor, no existe una respuesta infalible a esta difícil pregunta, la mejor solución es contar con un proceso de IR sólido.

Sommerville [327] define la IR como el proceso de establecer los servicios que el cliente requiere de un sistema y las restricciones bajo las cuales opera y es desarrollado. Esto implica, el uso de técnicas repetibles y sistemáticas para asegurar que los requisitos del sistema son completos, consistentes, relevantes, etc. [173]. Para Sommerville [327] los requisitos propiamente dichos son las descripciones de los servicios y restricciones del sistema que se generan durante el proceso de IR. Según el glosario de términos del IEEE [127] un requisito es una condición o capacidad que un usuario demanda para solucionar un problema o alcanzar un objetivo. Existen otras definiciones pero todas comparten estas ideas.

Hay muchos problemas que se derivan de errores en los requisitos y se estima que el coste de arreglar un error en los requisitos puede ser hasta 100 veces mayor que el coste de arreglar un simple error de programación [173]. Un estudio reciente [333] revela que el 44% de los proyectos analizados eran cuestionables (terminaron tarde, excedieron el

presupuesto y/o con menos características y funciones de las requeridas) mientras que el 24% fallaron (se cancelaron antes de terminar o se terminaron pero nunca llegaron a utilizarse). Un estudio anterior [332] establecía como causa principal del fracaso o puesta en entredicho de los proyectos la pobreza en la definición de los requisitos. Aunque la lista de los diez factores de éxito para los proyectos que establece el Standish Group ha sido actualizado en este último informe [333] se sigue destacando el papel primordial de los requisitos.

No obstante, a pesar del impacto negativo de una mala gestión de requisitos en el coste y funcionalidad del desarrollo de sistemas [103, 104, 324], una encuesta sobre el estado de la práctica en IR [244] revela que el 52% de los encuestados pensaban que su compañía no dedicaba el suficiente esfuerzo a las tareas de IR. Este trabajo volvió a repetirse en el 2008 [203] para comprobar si se había producido algún cambio significativo en los seis años que separan ambas encuestas y desafortunadamente el grado de satisfacción de los encuestados respecto a la dedicación a la IR seguía siendo el mismo.

Pensamos que esta situación podría deberse a que los directivos encuentran dificultades para poner en práctica los resultados de las investigaciones. La realidad es que existe una distancia entre el estado del arte y el estado de la práctica [286], y coincidimos con Wiegers [359], en que para acortar esta distancia no es necesario inventar un nuevo método, sino hacer más atractivos y ágiles los métodos actuales, en definitiva, hacerlos fáciles de poner en práctica. Otros trabajos, como el presentado en [322], también ponen de manifiesto los beneficios de un *enfoque práctico*.

Un proceso de IR es un conjunto estructurado de actividades que se siguen para derivar, validar y mantener un documento de requisitos del sistema [173]. Este proceso es iterativo y comprende las actividades de elicitación, análisis y negociación, documentación y validación de requisitos. La elicitación de requisitos tiene que ver con el descubrimiento de los requisitos del sistema en relación con el problema que se va a resolver, los servicios que se van a ofrecer, el rendimiento necesario, las restricciones de hardware, entre otros. Esta actividad no sólo requiere preguntar a las personas implicadas (*stakeholders*) que es lo que quieren, sino realizar un análisis cuidadoso del contexto en el que se va a utilizar el sistema que se va a desarrollar: la organización, el dominio de la aplicación y los procesos de negocio. El objetivo del análisis y negociación de requisitos es obtener un conjunto consensuado de requisitos completos y consistentes. Durante la actividad de análisis se deben detectar requisitos ambiguos, conflictivos, redundantes, no realistas o que se han olvidado. En el caso de que existan conflictos, los *stakeholders* deben negociar y llegar a compromisos y acuerdos sobre las modificaciones que se deben realizar en la especificación de requisitos. Los requisitos aceptados deben ser documentados con el suficiente nivel de detalle de manera que sea entendible por todos los *stakeholders*. Por último, la actividad de validación de requisitos, como su propio nombre indica, tiene por objetivo validar la especificación de requisitos para certificar que representan una descripción aceptable del sistema que se va a implementar. Se debe comprobar la consistencia, completitud y precisión del documento de requisitos.

1.1.1.1. Documentación de requisitos en lenguaje natural

El documento de requisitos es la pieza clave que se utiliza para comunicar los requisitos entre los stakeholders. Cada organización debe establecer la estructura adecuada para asegurar que se incluya toda la información esencial. Uno de los formatos estándar más conocidos es el IEEE-830 [131] que contiene consejos sobre la forma de escribir los requisitos y los criterios de calidad que deben tenerse en cuenta. De acuerdo con el estándar, los requisitos deben ser completos, no ambiguos, correctos, consistentes, verificables, priorizados, modificables y trazables. La mayoría de las organizaciones utilizan el lenguaje natural para escribir los requisitos del sistema complementados con diagramas [173] como pueden ser diagramas de casos de uso o diagramas de secuencia UML. La experiencia ha demostrado que los modelos no deben servir para reemplazar una especificación de requisitos textual [72]. Si se utilizan hay que procurar utilizar una notación adecuada, que el cliente entienda, y deben ser referenciados convenientemente desde los requisitos textuales [71]. Estudios recientes demuestran que los requisitos textuales siguen siendo el formato preferido en la industria [63, 203] puesto que es la notación que generalmente entiende cualquier lector potencial. A pesar de esta ventaja, los requisitos escritos en lenguaje natural pueden ser ambiguos o difíciles de entender.

Si bien autores como Sommerville y Sawyer [303] proponen guías para evitar los problemas de ambigüedad y complejidad de los requisitos, existen otras propuestas que apuntan como solución al uso de un lenguaje natural controlado (*Controlled Natural Language* [4]). Los CNLs son subconjuntos del lenguaje natural que restringen sus gramáticas y diccionarios para eliminar tanto la ambigüedad como la complejidad. Tradicionalmente, los CNLs se clasifican fundamentalmente en dos grandes categorías: aquellos que mejoran la legibilidad, en particular para los lectores no nativos, y aquellos que mejoran el procesamiento computacional de un texto. Por tanto, el beneficio fundamental de utilizar un CNL es que permite representar los requisitos de una forma cercana a la representación natural del hombre (el lenguaje natural) pero a la vez manteniéndola libre de ambigüedades e imprecisiones, por tanto se minimizan las malas interpretaciones puesto que los usuarios pueden validarlas fácilmente [271]. Por ejemplo, se podría utilizar como lenguaje de especificación de requisitos el lenguaje ACE (*Attempto Controlled English*) [2] o un lenguaje de especificación basado en requisitos parametrizados (*boilerplate requirements* [122]) que son plantillas de requisitos que restringen la forma en la que se escriben utilizando parámetros que pueden estar también limitados a un conjunto finito de valores pertenecientes al dominio del problema. La definición de una especificación de requisitos a partir de requisitos parametrizados implica la selección del requisito parametrizado más adecuado y la instanciación de sus parámetros con los valores adecuados. El uso de requisitos parametrizados favorece la reutilización de requisitos al tiempo que mejora la calidad de las especificaciones de requisitos definidas a partir de ellos. La Figura 1 muestra un ejemplo de requisito parametrizado.

La definición de lenguajes de especificación de requisitos es una línea activa de trabajo en algunos dominios como el de los sistemas embebidos (por ejemplo, CESAR Project [216], ReDSeeDS Project [9, 153]), los sistemas interactivos (por ejemplo, ProjectIT-RSL [20]) o los sistemas de control telescópico [160]. En esta última propuesta los autores

señalan la necesidad de que las herramientas de modelado permitan definir requisitos parametrizados y cuenten con un repositorio de artefactos de modelado reutilizables. Nosotros tendremos en cuenta la definición de lenguajes de requisitos en la forma de catálogos de requisitos reutilizables que incluya requisitos parametrizados en nuestra propuesta puesto que el uso de CNL parece la forma más efectiva de conseguir transformaciones automáticas de las especificaciones de requisitos a otros modelos [156, 368].

```
Requisito parametrizado (boilerplate):  
The HA system shall provide a [MECHANISM] to [ACTION] [X]  
[TYPE] light points.  
Instanciación del requisito:  
[MECHANISM] = PUSH_BUTTON  
[ACTION] = REGULATE  
[X] = 3.0  
[TYPE] = FLUORESCENT.  
dando como resultado el requisito:  
"The HA system shall provide a PUSH_BUTTON to REGULATE 3  
FLUORESCENT light points."
```

Figura 1 Ejemplo de requisito parametrizado.

1.1.1.2. Algunos enfoques para abordar la Ingeniería de Requisitos

El enfoque tradicional de IR que hemos planteado hasta el momento (y que el lector puede ampliar en [186, 293, 360]) trata de identificar tanto los requisitos funcionales como los no funcionales del sistema que se va a desarrollar atendiendo a *qué* tiene que hacer el sistema y no *cómo* tiene que hacerlo. A continuación presentamos brevemente otras propuestas que varían en su “orientación” lo que influye en la forma en la que se estructura la información necesaria para obtener la especificación de requisitos: puntos de vista, objetivos o aspectos.

Ingeniería de Requisitos orientada a puntos de vista (VORE, Viewpoint-Oriented RE)

Un enfoque de IR basado en puntos de vista se sustenta en la idea de que los requisitos del sistema no pueden extraerse atendiendo a una única perspectiva sino que es necesario recopilar y organizar los requisitos atendiendo a diferentes puntos de vista (*viewpoints*). Un punto de vista es una encapsulación de una información parcial sobre los requisitos del sistema. La información procedente de distintos puntos de vista se tiene que integrar y reconciliar para formar la especificación final del sistema [330].

En los métodos VORE que se han propuesto existen dos tipos diferentes de puntos de vista: asociados a los stakeholders (por ejemplo, el punto de vista de los clientes o del personal encargado de manejar el sistema), y asociados con el conocimiento organizacional o del dominio que restringen los requisitos del sistema (por ejemplo, el punto de vista de la base de datos, de la red o de la normativa de la empresa). Una fase esencial en cualquier método orientado a puntos de vista es seleccionar los puntos de vista críticos, puesto que hay que llegar a un compromiso entre las ventajas de añadir un nuevo punto de vista y el coste de análisis y los problemas de gestión de la información.

El enfoque orientado a puntos de vista se ha aplicado a distintas etapas del proceso de IR, como la etapa de elicitación donde destacan las propuestas del método VORD (*Viewpoint-Oriented Requirements Definition*) [173] y el modelo PREView (*Process and Requirements Engineering Viewpoints*) [331], o la etapa de negociación, donde cabe destacar los trabajos de Nuseibeh *et al.* [251].

Ingeniería de Requisitos orientada a objetivos (GORE, Goal-Oriented RE)

La IR orientada a objetivos utiliza el concepto de meta (*goal*) para la elicitación, elaboración, estructuración, especificación, análisis, negociación, documentación y modificación de los requisitos [175]. Una meta es un objetivo que el sistema debería alcanzar. La palabra sistema comprende tanto el software que se va a desarrollar como su entorno. La satisfacción de estas metas requiere la cooperación de *agentes* (o componentes activos) como son los seres humanos, dispositivos y el software. Estos agentes tienen un comportamiento que podrían restringir para asegurar las limitaciones que les hayan impuesto. Las metas cuyo cumplimiento es responsabilidad de un único agente en el software que se va a desarrollar es un requisito, mientras que si es responsabilidad de un único agente en el entorno del software que se va a construir es una suposición (*assumption*). A diferencia de los requisitos, las suposiciones serán satisfechas gracias a, por ejemplo, normas o regulaciones de la organización o leyes físicas. “Los requisitos, de hecho, ‘implementan’ metas de la misma forma que los programas implementan especificaciones de diseño” [176].

Entre las propuestas enmarcadas en GORE cabe destacar: 1) *NFR framework* [65] que se centra en el modelado y análisis de los requisitos no funcionales; 2) la metodología KAOS (*Knowledge Acquisition in auTOMated Specification* [70] o *Keep All Objectives Satisfied* [176]) que permite derivar gradualmente los requisitos operacionales de un sistema a partir de las metas subyacentes mediante la especificación de diferentes modelos que se relacionan mediante reglas de consistencia inter-modelo (modelo de metas, modelo de objetos, modelo de responsabilidades de los agentes, modelo de interfaz de los agentes y modelo de operaciones) y 3) *i* framework* [366] cuyo objetivo es apoyar en la fase inicial del proceso de IR, entendiendo que esta fase es la encargada de modelar y analizar los intereses de los stakeholders y cómo se podrían abordar o conciliar mediante distintas alternativas del sistema y del entorno. Son dos los componentes de modelado que conforman el framework i*: el modelo de dependencia estratégica (*Strategic Dependency model*) y el modelo de razones estratégicas (*Strategic Rationale model*). El primero se utiliza para describir las relaciones entre varios actores en un contexto organizacional. El segundo se utiliza para describir los intereses de los stakeholders y cómo podrían abordarse mediante distintas configuraciones de sistemas y entornos.

De acuerdo con la tesis de Pimenta [268] los enfoques VORE y GORE pueden complementarse, en concreto los métodos PREview y KAOS, de manera que se salven sus deficiencias y se sumen las ventajas que cada uno de estos enfoques aportan. Por un lado KAOS ofrece un conjunto de heurísticas de elicitación de requisitos basadas en técnicas de descomposición de objetivos (o metas, *goals*) junto con mecanismos para verificar la completitud de un objetivo. Sin embargo, podría beneficiarse de algún tipo de organización de los requisitos elicitados de manera atractiva para los stakeholders. Por otro lado, PREview se centra precisamente en entender y controlar la complejidad

de los sistemas separando los intereses de varios actores y formalizando en métodos de análisis esta vista multi-perspectiva. Sin embargo, carece de heurísticas que guíen el proceso de elicitación de requisitos, confiando para ello en la interpretación de los intereses del sistema por parte de los analistas, lo que puede llevar a requisitos incompletos o mal definidos. En el enfoque híbrido que propone Pimenta PREview proporciona las bases para encapsular los requisitos de forma atractiva para el usuario y trata la complejidad del sistema, mientras que KAOS proporciona los mecanismos para la descomposición de los requisitos de forma sistemática.

Ingeniería de Requisitos orientada a aspectos (AORE, Aspect-Oriented RE)

A principios del año 2000, surge la IR orientada a aspectos que proporciona nuevos mecanismos de composición para especificar y razonar sobre dependencias que afectan de forma transversal a los elementos de una especificación de requisitos [282]. AORE forma parte de la disciplina del desarrollo de software orientado a aspectos donde se utiliza el principio de Ingeniería del Software de “separación de intereses” (*separation of concerns*) para mejorar la modularización de los requisitos y el código fuente. En resumen, siguiendo el principio de separación de intereses, se divide el dominio del problema en aspectos concretos que luego se estudian por separado para su propia consistencia. En el caso de los requisitos software se descompone la funcionalidad de la aplicación y se clasifican los requisitos en: características básicas (*core features*) e intereses transversales (*crosscutting concerns*). Las características básicas se pueden estructurar en módulos de la aplicación y los intereses transversales se pueden estructurar por tipos especializados (Figura 2, extraída de [61]). A continuación, se debe analizar y documentar el impacto de los intereses transversales en el contexto de las características básicas utilizando reglas de composición definidas para mejorar la completitud de las especificaciones de requisitos. Por tanto, AORE no reemplaza ninguna de las metodologías de IR existentes sino que más bien las complementa proporcionando técnicas para la especificación y análisis de requisitos. Como resultado se pueden obtener requisitos mejor estructurados y especificaciones más completas que son más fáciles de analizar y más baratas de mantener. La Figura 3 muestra gráficamente los componentes del enfoque AORE extraída de [62]. El lector puede encontrar una explicación más detallada de los conceptos y técnicas de AORE en [61, 62].

Los objetivos de AORE son similares a los del enfoque orientado a objetivos presentado anteriormente. La característica que distingue las técnicas de AORE es la separación explícita de las especificaciones de composición que, en las técnicas orientadas a objetivos permanecen mezcladas [282]. Existen algunas propuestas que integran ambos enfoques como la de Yu *et al.* [367] que muestran cómo pueden descubrirse los aspectos durante el análisis de requisitos orientado a objetivos. Además, la propuesta incluye un proceso sistemático para encontrar los aspectos a partir de las relaciones entre objetivos funcionales y no funcionales.

Rashid y Chitchyan [283] presentan una visión completa del estado del arte en el área de AORE y los retos pendientes que deberían abordarse en los últimos años. En este trabajo se destacan, por su madurez, las propuestas de: 1) Whittle *et al.* [358] en cuando al modelado de escenarios con aspectos, 2) Moreira *et al.* [227, 228, 284] en relación al modelado de requisitos volátiles, la separación de intereses multidimensionales y el uso de ARCADE para dar soporte a AORE, respectivamente, o

3) Jacobson *et al.* [143] sobre un enfoque de casos de uso orientado a aspectos. El lector también puede encontrar en [282] una extensa selección bibliográfica comentada de trabajos relacionados con AORE.

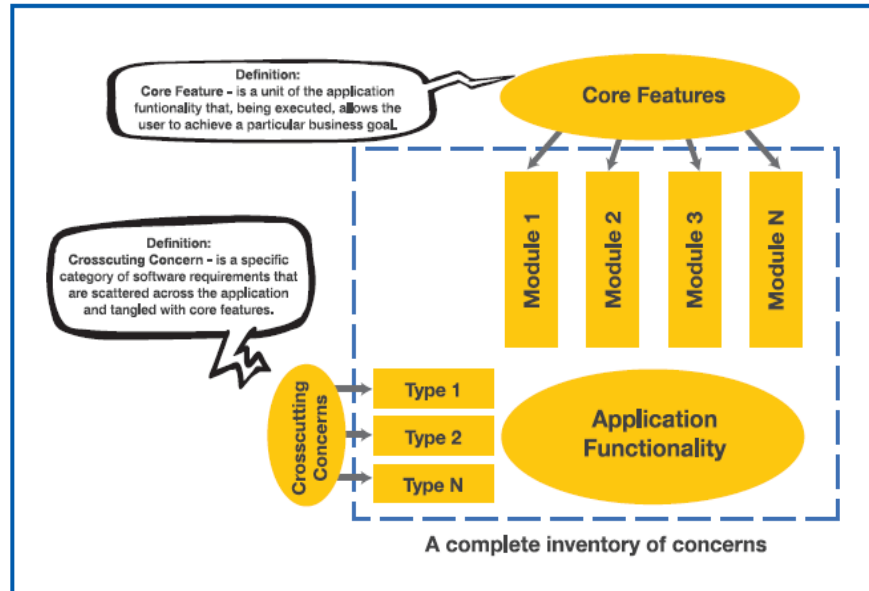


Figura 2 Separación de intereses en el análisis de requisitos [61].

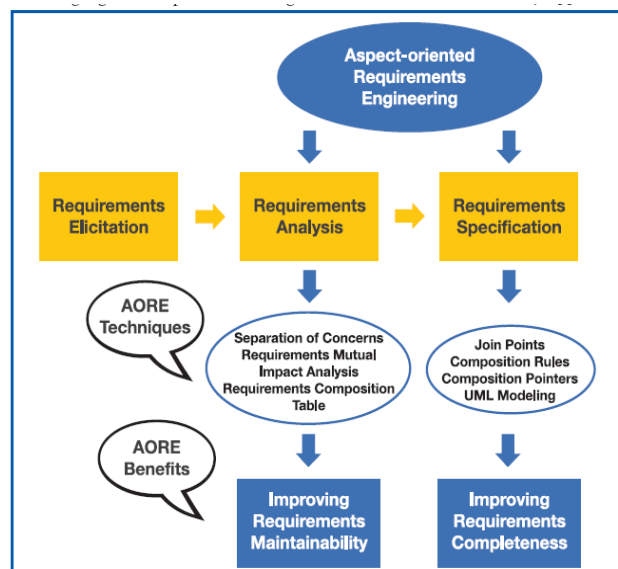


Figura 3 Resumen de las técnicas del enfoque AORE [62].

1.1.2. Ingeniería de Requisitos y reutilización

En la mayoría de las ingenierías, los sistemas se diseñan componiendo elementos que ya han sido utilizados en otros sistemas. Sin embargo, la Ingeniería del Software se ha centrado más en desarrollos originales a pesar de la naturaleza repetitiva del proceso de desarrollo de software. Pero ahora se reconoce que para conseguir un software mejor, más rápidamente y con un coste menor, es necesario utilizar un proceso de diseño basado en la reutilización sistemática del software [327]. Según Mili *et al.* [215] la reutilización es la única vía para desarrollar sistemas de información que tengan los niveles de calidad exigidos, dentro de las restricciones existentes de tiempo y presupuesto.

La reutilización del software se puede definir como el proceso de construir o ensamblar sistemas software a partir de componentes software predefinidos que han sido diseñados para ser reutilizados [210]. Estos componentes software pueden tener distinta naturaleza (código, diseños, patrones, etc.), pero antes de discutir qué es lo que se debería utilizar vamos a analizar los beneficios esperados de la reutilización. Según Meyer [151], de un software construido a partir de componentes reutilizables se puede esperar una disminución en el tiempo de desarrollo y el tiempo de llevar el producto al mercado, disminución del tiempo de mantenimiento, mejora en la eficiencia, fiabilidad y consistencia, además, al hacer el software reutilizable preservamos el conocimiento (*know-how*) de los mejores desarrolladores, transformando este recurso frágil en un valor permanente.

Un componente reutilizable podría ser el código fuente, pero los beneficios son mayores cuando se tiene una visión más amplia y de más alto nivel de lo que se puede reutilizar. Así, sería viable considerar como componentes reutilizables las especificaciones, diseños, planes de proyecto, documentación, etc. En definitiva, se podría producir cualquier entregable de un proyecto software a partir de componentes reutilizables [210]. Por tanto, aunque inicialmente el interés de la reutilización estuviera puesto en el código, diseños o arquitecturas, parece que la práctica de la reutilización se está moviendo hacia arriba en el proceso de desarrollo, de manera que actualmente podemos encontrar que se reutilizan los elementos más abstractos, incluyendo los requisitos [293, 343, 344]. Existe un amplio consenso (véase, por ejemplo, los trabajos [326] o [55]) en que los beneficios de la reutilización son mayores cuando se incrementa el nivel de abstracción, y no se reutiliza sólo código sino diseño y especificaciones. Según McClure [210] el tipo de componente reutilizable que estaría lo más alto en la lista serían esqueletos que podrían servir para producir cualquier tipo de documentación del proyecto. Estos esqueletos se pueden proporcionar como plantillas en un procesador de texto por lo que se pueden crear de forma rápida y sencilla y pueden estar disponibles en un catálogo reutilizable. El beneficio esperado no es sólo ahorrar tiempo sino también asegurar que se siguen los estándares corporativos y hace que se mantengan la consistencia entre los sistemas y procesos de desarrollo de software. Esto nos lleva a sistemas con mejor calidad que son más fáciles de entender, de mantener y reutilizar. Aunque McClure no lo nombra explícitamente, a este nivel podemos incluir las especificaciones de requisitos textuales reutilizables, que gracias a las herramientas de gestión de requisitos podemos manejar de forma más eficiente que con un procesador de

texto, aunque la mayoría de estas herramientas no soportan la reutilización sistemática de requisitos [208].

En la medida de nuestro conocimiento, son Rine y Nada [287] los únicos que muestran empíricamente que el nivel de reutilización determina la efectividad de las mejoras en productividad, calidad y tiempo de desarrollo (*time-to-market*), concluyendo que se obtienen mayores beneficios cuando la reutilización se considera durante las primeras etapas del desarrollo de software. En esta línea, autores como Nuseibeh y Easterbrook [250] o Zave [369] afirman que uno de los principales desafíos de la IR es la reutilización de modelos de requisitos. Se espera que se desarrollen modelos de referencia de requisitos en muchos dominios de aplicación, de manera que se reduzca drásticamente el esfuerzo de desarrollo de nuevas especificaciones de requisitos, facilitando además la selección de paquetes de software (COTS, *Comercial Off-The-Shelf Software*). En un estudio posterior sobre las tendencias presentes y futuras en la investigación en IR [60] se ha señalado la reutilización de requisitos como una de las necesidades más apremiantes y uno de los mayores desafíos, cuya solución posiblemente tenga el mayor impacto en la investigación y la práctica de la Ingeniería del Software. En consecuencia, bajo nuestro punto de vista, un trabajo práctico en IR debe tener en cuenta la reutilización de requisitos como pieza clave para reducir el esfuerzo y tiempo dedicado a esta fase y mejorar, por tanto, la productividad.

En la literatura se distinguen dos vías no disjuntas de reutilización: enfoques basados en generación y enfoques basados en composición: (1) los enfoques basados en generación se centran en la instanciación de abstracciones reutilizables, como por ejemplo en líneas de productos o en la instanciación de patrones; y (2) los enfoques basados en composición se centran en componer componentes reutilizables: se establecen relaciones de traza entre los requisitos entre sí y entre los requisitos y la arquitectura, implementación y pruebas que se construyen a partir de ellos, de manera que si un requisito del sistema es reutilizado, se puede acelerar el proceso de desarrollo posterior siguiendo las trazas y componiendo los elementos trazados. En reutilización de requisitos tenemos ejemplos de ambos enfoques, si bien quizás los enfoques generativos son más frecuentes [171]. El lector interesado puede consultar la sección 2.3 de la tesis doctoral de Joaquín Nicolás [245] donde se hace un estudio exhaustivo de los trabajos presentes en la literatura relacionados con la reutilización de requisitos en lenguaje natural, que es el formato de requisitos que vamos a utilizar en nuestra propuesta.

A pesar de la importancia de la reutilización de requisitos, a menudo se aborda de manera no sistemática [171, 319], en el sentido de que no se modelan las dependencias entre los requisitos en la mayoría de los casos. En consecuencia, cuando se reutiliza un requisito, los requisitos (o de manera más general, aquellos elementos como por ejemplo, stakeholders, casos de prueba, etc.) relacionados con él, no se tienen en cuenta de manera adecuada en el momento de la reutilización. De acuerdo con [171], estos enfoques no sistemáticos son propensos a errores y consumen mucho tiempo puesto que, al no modelarse de manera explícita las dependencias entre los requisitos no se reutilizan todos los requisitos relacionados con el requisito que se ha seleccionado para ser reutilizado o se copian demasiados requisitos. En consecuencia, el nuevo documento omitiría algunas características o contendría características inconsistentes o superfluas [171]. Esta tesis doctoral se apoya en la idea de que este problema se podría resolver

fácilmente siguiendo un enfoque de DSDM, en la sección 1.3.3 se detallan los beneficios esperados de la aplicación de las técnicas del DSDM a la IR.

11.2.1 Variabilidad en Líneas de Producto Software

En el campo de la reutilización del software, los enfoques de Líneas de Producto Software (SPL, *Software Product Line*) [53] han sido reconocidos como una técnica destacada para alcanzar los objetivos de reutilización. Los procesos implicados en la especificación de requisitos de una SPL son: ingeniería del dominio e ingeniería de la aplicación. Por un lado, los objetivos de la ingeniería del dominio (modelado *para* reuso) son la identificación y determinación de los requisitos comunes y los puntos de variación de la línea de producto. Por otro lado, el objetivo de la ingeniería de la aplicación (modelado *con* reuso) es la especificación de los requisitos de un producto adaptado a las necesidades del cliente utilizando la variabilidad de la familia de productos, esto es, la instanciación de los requisitos de la SPL para un producto concreto dentro de la familia.

La información acerca de la variabilidad se puede modelar junto con la especificación de requisitos o de forma separada [19]. En el primer caso, para documentar la variabilidad junto con la especificaciones textuales de requisitos, se debe utilizar un formato textual estructurado similar a los proporcionados por [108, 276] de manera que se hagan explícitos los puntos de variación y sus variantes (ver Figura 4). De esta forma se añade precisión y es posible documentar la variabilidad de forma más concisa y menos ambigua que si se utiliza un formato textual libre. El uso de un lenguaje estructurado a menudo permite reducir el tamaño de la especificación y ayuda a restringir la forma en la que los desarrolladores elaboran la especificación de requisitos [19].

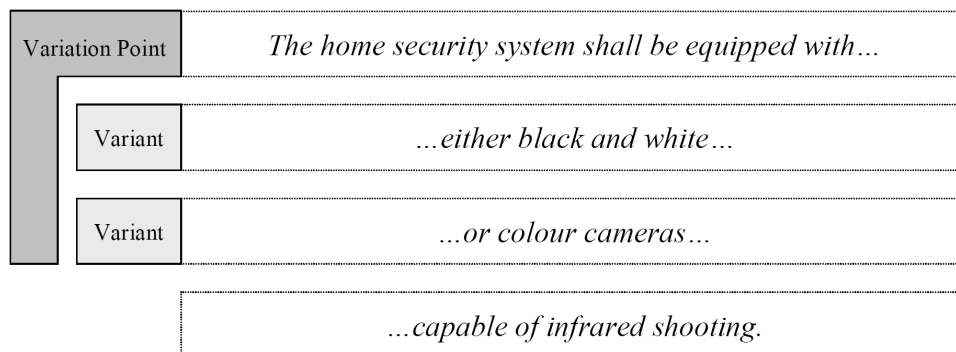


Figura 4 Variabilidad en requisitos textuales [276].

En el caso de separar la información de la variabilidad de la especificación de requisitos uno de los enfoques más reconocidos es ligar los requisitos con un modelo de características. Los modelos de características [56, 158] se utilizan para representar las propiedades del sistemas (características) que son relevantes para algún stakeholder y para capturar los puntos en común y los discriminantes entre los sistemas de una familia (ver Figura 5). Sin embargo, estos enfoques ortogonales raramente definen la relación que existe entre las características y los requisitos [314] y es posible que la

información sobre la variabilidad capturada en los modelos de características no sea suficiente, por ejemplo, para representar la variabilidad relacionada con las dependencias entre requisitos [44]. En consecuencia, algunos autores prefieren modelar la variabilidad junto con los artefactos implicados en la especificación de requisitos [44, 45, 47, 337]. Este es el enfoque que nosotros hemos seguido en nuestra propuesta, integrar la variabilidad de requisitos directamente en los modelos de requisitos textuales.

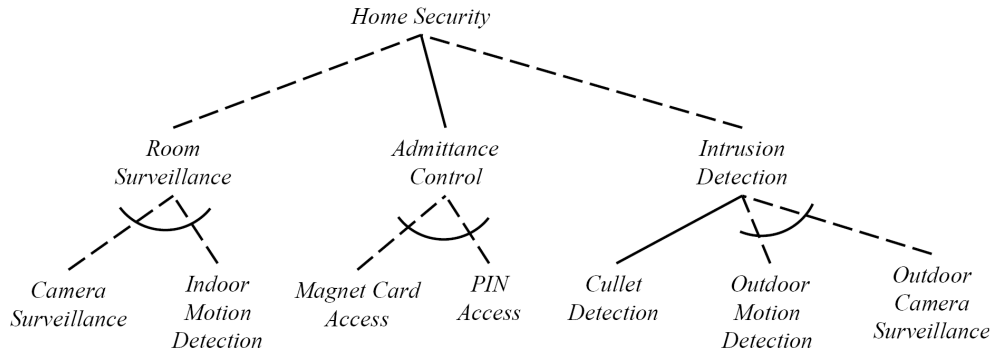


Figura 5 Árbol de características de una SPL de seguridad doméstica [276].

1.1.2.2. El método SIREN

Por último, no sólo hay que proporcionar técnicas para reutilizar sino que hay que integrarlas en un proceso de IR. En esta línea, para aprovechar los beneficios de la reutilización a nivel de los requisitos, el grupo de investigación de Ingeniería del Software (al que pertenece la doctoranda) propuso el método SIREN (*Simple REuse of RequiremeNts*) [343, 344]. SIREN es un enfoque práctico para la selección y especificación de los requisitos de un sistema basado en reutilización y en estándares de Ingeniería del Software. El método SIREN incluye un modelo de proceso en espiral, plantillas de documentos de requisitos, un repositorio de requisitos reutilizables organizado en catálogos y una herramienta CARE (*Computer-Aided Requirements Engineering*) de soporte denominada SirenTool [180]. Con SIREN se ha intentado acercar el estado del arte y el estado de la práctica en IR, proporcionando un enfoque flexible y fácil de poner en práctica para reducir el tiempo dedicado a la fase de IR y aumentar la productividad de los sistemas de información.

La experiencia obtenida a lo largo de la definición y aplicación del proceso SIREN [343, 344] e investigaciones sobre la reutilización de requisitos, nos llevaron a identificar ocho aspectos claves que, en nuestra opinión, deberían tenerse en cuenta en cualquier método de IR basado en reutilización que quiera tener éxito [339]. La experiencia ganada por la doctoranda durante la realización de estos trabajos ha servido como base para la realización de la presente tesis doctoral. Por tanto, el método SIREN servirá como marco para la definición del proceso de IR dirigido por modelos que proponemos, que debe permitir modelar la variabilidad de la especificación de requisitos y reutilizar parte de las especificaciones de manera sistemática. El método SIREN y los aspectos claves para la reutilización de requisitos serán presentados en detalle en el capítulo 4.

1.1.3. Ingeniería de Requisitos dirigida por modelos

La comunidad de Ingeniería del Software viene prestando especial atención a los modelos como eje central del proceso de desarrollo de software. Los modelos se refinan desde niveles de abstracción altos a otros más específicos mediante técnicas de transformación de modelos con el objeto de automatizar el proceso de desarrollo lo máximo posible. En este contexto, se fomenta la definición de lenguajes específicos de dominio (DSL, *Domain Specific Language*) frente a lenguajes de propósito general como UML, para modelar las soluciones utilizando conceptos cercanos al dominio del problema. Siguiendo el enfoque de modelado específico de dominio (DSM, *Domain Specific Modelling*) [163], una vez que el problema ha sido modelado utilizando un DSL, los productos finales se podrían generar automáticamente a partir de estas especificaciones de alto nivel utilizando generadores de código específicos de dominio.

Para ver de forma intuitiva la diferencia entre modelar un sistema con un lenguaje de propósito general y un DSL, pensemos en una pequeña aplicación para registrarse en una conferencia desde el teléfono móvil. Utilizando UML es posible que comenzásemos con la definición del caso de uso como el de la Figura 6 (lado izquierdo) para definir la funcionalidad, un diagrama de clases para definir la estructura estática de la aplicación y, si fuera necesario, diagramas de secuencia o de estados para especificar el comportamiento. Frente a este enfoque, en el DSM se utiliza un lenguaje de modelado que se ha creado de forma natural para el desarrollo de aplicaciones de telefonía móvil. En este caso, para modelar la aplicación, utilizaríamos directamente conceptos del dominio como SMS (Figura 6. lado derecho). Vemos como los conceptos propios de los servicios de los teléfonos móviles son también constructores del lenguaje. El lector interesado puede ver el ejemplo de forma detallada en el primer capítulo de [163].

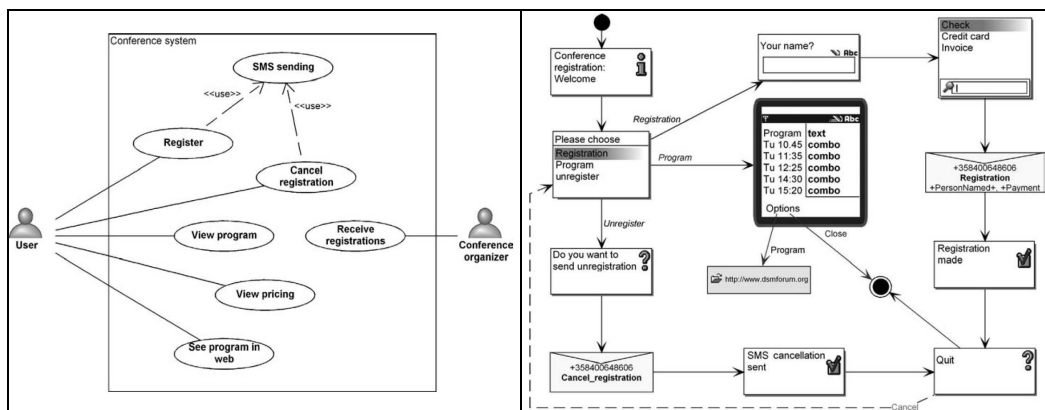


Figura 6 Modelado con UML vs. DSL [163].

A pesar de los avances del DSDM, la comunidad de IR parecía al margen de esta tendencia. Al comienzo de los trabajos que han dado lugar a esta tesis doctoral era difícil encontrar referencias a DSDM en las principales fuentes (revistas y congresos) relacionados con la IR y al contrario. Esta situación ha empezado a cambiar recientemente, y es en el año 2011 cuando se celebra el primer workshop titulado *Model Driven Requirements Engineering* (MoDRE) en paralelo con la principal conferencia

sobre IR, el RE (*Requirements Engineering Conference*). Este workshop inaugural nace con el propósito de proporcionar un foro para discutir los retos del DSDM para IR y reconoce que, dado el éxito que ha tenido el DSDM para el diseño y la implementación, la IR se podría beneficiar también de estas técnicas. Esta ha sido, como veremos posteriormente, la hipótesis de partida que sustenta esta tesis doctoral definida en el año 2007.

En las siguientes secciones vamos a: justificar en primer lugar la necesidad de definir modelos de requisitos para integrarlos en un proceso de DSDM (sección 1.1.3.1). A continuación, presentaremos las propuestas del OMG tanto para el DSDM como para el modelado de requisitos (sección 1.1.3.2). Existen otros enfoques para el DSDM como el de las Factorías de Software [113] pero nos centramos en las propuestas del OMG puesto que son las más populares. Después, introducimos las técnicas utilizadas en el DSDM, metamodelado (sección 1.1.3.3) y transformaciones de modelos (sección 1.1.3.4), y los beneficios esperados de la aplicación de dichas técnicas a la IR (sección 1.1.3.5). Un estudio detallado de las propuestas de metamodelos de requisitos y de transformaciones desde los requisitos se pospone hasta el Capítulo 3.

1.1.3.1. Justificación de la necesidad de los modelos de requisitos

Los requisitos representan la información que es elaborada en los modelos en las etapas posteriores del desarrollo. De ahí que exista una fuerte relación entre los requisitos y los modelos que especifican cómo se satisfacen. Sin embargo, este hecho a menudo se pasa por alto en los enfoques y herramientas del DSDM, en los que, mientras que se requiere que se definan claramente las transformaciones entre diferentes modelos, el modelo inicial del proceso de DSDM se deja que sea creado a parte por el analista en una especificación de requisitos separada (Figura 7). En consecuencia, los beneficios de utilizar DSDM para mejorar la eficiencia del desarrollo del software se pierde al principio del proceso [374].

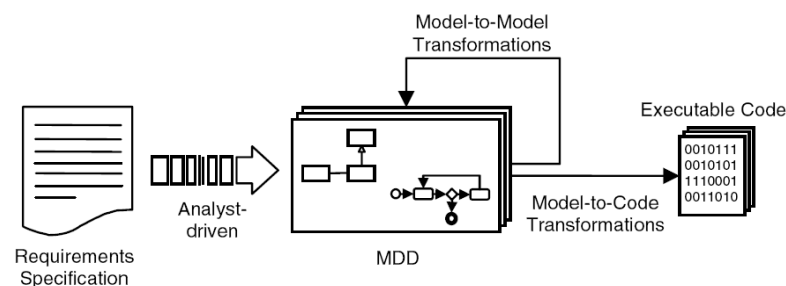


Figura 7 Esquema del proceso DSDM [figura tomada de [374]].

Un estudio reciente sobre el uso de las técnicas de IR en el DSDM [194] concluye que: 1) no existe todavía una solución completa que incluya los modelos de requisitos como parte del proceso de DSDM; 2) es escaso el soporte mediante herramientas para gestionar los modelos de requisitos y poder utilizarlos posteriormente de manera automática; 3) las actividades de IR casi nunca se integran en el flujo de transformaciones de modelos automáticas y generación de código; 4) las transformaciones de modelos automáticas se utilizan muy poco y 5) las relaciones de

traza desde los requisitos a modelos de análisis o diseño no se definen convenientemente. Los resultados de esta tesis tratan de aportar una solución a estos desafíos detectados por Loniewski *et al.*

Champeau y Rochefort [58] señalan que en el contexto de DSDM los requisitos deben ser modelados y se debe mantener una continuidad entre los requisitos y el modelo de implementación del sistema final. Debemos ser capaces de trazar los modelos, y los elementos que contienen, con los requisitos de manera que se pueda seguir el impacto de los requisitos a lo largo de todo el proceso de desarrollo. Los autores afirman que los requisitos se utilizan como elementos textuales entre los que existen relaciones simples pero que no existe ningún modelo que especifique dichas relaciones. En particular, destacan que existe un problema al intentar trasladar los requisitos a los modelos UML donde no existe el concepto de requisito. Por tanto, desde nuestro punto de vista, la definición de un metamodelo de requisitos servirá para hacer explícitas estas relaciones y para poner los requisitos al mismo nivel que los modelos UML.

Schätz *et al.* [304] ponen de manifiesto que mientras el diseño del software se basa cada vez más en modelos, la IR generalmente se aborda utilizando lenguaje natural, de manera que sólo es posible establecer conexiones débiles entre los requisitos y el diseño. Los autores consideran que un enfoque de IR dirigida por modelos puede servir para salvar este problema. En nuestra opinión, alineando los requisitos textuales con el resto de modelos del desarrollo se facilitará la definición de las relaciones de traza desde los requisitos al resto de artefactos del desarrollo.

Por su parte, Bragança y Machado [37] apuntan que los enfoques de DSDM tienden a centrarse en transformaciones que normalmente se aplican a los modelos de diseño e implementación. Sin embargo, el análisis y la especificación de requisitos son actividades cruciales del proceso de desarrollo de software, puesto que dirigen el diseño de la arquitectura del sistema, y como tales, deberían integrarse en los métodos dirigidos por modelos. En consecuencia, debemos contribuir en esta línea proporcionando un proceso de DSDM completo que comience desde los requisitos y llegue hasta el código aplicando transformaciones automáticas modelo a modelo.

Baundry *et al.* [30] estudian si las técnicas del DSDM ofrecen buenas soluciones para unificar las actividades de la IR. Según estos autores existen numerosas herramientas y métodos para abordar cada una de las actividades propias de la IR (elicitación, modelado, negociación, comunicación y validación de requisitos), pero fundamentalmente permanecen separadas de manera que es difícil capturar la consistencia global de un documento de requisitos extenso. Baundry *et al.* confían en las técnicas del DSDM para relacionar las actividades de la IR de la misma forma que se utilizan para relacionar las actividades del desarrollo del software. De esta forma, el DSDM proporciona una solución técnica para la integración de las actividades de IR en un marco común.

Goknil *et al.* [107] señalan también la conveniencia de representar los requisitos como modelos para mantener la continuidad en el DSDM tratando todos los artefactos como modelos. Actualmente, aunque existen lenguajes de modelado estándar para expresar arquitecturas, diseños o la implementación de sistemas, las descripciones de los requisitos se consideran artefactos textuales cuya estructura no se especifica explícitamente. Para conseguir modelar los requisitos, los autores señalan que los

desarrolladores necesitan utilizar un metamodelo de requisitos que incluya los conceptos comunes a las técnicas existentes de IR.

En el dominio de la automoción es donde se ha puesto de manifiesto con más énfasis la necesidad de modelar los requisitos. Según Broy [41] la IR es crucial en este dominio y uno de los grandes problemas es contar con una IR adecuada. Este autor considera que una de las grandes opciones para conseguir mejoras es aplicar un proceso dirigido por modelos continuo que incluya una cadena de modelos para las actividades clásicas de: modelado de requisitos, modelado del diseño, modelado de la implementación y modelado de casos de prueba. Además, en el dominio de la automoción a menudo se combinan disciplinas heterogéneas como son Ingeniería del Software, ingeniería mecánica, ingeniería hardware cada una con diferentes herramientas y técnicas, esta diversidad hace necesario el uso de requisitos textuales como medio de comunicación común entre los equipos [212]. La representación gráfica de los requisitos textuales mediante modelos de requisitos permite, según Mellegård y Staron [212]: 1) visualizar gráficamente las relaciones entre los requisitos, 2) generar especificaciones textuales a partir de los modelos, 3) evaluar el impacto de los cambios en los requisitos, identificando requisitos contradictorios o que faltan y 4) trazar los requisitos hacia el diseño.

1.1.3.2. Propuestas del OMG: MDA y SysML

MDA (*Model Driven Architecture*) [253] es el enfoque DSDM propuesto por el OMG (*Object Management Group*) basado en sus estándares (MOF, UML, XMI, OCL, entre otros). La propuesta establece un marco de meta-modelado dividido en 4 capas (ver Figura 8):

- Nivel M3: lenguaje de definición de metamodelos, meta-metamodelo. En este nivel se encuentra MOF (*Meta-Object Facility*).
- Nivel M2: Definición de metamodelos, lenguaje para describir un dominio de interés específico. En este nivel se encuentra el metamodelo de UML que detalla los conceptos que pueden aparecer en los modelos de clases, de casos de uso, etc.
- Nivel M1: Modelos construidos a partir de los conceptos del metamodelo. En este nivel se encontrarían los modelos de UML.
- Nivel M0: especificación de un sistema concreto.

En este contexto, para describir modelos de requisitos (nivel M1) que puedan alinearse con otros modelos de análisis o diseño, sería necesario definir un metamodelo de requisitos (nivel M2) —que en nuestra propuesta hemos denominado REMM (*Requirements Engineering Meta-Model*). El metamodelo de requisitos debe especificar los conceptos que se deben tener en cuenta en el proceso de IR y las relaciones entre ellos utilizando algún lenguaje de definición de metamodelos (meta-metamodelo) como MOF (nivel M3). Como se ve de forma esquemática en la Figura 8, en nuestra propuesta utilizamos el lenguaje de metamodelado EMOF (un subconjunto del lenguaje MOF denominado *Essential MOF*) para la definición del metamodelo de requisitos (REMM) que nos permitirá definir modelos de requisitos que podamos trazar hacia otros modelos del desarrollo (REMM será presentado en detalle en el Capítulo 5). Una vez definido el metamodelo de requisitos, siguiendo el enfoque DSDM, se podrían definir transformaciones entre este metamodelo y otros, como por ejemplo el DSL para un

dominio particular como los sistemas domóticos. Este será el caso de estudio que presentaremos en el Capítulo 8.

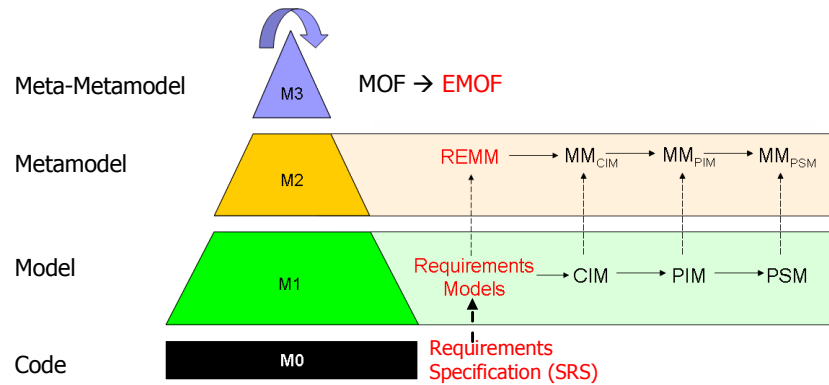


Figura 8 La propuesta en el marco de MDA.

El enfoque de desarrollo de software MDA se basa en la separación de la especificación de la funcionalidad del sistema, de la especificación de la implementación de esta funcionalidad en una plataforma concreta. Así, el desarrollo de software en MDA comienza con la definición del modelo del dominio (CIM, *Computation Independent Model*) en el que no se muestra ningún detalle sobre la estructura del sistema sino el vocabulario del dominio que se está modelando. La especificación de la funcionalidad y el comportamiento del sistema comienza con la definición un modelo independiente de la plataforma (PIM, *Platform-Independent Model*) que se construye utilizando un lenguaje de modelado (metamodelo) basado en MOF. La ventaja de los modelos PIM es que permanecen estables cuando la tecnología evoluciona. Mediante transformaciones de modelos, a partir de los modelos PIM se pueden obtener los modelos específicos de la plataforma (PSM, *Platform-Specific Model*) y a partir de estos, el código de la aplicación.

En el contexto de MDA, los requisitos se capturan en el modelo CIM. A este nivel, los requisitos se suelen especificar mediante modelos de proceso de negocio utilizando alguna técnica como BPMN (*Business Process Modeling Notation*) [258] o UML (diagramas de casos de uso o de actividades), de manera que, la IR tradicional basada en especificaciones de requisitos textuales se deja de lado en estos enfoques MDA. Cabe destacar que el modelo CIM es el modelo al que la comunidad investigadora de MDA ha prestado menos atención y existen pocas propuestas para la definición de CIMs [87]. Tampoco se han dedicado muchos esfuerzos a las transformaciones a partir de modelos CIM que es donde la IR hace hincapié [192], de hecho, las transformaciones desde los requisitos a los modelos de análisis no forman parte del ciclo de vida MDA [368]. Es más, Loniewski *et al.* [193] afirman que no existe un proceso de desarrollo sistematizado que aplique las técnicas de IR en el contexto del DSDM. Esta tesis doctoral supone una contribución a este nivel puesto que la propuesta de modelado de requisitos se encuentra en el nivel más alto de abstracción, constituyendo un primer paso antes de la construcción de los modelos CIM de la aplicación. La construcción de los modelos de requisitos servirá como guía para la especificación de los modelos CIM, ayudando a determinar el vocabulario del dominio de la aplicación (ver Figura 8).

Recientemente, el OMG ha propuesto el lenguaje estándar SysML [255] como lenguaje de modelado específico del dominio para aplicaciones de ingeniería de sistemas (*systems engineering*). SysML soporta la especificación, análisis, diseño, verificación y validación de un amplio rango de sistemas complejos. Estos sistemas podrían incluir hardware, software, información, procesos, personal e instalaciones (*facilities*). Una de las novedades de SysML frente a lenguajes de modelado como UML, es la inclusión del modelo de requisitos que permite integrar los requisitos textuales con otros artefactos del desarrollo. Se especifican además distintas relaciones entre los requisitos: derivación, jerarquía, satisfacción, verificación y refinamiento. SysML se ha utilizado en el dominio de la automoción [16] y los sistemas embebidos [69] que han visto en los modelos de requisitos que proporciona SysML la oportunidad de trazar las especificaciones de requisitos textuales con el resto de artefactos del desarrollo. Sin embargo, desde el punto de vista de la IR es cuestionable si el modelo de requisitos de SysML incluye todas las características fundamentales para la IR [325], en particular Herzog y Pandikow [120] cuestionan la completitud del conjunto de relaciones de traza inter-requisito. Jin *et al.* [145] proponen una extensión a los constructores que proporciona SysML porque consideran que no son suficientes para la especificación de una aplicación industrial real. Nosotros coincidimos con estas apreciaciones y, además, pensamos que para fomentar la reutilización de requisitos, los modelos de requisitos deben dar soporte a algún mecanismo de variabilidad más allá de la relación de traza “copy” que proporciona SysML. La relación de traza “copy” permite incluir una copia de solo lectura de un requisito (y, recursivamente, todos sus requisitos derivados) que ha sido especificado en un paquete diferente. La propuesta de esta tesis doctoral estaría en la línea del lenguaje SysML aprendiendo de sus bondades y tratando de solucionar sus carencias.

En resumen, la presente tesis doctoral trata de contribuir a la IR dirigida por modelos aplicando las técnicas propias del DSDM: metamodelado y transformaciones de modelos. A continuación destacamos los retos en cada una de estas líneas y los beneficios esperados de esta integración.

1.1.3.3. Metamodelo de requisitos

A pesar de la cantidad de información existente relacionada con requisitos (artículos, libros, estándares, etc.) no existe un modelo de referencia para IR comúnmente aceptado. De acuerdo con Weber and Weisbrod [357] los diferentes tipos de requisitos pueden llegar a ser confusos y por tanto, hay una necesidad urgente de definir un metamodelo de requisitos que defina formalmente los *conceptos* y *relaciones de traza* implicados en el proceso de IR. Según Marschall y Schoenmakers [204] mientras la mayoría de los modelos conceptuales existentes son bastante claros para las últimas etapas del desarrollo del software (por ejemplo el metamodelo de UML para el diseño de sistemas) las primeras etapas de IR todavía carecen de un metamodelo apropiado que permita estructurar y clasificar la información que se produce en esta fase.

La ausencia de dicho modelo de referencia también se pone de manifiesto por la iniciativa del Grupo de Trabajo de Requisitos (RWG, *Requirements Working Group* [134]) del INCOSE (*Internacional COuncil on Systems Engineering*) para recopilar

información sobre buenas prácticas en IR de la comunidad de Ingeniería de Sistemas. Entre los objetivos del RWG se encuentra capturar y hacer evolucionar el Libro de conocimiento sobre IR (REBOK, *Requirements Book of Knowledge*): teoría, casos de estudio, aplicación de evidencias, etc. La definición de REBOK se está llevando a cabo en el marco del proyecto REGAL (*Requirements Guide for All*) [75]. Si bien, de acuerdo con el sitio oficial del RWG uno de los resultados alcanzados debía ser la compilación de buenas prácticas sobre requisitos que fuese accesible para la comunidad de IR, en el momento de la escritura de esta memoria la información sobre el proyecto sólo es accesible para los miembros del INCOSE y no ha habido nueva difusión desde el año 2006.

Para la IR, siguiendo el estándar IEEE 830 [131], la trazabilidad se refiere a la claridad para determinar el origen de cada requisito y a la facilidad de referenciar cada requisito en el desarrollo futuro o en la documentación mejorada. Una de las definiciones de trazabilidad más referenciada es la de Gotel y Finkelstein [111]: “*la trazabilidad de requisitos se refiere a la capacidad de describir y seguir la vida de un requisito tanto hacia delante como hacia atrás (esto es, desde sus orígenes hacia su desarrollo y especificación, a su posterior despliegue y uso, y a través de periodos de refinamiento e iteración de estas fases)*”.

La diversidad de relaciones de traza queda reflejada en la taxonomía de modos de traza propuesta por Pinheiro [269]. Pinheiro clasifica las relaciones de traza atendiendo a la evolución de los requisitos, los tipos de elementos implicados en la traza y la dirección de la misma. En la Figura 9 se puede ver como estos modos no son disjuntos sino que existen solapamientos. Cada uno de estos modos se puede resumir como sigue:

- Evolución de los requisitos: un requisito se podría trazar hacia elementos que ocurren antes de su inclusión en la especificación de requisitos (RS, *Requirements Specification*), *trazabilidad Pre-RS*, o después de su inclusión, *trazabilidad Post-RS*. Luego, el metamodelo de requisitos debe incluir todos estos elementos.
- Tipos de elementos implicados en la traza: un requisito se podría trazar hacia otro requisito (*trazabilidad entre-requisitos*) o hacia otros “artefactos” (*trazabilidad extra-requisitos*). Por tanto, el metamodelo de requisitos debe especificar todos los tipos de elementos implicados en la fase de IR relacionados con la especificación de requisitos (stakeholders, fuentes, etc.).
- Dirección de la traza: un requisito se puede trazar *hacia delante*, esto es, hacia artefactos del desarrollo creados en fases posteriores a la IR como elementos de diseño o implementación, o *hacia atrás*, hacia su fuente. En este último caso podría coincidir con la relación de traza *pre-RS*, lo que pone de manifiesto lo que comentábamos al principio de que los modos no eran disjuntos.

El reto más importante es trazar los requisitos hacia delante. Según Mellegård y Staron [212] a pesar de las numerosas ventajas de las técnicas de modelado existentes que se pueden utilizar para modelar los requisitos (como UML o SysML) los ingenieros todavía luchan para enlazar de manera eficiente los requisitos con los diseños. En el marco del DSDM podemos seguir dos estrategias diferentes para trazar los requisitos hacia otros artefactos del desarrollo: trazabilidad explícita e implícita. En el primer caso se refiere a la inclusión explícita de este tipo de traza en el metamodelo de requisitos. En el segundo caso, se proponen transformaciones de modelos desde los modelos de

requisitos a otros modelos del desarrollo de manera que la relación de traza queda implícita en el proceso de transformación. En el apartado siguiente se analiza los problemas de las transformaciones de modelos a partir de los requisitos y la presentación de las diferentes propuestas en esta línea se posponen hasta el Capítulo 3.

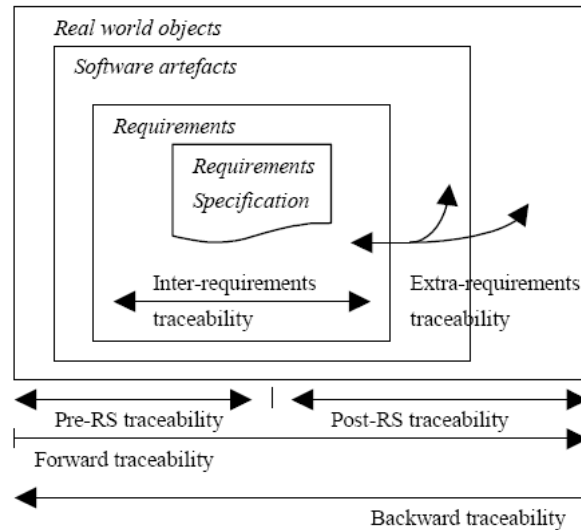


Figura 9 Modos de trazabilidad [figura tomada de [269]].

Las relaciones de traza constituyen uno de los aspectos claves de la IR ya que, por un lado, la gestión de la trazabilidad de requisitos puede contribuir a reducir los errores en la especificación de requisitos [184] y, por otro lado, uno de los beneficios potenciales de la implementación de la trazabilidad de requisitos es la reutilización [360]: “La trazabilidad facilita la reutilización de componentes del producto mediante la identificación de paquetes de requisitos, diseños, código y casos de prueba relacionados”. Haciendo explícitas las relaciones entre los elementos implicados en la especificación de requisitos damos soporte a la definición de procesos de reutilización sistemáticos. La ventaja de especificar en un modelo de requisitos todos los elementos que están relacionados con un requisito es que permitirá que al reutilizar este requisito se pueda “tirar de las trazas” (en sentido figurado) y reutilizar al mismo tiempo, todos los elementos que estén relacionados con él: requisitos de los que depende, requisitos que lo refinan, experto que lo propuso, casos de prueba, fuente del requisito, etc.

La importancia de la trazabilidad también se desprende del informe “*Grandes Retos de la Trazabilidad*” [24] publicado por el Centro de Excelencia para la Trazabilidad (*Center of Excellence for Traceability*) en el que destaca como una de las áreas de investigación claves la “*definición de un meta-modelo para representar la información semántica de las enlaces de trazabilidad y proporcionar ejemplos de instanciación en dominios específicos*”. Además, la necesidad del metamodelo de trazabilidad también se deriva de las exigencias de los estándares como el IEEE 830, CMMI e ISO 9001, en los que la trazabilidad constituye uno de los elementos esenciales.

A pesar de la importancia de la trazabilidad de requisitos, no existe un consenso sobre los tipos de relaciones de traza, y la semántica de dichas relaciones, que se deben

tener en cuenta en el proceso de IR. Existen diversas propuestas, unas centradas en las relaciones entre requisitos [48, 66] y otras más generales [58, 189, 281] que incluyen no sólo relaciones entre requisitos sino también relaciones de traza hacia elementos del diseño. Para la definición de un nuevo metamodelo de requisitos será necesario evaluar cada una de las propuestas existentes, analizando las similitudes y destacando las carencias de cada uno de los enfoques. Este estudio se presenta en el capítulo 3.

Técnicas de meta-modelado

Una vez definido en un metamodelo los conceptos propios de un dominio se debe elegir una técnica para dar soporte a dicho metamodelo, de manera que se puedan construir modelos concretos utilizando los elementos del metamodelo. En el contexto de UML, son dos las técnicas de meta-modelado a tener en cuenta: (1) definir un Perfil UML o (2) definir un nuevo metamodelo utilizando el meta-metamodelo (lenguaje para definir metamodelos) MOF.

En [239] se afirma que no es obvia la elección entre una u otra técnica, y en este mismo trabajo se recomienda crear un nuevo metamodelo cuando la distancia semántica entre los elementos de modelado UML existentes y los nuevos elementos que se van a modelar sea muy grande. Coincidimos con Robertson y Robertson [290] en que “la mayoría de los modelos UML no son apropiados para el trabajo con requisitos. Son buenos modelos de diseño pero pésimos modelos de requisitos”. En consecuencia, parece que la técnica más conveniente para la definición del metamodelo de requisitos sería la definición de un nuevo metamodelo utilizando MOF.

Una de las implementaciones más extendidas de MOF (en realidad un subconjunto de MOF llamado EMOF, *Essential MOF*) la proporciona un plug-in de Eclipse denominado EMF (*Eclipse Modelling Framework*) [43], que permite a los diseñadores crear, manipular y almacenar tanto los metamodelos, como los modelos construidos utilizando los conceptos del metamodelo.

Además de EMF, existen otras iniciativas relacionadas con DSDM desarrolladas en el marco de Eclipse como GMF (*Graphical Modelling Framework*, <http://www.eclipse.org/gmf>), que permite a los diseñadores: (1) crear una representación gráfica para cada concepto del dominio especificado en el metamodelo, (2) definir una paleta de herramientas para crear y añadir estos conceptos a los modelos, y (3) definir una correspondencia entre cada elemento del metamodelo, su representación gráfica y la herramienta para incluirlo en los modelos.

El uso extendido de Eclipse en la comunidad de DSDM, junto con el hecho de que es un entorno gratuito, abierto e independiente de la plataforma, hace que sea el entorno candidato para el desarrollo de una herramienta que pueda dar soporte al metamodelo de requisitos. Dicha herramienta debe permitir, fundamentalmente, la especificación de modelos de requisitos, validando su corrección con respecto al metamodelo. Una especificación más detallada de la tecnología utilizada se pospone hasta el capítulo 7.

1.1.3.4. Transformaciones a partir de los modelos de requisitos

Según Almeida y Rodríguez da Silva [20] hasta donde ellos saben, no existe ningún enfoque de IR que integre los paradigmas de IR y DSDM, esto es, no existe ninguna propuesta que comience con el proceso de desarrollo de requisitos utilizando

especificaciones en lenguaje natural y termine con la aplicación de técnicas generativas para producir artefactos tales como documentación y código fuente.

Mientras podemos encontrar en la literatura diferentes propuestas de desarrollo de software que incluyen un metamodelo de requisitos (estas propuestas se presentan en el capítulo 3), las propuestas relacionadas con las transformaciones de requisitos son escasas. Probablemente, los motivos de esta escasez tienen que ver con la dificultad de transformar requisitos textuales, aunque estén expresados en un modelo, a un modelo diferente. Según Kaindl y Falb [152] en general, el diseño de la arquitectura no se puede transformar completamente a partir de los requisitos puesto que esto implicaría que la transformación consistiría en un solucionador de problemas “mágico”. No es una coincidencia que la mayoría de los enfoques que proponen estrategias de transformación en el contexto de MDA toman los modelos PIM como modelos iniciales saltándose la definición de los modelos CIM, que es donde se incluyen las especificaciones de requisitos [73]. Según Kleppe *et al.* [170] no es posible derivar de forma automática modelos PIM a partir de los modelos CIM. De hecho, Yue *et al.* [368] afirman que las transformaciones desde los requisitos a los modelos de análisis no forman parte del ciclo de vida de MDA, sin embargo, estos autores señalan que si se pudieran automatizar estas transformaciones, se podría conseguir al mismo tiempo la trazabilidad desde los requisitos hasta el código.

Un estudio reciente sobre las propuestas para la transformación de los requisitos del usuario en modelos de análisis [368] revela que un enfoque adecuado debería imponer restricciones razonables al lenguaje natural para poder definir las transformaciones de modelos. Algunas propuestas [20, 156, 271] utilizan un lenguaje natural controlado para representar los requisitos textuales de manera que se reduce la ambigüedad y la complejidad de las especificaciones de requisitos. Restringiendo la forma en la que se escriben los requisitos es posible automatizar la generación de diferentes modelos de desarrollo a partir de las especificaciones de requisitos. Además, el proceso puede mejorarse si el lenguaje de especificación de requisitos que se utilice incluye conocimiento del dominio. Si el dominio se ha definido mediante un DSL, la especificación de requisitos podría trazarse a los conceptos del DSL puesto que los requisitos y los conceptos de análisis están definidos al mismo nivel de abstracción. La definición de lenguajes de requisitos específicos de dominio ha sido destacada como uno de los retos para mejorar el procesamiento automático de especificaciones de requisitos [42]. Un estudio detallado de las propuestas de transformaciones de modelos de requisitos será presentado en el capítulo 3.

Además de las transformaciones modelo a modelo, el enfoque de DSDM ofrece la posibilidad de definir transformaciones modelo a texto. Este tipo de transformaciones son utilizadas ampliamente para la generación del código de la aplicación a partir de los modelos. En el caso de la IR, hemos destacado en la sección 1.2 la importancia del documento de requisitos como medio de comunicación entre los stakeholders. El documento de requisitos asegura que todos tienen una idea común sobre el sistema que se va a desarrollar. En este contexto, se pueden utilizar las transformaciones modelo a texto para generar la especificación textual de los requisitos de acuerdo al formato establecido en la organización, por ejemplo el IEEE-830 [131] o bien para generar la documentación de requisitos en un formato estándar como ReqIF (*Requirements*

Interchange Format) [260] que permita la exportación de los requisitos a otras herramientas de gestión de requisitos.

11.3.5. Beneficios esperados de la aplicación de las técnicas del DSDM a la IR

La integración de los principios del DSDM en la IR puede beneficiar a ambos dominios. Por un lado, la definición de un metamodelo de requisitos: 1) permite integrar en el mismo modelo todos los conceptos de la IR haciendo explícitos los elementos que forman parte del proceso de IR y cómo se relacionan; 2) se puede utilizar como modelo conceptual común para las herramientas de gestión de requisitos que soporten el proceso de IR, el modelo conceptual soportado por la mayoría de las herramientas de gestión de requisitos es simple puesto que normalmente incluyen el concepto de requisito y únicamente las relaciones “sub-requisito” y “requisitos asociados” [304], y 3) permite validar la conformidad de los modelos de requisitos con el metamodelo. Por otro lado, el metamodelo de requisitos podría relacionarse con otros metamodelos que describan los artefactos del análisis o el diseño. De esta forma, se alinean los requisitos con los modelos y, en consecuencia, los requisitos se pueden integrar en un enfoque de DSDM.

La aplicación de las técnicas del DSDM a la IR también puede contribuir a conseguir los objetivos de reutilización sistemática y trazabilidad de requisitos, y por tanto, contribuirá a mejorar la productividad y la calidad de los productos desarrollados. En cuanto a la reutilización sistemática de requisitos, sea cual sea el metamodelo de requisitos que se adopte:

- Permite modelar explícitamente tanto las trazas entre-requisitos como las trazas existentes entre los requisitos y el resto de elementos implicados en el proceso de IR.
- Proporciona uno o más mecanismos explícitos de reutilización que permite: (1) definir modelos de requisitos reutilizables (modelado *para reuso*), y (2) construir una especificación de un producto específico reutilizando alguno de los requisitos definidos previamente (modelado *con reuso*). En este último caso, cuando se reutiliza un requisito en una especificación de un producto concreto, todos los elementos relacionados con él pueden incorporarse fácilmente a la especificación. Además, la definición de trazas entre requisitos (en particular, dependencias de incompatibilidad entre requisitos) permite identificar y prevenir muchos problemas en el momento de la reutilización.

En cuanto a la trazabilidad de requisitos, como ya se ha señalado, el metamodelo de requisitos debe recoger explícitamente todas las relaciones que existen entre los artefactos implicados en el proceso de IR. La trazabilidad hacia delante, esto es desde los requisitos hacia otros artefactos del desarrollo, se puede abordar de dos formas en el contexto del DSDM: explícitamente en el metamodelo o implícitamente mediante transformaciones de modelos. Desde nuestro punto de vista, el enfoque más beneficioso es una combinación de ambas estrategias, por ejemplo, la generación automática de los modelos de análisis a partir de los modelos de requisitos aplicando reglas de transformación y, al mismo tiempo, enlazar explícitamente los requisitos con los elementos del modelo de análisis generados. El desarrollo de esta hipótesis constituye

una contribución de esta tesis doctoral puesto que según [368] la integración de transformaciones y trazabilidad todavía no se ha resuelto.

Los beneficios de la integración de los enfoques de IR y DSDM también son señalados por Silva *et al.* [321] quienes destacan: 1) la definición de especificaciones de requisitos rigurosas y formales, 2) trazabilidad entre artefactos, 3) producción automática de artefactos software y prototipos y 4) mejora de la productividad.

Coincidimos con Zikra *et al.* [374] en que la captura de las relaciones entre los requisitos y los modelos del DSDM puede: 1) mejorar la comprensión de las decisiones de diseño; 2) facilitar la propagación desde los requisitos a los modelos; 3) mejorar la corrección y completitud de los modelos y del software final; e 4) incrementar el potencial de reutilización de los modelos que proceden de requisitos similares.

1.2. Hipótesis y objetivos

La *hipótesis de partida* de esta tesis doctoral es que la IR se puede beneficiar al aplicar las técnicas del DSDM, puesto que la formalización en un metamodelo de los conceptos y relaciones involucradas en esta fase favorecerá la reutilización sistemática de requisitos y la trazabilidad, y por su parte el enfoque DSDM se beneficiará de la definición del metamodelo de requisitos puesto que permitirá integrar los requisitos en el proceso de desarrollo de forma compatible con su filosofía basada en modelos.

En consecuencia, el *objetivo principal* que se define para probar la hipótesis anterior es: aplicar las técnicas del DSDM (metamodelado y transformaciones de modelos) para dar soporte a las actividades de IR, favoreciendo la reutilización y trazabilidad de requisitos, de manera que los requisitos se integren en un proceso completo de DSDM (desde los requisitos hasta el código).

Este objetivo principal se desglosa en los siguientes subobjetivos:

- Objetivo 1.** Definir un método práctico de IR basado en reutilización, en el que integrar las técnicas del DSDM.
- Objetivo 2.** Estudiar las propuestas de integración de las técnicas de DSDM en IR, en concreto analizaremos los metamodelos de requisitos existentes en la literatura y las transformaciones desde modelos de requisitos hacia otros modelos del desarrollo.
- Objetivo 3.** Definir un metamodelo de requisitos que incluya los conceptos fundamentales para dar soporte a las actividades de IR y que dé soporte a la reutilización sistemática de requisitos.
- Objetivo 4.** Definir transformaciones modelo a texto, para generar la documentación de requisitos, y transformaciones modelo a modelo, para integrar los modelos de requisitos en un proceso de DSDM completo (desde los requisitos al código).
- Objetivo 5.** Diseñar e implementar el prototipo de una herramienta de soporte a los modelos definidos en el Objetivo 3 y las transformaciones definidas en el Objetivo 4. La herramienta debe permitir: dibujar modelos de requisitos tanto en la fase *para* reutilizar como en la fase *con* reutilización, reutilizar modelos previos, validar la corrección de los modelos, generar documentos de especificación de

- requisitos a partir de los modelos y transformar los modelos de requisitos en modelos de análisis conformes a un DSL.
- Objetivo 6.** Validar los resultados anteriores en el marco de un caso de estudio real.

1.3. Marco de la tesis

La realización de la presente tesis doctoral se ha realizado en el seno de un grupo de investigación y como resultado de la experiencia investigadora de la doctoranda a lo largo de varios proyectos de investigación, contratos con empresas y su participación en redes temáticas relacionadas con el tema de la tesis. En las siguientes secciones se detalla cada una de las contribuciones.

1.3.1. Grupo de investigación

La doctoranda pertenece al *Grupo de Investigación de Ingeniería del Software* (GIIS, E097-01, <http://www.um.es/giisw/ES/>) de la Universidad de Murcia oficialmente desde el año 1998, aunque se establecieron los primeros contactos con el grupo durante la realización del Proyecto Final de Carrera desarrollado en el marco del proyecto MENHIR-OM (sección 3.2.1), cuyos resultados fueron presentados en las II Jornadas de Ingeniería del Software [229]. Desde su incorporación al GIIS, sus intereses y líneas de investigación están centrados en las primeras etapas del modelado de los Sistemas de Información, abordando dicha etapa, en un primer momento, mediante la combinación de prototipos (desechable y evolutivo) [231] o modelando el negocio (*Business Modeling*) como punto de partida para obtener los primeros modelos de requisitos [99]. En el año 2000, en el marco del proyecto DOLMEN-SIREN (sección 3.2.2), es cuando comienza la línea de investigación relacionada con la reutilización de requisitos que luego se concreta en el modelado de requisitos durante el proyecto META-DÉDALO (sección 3.2.5).

1.3.2. Proyectos de investigación

Desde su incorporación al grupo de investigación, la doctoranda ha participado en cinco proyectos de investigación coordinados con otras universidades españolas, solicitados en el marco de los distintos Planes Nacionales de I+D+i, así como en un proyecto PROFIT (*PROgrama de Fomento de Investigación Técnica*) para la transferencia tecnológica a las empresas. A continuación se presenta los detalles de cada uno de los proyectos así como un breve resumen de los objetivos propuestos en cada subproyecto, los resultados obtenidos más directamente relacionados con la línea de investigación de la doctoranda y que, en consecuencia, constituyen los antecedentes de la presente tesis doctoral.

1.3.2.1 Proyecto MENHIR-OM

El proyecto OM (*OASIS Method*) se encuadra dentro del proyecto coordinado MENHIR (Tabla 1). El objetivo del proyecto OM era la definición de un modelo de proceso de desarrollo de software orientado a objetos (OO), fiable (proporcionando técnicas de verificación/validación formales), compatible con los principales estándares industriales (notación UML, gestión de requisitos, etc.) y orientado a sistemas con uso intensivo de datos. Aunque la idea inicial era utilizar el lenguaje de especificación OO OASIS, de ahí el nombre del proyecto, finalmente se cambió por el lenguaje UML (*Unified Modelling Language*) puesto que fue adoptado por el OMG (*Object Management Group*) como lenguaje estándar de modelado OO.

El modelo de proceso para el desarrollo de sistemas de información de gestión propuesto como resultado del proyecto OM, surge de la combinación de la metodología IDEA y del método OOram, unificando sus notaciones mediante el uso de UML [97, 100]. El aspecto clave de la propuesta es la estrategia para obtener de modo sistemático el modelo de casos de uso y el modelo conceptual (modelo de requisitos), a partir del modelado del negocio basado en diagramas de actividades UML [95, 96, 98, 99].

Título del proyecto coordinado	MENHIR (Modelos, Entornos y Nuevas Herramientas para la Ingeniería de Requisitos)
Título del subproyecto	OM (Oasis Method: Constructing Trusted Information Systems from OO Specifications)
Referencia	TIC97-0593-C05-02
Entidad financiadora	CICyT (Comisión Interministerial de Ciencia y Tecnología)
Coordinador del proyecto	Dr. D. Isidro Ramos Salavert
Entidades participantes	Politécnica de Valencia, Murcia, Sevilla, Granada y Valladolid
Investigador Principal del subproyecto	Dr. D. Ambrosio Toval Álvarez
Duración	1997- 2000
Importe total del proyecto:	9.555.000 ptas

Tabla 1 Ficha del proyecto MENHIR

Por otro lado, se estudió otra estrategia para la especificación de requisitos del sistema basada en especificaciones formales a partir de las cuales se pueda generar automáticamente un prototipo que permita la validación de las mismas [229, 230]. Finalmente, se definió una estrategia de desarrollo de software OO basada en la combinación de prototipos desechable y evolutivo, a partir de la especificación formal de los requisitos utilizando el lenguaje OO, formal y declarativo OASIS. A partir de las especificaciones formales escritas en OASIS, se genera automáticamente un prototipo desechable que sirve para validar y refinar los requisitos de usuario. Una vez validada la especificación, se genera el primer prototipo evolutivo (que evolucionará hasta el sistema final), que comprende un conjunto de clases (implementadas en un lenguaje de programación como Eiffel) dotadas de la semántica de la especificación formal mediante la definición de asertos [231].

1.3.2.2. Proyecto DOLMEN-SIREN_{rm}

El proyecto SIREN_{rm} (*SIREN_{rm}: Simple REuse of software requiremeNts and rigorous modelling*) se enmarca en el proyecto coordinado DOLMEN (Tabla 2). El objetivo del proyecto SIREN_{rm} era la definición de un método ágil de IR basado en reutilización y en los principales estándares de Ingeniería del Software.

Título del proyecto coordinado	DOLMEN (<i>Distributed Objects, Languages, Models and Environments</i>)
Título del subproyecto	SIREN _{rm} (<i>SIREN_{rm}: Simple REuse of software requiremeNts and rigorous modelling</i>)
Referencia	TIC2000-1673-C06-02
Entidad financiadora	Ministerio de Ciencia y Tecnología. Plan Nacional de I+D+I/FEDER
Coordinador del proyecto	Dr. D. Isidro Ramos Salavert
Entidades participantes	Politécnica de Valencia, Murcia, Sevilla, Granada, Valladolid y Castilla-La Mancha
Investigador Principal del subproyecto	Dr. D. Ambrosio Toval Álvarez
Duración	2001- 2003
Importe total del proyecto	14.840.000 ptas

Tabla 2 Ficha del proyecto DOLMEN

El método SIREN [341, 343] incluye un modelo de proceso en espiral, plantillas de documentos de requisitos, un repositorio de requisitos reutilizables organizado en catálogos y una herramienta CARE (*Computer-Aided Requirements Engineering*) de soporte denominada SirenTool [180].

Durante la realización del proyecto no sólo se trabajó en la definición del modelo de proceso de SIREN sino también en la definición de catálogos de requisitos reutilizables de utilidad para el desarrollo de sistemas. Así, se definieron los catálogos para el perfil de seguridad [340, 343] y para el perfil de la ley de protección de datos de carácter personal [344].

1.3.2.3. Proyecto PROFIT para el desarrollo de aplicaciones web

Los objetivos del proyecto PROFIT que se detalla en la Tabla 3 eran: (1) definir un proceso software para la empresa Sinergia Tecnológica (ST) destinado a proyectos de desarrollo de aplicaciones web que implican el uso de la tecnología asociada al lenguaje orientado a objetos Java y (2) proporcionar una herramienta de soporte a dicho proceso.

Además, se estudió la viabilidad de integrar las técnicas de XP (*eXtreme Programming*) en el proceso de desarrollo web definido puesto que pensamos que podría ser adecuado para una empresa como ST en el que los equipos de desarrollo no estaban formados por más de 10 ó 12 personas, y la naturaleza de los proyectos requería un desarrollo rápido, de dos a tres meses a lo sumo.

Título del proyecto coordinado	Definición y Aplicación de un proceso software basado en UML para el desarrollo de aplicaciones web
Referencia	FIT-070000-2000-411
Entidad financiadora	Ministerio de Ciencia y Tecnología
Entidades participantes	Universidad de Murcia y la empresa Sinergia Tecnológica (ST)
Investigador Principal	Dr. D. Jesús García Molina
Duración	2001-2002

Tabla 3 Ficha del proyecto PROFIT

1.3.2.4. Proyecto DYNAMICA-PRESSURE

El proyecto PRESSURE (*PREcise Software modelS and reqUirements ReusE*) se encuadra dentro del proyecto coordinado DYNAMICA (Tabla 4).

Título del proyecto coordinado	DYNAMICA (<i>DYNamic and Aspect-Oriented Modeling for Integrated Component-based Architectures</i>)
Título del subproyecto	PRESSURE (<i>PREcise Software modelS and reqUirements ReusE</i>)
Referencia	TIC2003-07804-C05-05
Entidad financiadora	Ministerio de Ciencia y Tecnología. Plan Nacional de I+D+I/FEDER
Coordinador del proyecto	Dr. D. Isidro Ramos Salavert
Entidades participantes	Politécnica de Valencia, Politécnica de Cartagena, Castilla La Mancha, Carlos III y Murcia
Investigador Principal del subproyecto	Dr. D. Ambrosio Toval Álvarez
Duración	2004-2006
Importe total del proyecto	163.280€ (subproyecto PRESSURE)

Tabla 4 Ficha del proyecto DYNAMICA.

Este proyecto es la extensión natural de las investigaciones y resultados obtenidos en el proyecto coordinado previo (ver SIREN_{rm} en la sección 1.3.2.2) para consolidar los catálogos definidos (seguridad [116, 179] y protección de datos de carácter personal [207]) y para definir nuevos catálogos, en concreto en el dominio de los sistemas teleoperados (STO).

El desarrollo del catálogo para los STO nos hizo ver las limitaciones del modelo de proceso de SIREN para trabajar con dominios poco estructurados y la conveniencia de extenderlo para guiar en la definición de requisitos para este nuevo tipo de dominios. Este nuevo enfoque parte de la especificación de un modelo de características que pueda trazarse hacia los requisitos del sistema [179, 246-248] y cuyos resultados culminaron en la tesis doctoral de Joaquín Nicolás [245].

1.3.2.5. Proyecto META-DEDALO

Los trabajos relacionados con el modelado de requisitos se inician en el marco del proyecto DEDALO (*Desarrollo de sistEmas De calidad bAsado en modeLos y requisitOs*) dentro del proyecto coordinado META (Tabla 5). El proyecto coordinado giraba en torno al DSDM por lo que se estudia la conveniencia de aplicar las técnicas que ofrece este enfoque en las actividades de IR. En consecuencia se establece la hipótesis de partida de esta tesis doctoral [233] y se obtienen los primeros resultados en la definición del metamodelo de requisitos [348], la definición del modelo de variabilidad para dar soporte a la reutilización sistemática de requisitos [234] y un primer prototipo de la herramienta de soporte a los modelos [236, 348].

De forma paralela a los trabajos de la doctoranda, y dentro de esta misma línea de investigación sobre metamodelos de requisitos, se trabajó en la integración en un metamodelo de requisitos siguiendo el enfoque GORE con un metamodelo de medición (que permitan evaluar los requisitos) y de usabilidad en el contexto del desarrollo de sistemas de información web. Estos trabajos culminaron en la tesis doctoral de Fernando Molina [222]. Este trabajo podría considerarse un complemento al metamodelo propuesto en la presente tesis doctoral, puesto que aborda de forma específica cuestiones relacionadas con el modelado de la calidad en las primeras etapas del desarrollo del software que quedan fuera de nuestra propuesta.

Título del proyecto coordinado	META (Modelos, Entornos, Transformaciones y Aplicaciones)
Título del subproyecto	DEDALO (Desarrollo de sistEmas De calidad bAsado en modeLos y requisitOs)
Referencia	TIN2006-15175-C05-03
Entidad financiadora	Ministerio de Ciencia y Tecnología. Plan Nacional de I+D+I/FEDER
Coordinador del proyecto	Dr. D. Isidro Ramos Salavert
Entidades participantes	Universidad Politécnica de Valencia, la Universidad Politécnica de Cartagena, la Universidad de Murcia, el Instituto Europeo del Software y la Universidad de Castilla La Mancha.
Investigador Principal del subproyecto	Dr. D. Ambrosio Toval Álvarez
Duración	2007-2009
Importe total del proyecto	110.000€ (subproyecto DEDALO)

Tabla 5 Ficha del proyecto META

1.3.2.6. Proyecto PEGASO-PANGAEA

El proyecto PANGAEA (*Process for globAl requiremeNts enGinEering and quAlity*) es un subproyecto del proyecto coordinado PEGASO (*Process for globAl requiremeNts enGinEering and quAlity*) (Tabla 6). El objetivo principal de PANGAEA consiste en la definición de un método homónimo de IR que pueda ser usado en el contexto del desarrollo global de software. En relación al modelado de requisitos se trata de definir

un modelo de referencia de requisitos para el desarrollo global de software, con vistas a establecer formalmente los elementos que forman parte de la especificación de requisitos y el modelo de trazabilidad entre requisitos y entre requisitos y el resto de artefactos software. En esta línea se ha trabajado en la trazabilidad de los modelos de requisitos hacia otros modelos del desarrollo mediante transformaciones de modelos, en concreto se han definido reglas de transformación desde los modelos de requisitos hacia modelos conformes con la definición de un DSL para un dominio concreto como puede ser el de los sistemas domóticos.

Título del proyecto coordinado	PEGASO (<i>Process for globAl requiremeNts enGinEering and quAlity</i>)
Título del subproyecto	PANGEA (<i>Process for globAl requiremeNts enGinEering and quAlity</i>)
Referencia	TIN2009-13718-C02-02
Entidad financiadora	Ministerio de Ciencia y Tecnología. Plan Nacional de I+D+I/FEDER
Coordinador del proyecto	Dr. D. Mario Piattini Velthuis
Entidades participantes	Universidad de Castilla La Mancha Universidad de Murcia y 3 EPOs
Investigador Principal del subproyecto	Dr. D. Ambrosio Toval Álvarez
Duración	2010-2012
Importe total del proyecto	286.407€ (subproyecto PANGEA)

Tabla 6 Ficha del proyecto PEGASO

1.3.3. Relación con empresas

La doctoranda ha establecido contacto con empresas en temas relacionados con la IR a través de su participación en un proyecto de transferencia tecnológica y un contrato que se detallan a continuación. En las actividades de transferencia tecnológica que siguen el contacto con clientes reales y la especificación de un sistema real refuerzan los conocimientos teóricos adquiridos a través de la literatura.

1.3.3.1. Proyecto de transferencia tecnológica: GARTIC

El proyecto GARTIC (*Gestión Automatizada de Requisitos basada en reutilización para PYMES del sector TIC*, Tabla 7) es un proyecto de innovación centrado en la transferencia tecnológica de un proceso general de IR y del método SIREN de reutilización de requisitos a cinco empresas de desarrollo de software (PYMES del sector TIC) en la Región de Murcia. El objetivo general de GARTIC era introducir en las empresas participantes un método de IR basado en reutilización (técnicas, proceso, guías y herramienta de soporte), que les ayude a mejorar los índices de calidad y productividad en el desarrollo de software. SIREN fue puesto en práctica en cinco proyectos piloto en el marco de GARTIC -uno por empresa- y redefinido de acuerdo con la retroalimentación recibida en todos ellos. GARTIC también sirvió para redefinir SIREN y para desarrollar una nueva versión de SirenTool.

Título	GARTIC: Gestión Automatizada de Requisitos basada en reutilización para PYMES del sector TIC
Entidad financiadora	Consejería de Industria y Medio Ambiente de la Comunidad Autónoma de la Región de Murcia
Entidades participantes	CentTIC (<i>Centro Tecnológico de las Tecnologías de la Información y las Comunicaciones de la Región de Murcia</i>), GIIS, Adalid Management y Outsourcing S.L., Bahía Information Technology S.A., Foro Digital S.L., Global Touch Express S.L., S.Q.A MURCIA S.L.
Investigador Principal	Dr. D. Ambrosio Toval Álvarez
Duración	Enero 2008- febrero 2009
Importe total del proyecto	32.480€ (GIIS)

Tabla 7 Ficha del proyecto GARTIC

1.3.3.2. Contratos con empresas

La experiencia en el campo de la IR también se ha visto enriquecida a través de un contrato con la empresa TECNOPRODUCCIONES MULTIMEDIA, S. L. (Molina de Segura, Murcia) titulado “*Ingeniería de Requisitos y construcción de Prototipo para proyecto de software multimedia*” en el año 2003. El trabajo realizado por la doctoranda en este proyecto incluía la definición del documento de requisitos para un proyecto de software multimedia que sirviera como base para la implementación de un primer prototipo del producto que la empresa estaba pensando lanzar al mercado.

1.3.4. Redes temáticas

El objetivo fundamental de las redes temáticas es facilitar la movilidad de los investigadores interesados en un tema común, entre los nodos participantes en la red, con el objeto de intercambiar conocimientos específicos sobre el tema que versa la red. La mayoría de las redes en la que ha participado la doctorando tienen que ver con la seguridad, puesto que es una de los perfiles de requisitos en los que el grupo de investigación ha estado trabajando desde la definición de SIREN, aunque también hay otras más generales relacionadas con la Ingeniería del Software o el proceso software, en particular con el DSDM. Las redes en las que ha participado la doctoranda son:

- **RETISBD:** Red Temática de Investigación en Seguridad en Bases de Datos (2000-2001). Acción Especial TIC 2001-1873-E.
- **RETIS:** Red Temática de Investigación en Ingeniería del Software (2000-2001). Acción Especial TIC 2000-2052-E.
- **RETISTIC:** Red Temática española de Investigación en el campo de la Seguridad de las Tecnologías de Información y las Comunicaciones (2004-2006). TIC2002-12487-E.
- **CALIPSO:** *Calidad del producto y proceso software* (2005-2007). TIN2005-24055-E.

- **RETISTRUST:** Red temática de investigación en el campo de la seguridad y confianza para los sistemas de información en una sociedad conectada (2006-2007). TIN2006-26885-E.
- **DSDM2:** *Desarrollo de Software Dirigido por Modelos* (2008-2010), Acción Complementaria TIN2008-00889-E/TIN del Ministerio de Educación y Ciencia.
- **TeDIS:** *Tecnologías para el Desarrollo Industrial de Software* (2012-2013). TIN2011-15009-E.

1.4. Organización de la tesis

El resto del contenido de la tesis doctoral se articula alrededor de nueve capítulos:

Capítulo 2. Estrategia de investigación. En este capítulo explicaremos los métodos de investigación utilizados: Revisión sistemática de la literatura para el estudio del estado del arte en las distintas áreas de interés que confluyen en esta tesis, el Método de investigación constructivo para el desarrollo de la propuesta integrada de DSDM a partir de los requisitos y la encuesta para la validación de la propuesta.

Capítulo 3. Análisis del estado del arte. En este capítulo se presenta el resultado de las revisiones sistemáticas derivadas del estudio de la integración de los requisitos en el DSDM: propuestas de metamodelos de requisitos y transformaciones de modelos a partir de los requisitos.

Capítulo 4. SIREN: un método práctico de Ingeniería de Requisitos basado en reutilización. En este capítulo se describe el método SIREN (modelo de proceso y guías) que será el método de IR en el que se aplique las técnicas del DSDM y los aspectos claves que hay que tener en cuenta en la reutilización de requisitos.

Capítulo 5. REMM: un metamodelo para Ingeniería de Requisitos. Este capítulo detalla el metamodelo de requisitos que constituye el eje central de la tesis: los conceptos y la semántica de las relaciones de traza que en él se especifican.

Capítulo 6. ReToCo: un proceso completo de DSDM, desde los requisitos hasta el código. En este capítulo se definen las transformaciones modelo-texto y modelo-modelo desde los modelos de requisitos que sirven como base para la definición de un proceso completo de DSDM (desde los requisitos al código) a partir de los requisitos basándonos en los principios del modelado específico de dominio (DSM).

Capítulo 7. REMM-Studio^{*}: un entorno integrado de IR dirigido por modelos y basado en reutilización. Este capítulo muestra la funcionalidad de la herramienta construida a partir del metamodelo de requisitos, REMM, cuyo objetivo no es sólo la especificación de modelos de requisitos sino dar soporte al proceso de IR basado en reutilización de modelos, validando la corrección de los modelos y generando automáticamente documentos de especificación de requisitos.

Capítulo 8. Caso de estudio: Sistemas domóticos. En este capítulo se concretan las transformaciones modelo a modelo y el proceso de DSDM en el dominio de los sistemas domóticos. Se define un catálogo reutilizable de requisitos domóticos que constituye un lenguaje restringido para la especificación de los requisitos de una aplicación de este dominio. Reutilizando los requisitos del catálogo se

pueden definir los requisitos de una nueva aplicación. A partir de la especificación de requisitos se puede generar automáticamente el modelo de la aplicación conforme al DSL definido para los sistemas domóticos. Siguiendo el enfoque DSM, a partir de los modelos conformes al DSL se puede generar el código de la aplicación. Además, en este capítulo se resumen los resultados de una encuesta anónima que se llevó a cabo con el objeto de validar la aplicación de la propuesta al dominio de los sistemas domóticos.

Capítulo 9. Análisis de resultados, conclusiones y vías futuras. Este capítulo resume a modo de conclusiones las principales aportaciones de la tesis, así como las publicaciones que avalan las técnicas y el proceso de DSDM a partir de los modelos de requisitos. Por último, se presentan las líneas de trabajo futuras.

Bibliografía y referencias En este capítulo se presenta la bibliografía y referencias que se citan a lo largo de la tesis doctoral

Anexos Por último, se detallan los anexos que complementan la información dada en los capítulos de la tesis, de los que se ha omitido para favorecer la legibilidad del documento.

ANEXO I: Trabajos excluidos en la RSL acerca de los metamodelos de requisitos.

Capítulo 2

Estrategia de investigación

2.1. Introducción

De acuerdo con Mary Shaw [316, 318], el objetivo de la investigación en Ingeniería del Software es, por lo general, resolver los problemas que surgen al utilizar y poner en producción el software del mundo real. Los investigadores en este campo buscan, por tanto, formas mejores de desarrollar y mantener el software. Según esta autora, el paradigma de investigación en IS está definido de acuerdo a tres dimensiones: el problema a resolver, el resultado de la investigación y la técnica de validación.

En cuanto al problema a resolver, las investigaciones en IS varían desde métodos de desarrollo o análisis, el diseño o evaluación de una instancia particular, a la generalización de técnicas o estudios de viabilidad. De acuerdo con Lázaro y Marcos [182], la naturaleza del problema de investigación determinará la manera en la que se debe abordar el proceso de investigación. Estos autores clasifican los problemas de investigación en IS atendiendo a su naturaleza en: problemas de *naturaleza científica* o problemas de *naturaleza ingenieril*. Los problemas de naturaleza científica son aquellos relacionados con la base teórica de la IS que tratan con el estudio de objetos y fenómenos existentes física, metafísica o conceptualmente. Por su parte, los problemas de naturaleza ingenieril son aquellos que tienen como objetivo la construcción de nuevos artefactos software y, por tanto, se basan en la manera de hacer las cosas y cómo crear nuevos objetos.

El resultado de la investigación comprende tanto el método de investigación utilizado como el producto obtenido. El método de investigación que se vaya a utilizar para resolver el problema detectado depende tanto de la naturaleza del problema como del resultado tangible que se espera obtener. Los resultados de una investigación en IS podría ser procedimientos o técnicas de análisis o desarrollo, modelos generalizados a partir de ejemplos específicos, herramientas o soluciones específicas para un problema particular.

Por último, es esencial seleccionar la técnica de validación adecuada atendiendo al resultado obtenido y al método de investigación utilizado. No es suficiente con presentar el resultado de la investigación, sino que hay que proporcionar suficientes evidencias que lo respalden. Las técnicas de validación pueden variar desde la persuasión, la utilización de métodos analíticos, la implementación de un prototipo, evaluación frente a criterios establecidos o la experiencia de uso en casos reales. De acuerdo con Wohlin *et al.* [361], en una disciplina como la IS es crucial el uso de métodos empíricos (como experimentos controlados, casos de estudio, encuestas o análisis post-mortem), que nos permitan evaluar y validar los resultados de la investigación. En IS, los casos de estudio son particularmente importantes para la evaluación de métodos y herramientas porque pueden evitar los problemas de escala que normalmente se asocian con los experimentos [169]. En general, es conveniente utilizar los casos de estudio para resolver las preguntas “¿Cómo?” o “¿Por qué?” en relación a eventos sobre los que el investigador tiene poco o ningún control [363]. Los casos de estudio también son apropiados para responder a la pregunta “¿Cuál es mejor?” [169].

De acuerdo con Johnson [150], la validación de la propuesta debe cubrir: 1) *usabilidad*, normalmente se hace con entrevistas, cuestionarios o bien observaciones grabadas, 2) *efectividad*, implica algún tipo de comparación para ver si la tecnología desarrollada produce algún cambio o efecto, y 3) *generalidad*, ver si proporciona infraestructura para una investigación más amplia en IS.

Para configurar una buena estrategia de investigación debemos combinar adecuadamente, al menos un elemento de cada una de las dimensiones que acabamos de presentar, teniendo en cuenta que no todas las combinaciones son válidas. Por ejemplo, si el problema a resolver fuese: “¿Podríamos mejorar el método de desarrollo X?”, el resultado de la investigación podría ser una nueva técnica que supuestamente mejore X, pero para validar este resultado, no sería suficiente con utilizar la persuasión, esto es, demostrar, implementando una herramienta que soporte la técnica, que la técnica funciona, sino que habría que utilizar algún otro método que nos permita obtener evidencias de que efectivamente la nueva técnica desarrollada mejora X de alguna manera. La persuasión rara vez debe utilizarse como método de validación salvo en el caso de que la pregunta de partida sea estudiar la viabilidad de una propuesta [316].

Siguiendo las recomendaciones de [316], en la siguiente sección vamos a detallar la estrategia de investigación seguida en la presente tesis (Sección 2.2) y, a continuación, describiremos brevemente los métodos de investigación utilizados: método constructivo (Sección 2.3), revisiones sistemáticas de la literatura (Sección 2.4) y métodos empíricos para la evaluación de resultados (Sección 2.5): caso de estudio y encuesta.

2.2. Estrategia de investigación en el desarrollo de la tesis

La naturaleza de nuestra investigación es puramente ingenieril, dado que el problema que se quiere resolver es la integración de la etapa de Ingeniería de Requisitos (IR) en un enfoque de Desarrollo de Software Dirigido por Modelos (DSDM). Para ello se propondrán, como resultado de la investigación, nuevos procedimientos y técnicas para la especificación de requisitos que posibilite dicha integración. En concreto, como iremos presentado en el desarrollo de la tesis, la propuesta incluye: 1) la

definición de un metamodelo de requisitos a partir del cual puedan definirse modelos de requisitos, 2) transformaciones de los modelos de requisitos a modelos conformes a un DSL (lenguajes específico de dominio) para el dominio concreto de los sistemas domóticos de manera que se integren los requisitos en enfoque de modelado específico de dominio y 3) una herramienta de soporte.

Para abordar este problema utilizaremos un *enfoque de investigación constructivo* [197] cuyo objeto es producir artefactos innovadores, modelos de requisitos en el caso de nuestra investigación, con la intención de solucionar un problema que existe en el mundo real y, de esta forma, contribuir a la teoría de la disciplina en la que se aplica. Lo característico de los artefactos construidos es que se inventan y desarrollan, no se descubren (una descripción más detallada del proceso de investigación se presenta más adelante en la sección 2.3).

La nueva técnica que desarrollemos será validada a través de la experiencia, utilizando métodos empíricos (sección 2.5) como son: 1) *caso de estudio*, en el que se utilicen los resultados en un caso real y 2) *encuesta*, para evaluar el grado de satisfacción de los usuarios potenciales de la propuesta. La elección de métodos empíricos como métodos de validación viene motivada por el tipo de problema de investigación que queremos resolver, puesto que, de acuerdo con [363], la estrategia de caso de estudio es la preferida cuando la pregunta de investigación que se plantea es “cómo” o “por qué”. En nuestro caso, la pregunta de investigación planteada de forma general es “¿Cómo podemos integrar los requisitos textuales en el DSDM?”. Por otro lado, Mary Shaw [317, 318] sugiere que los modelos conceptuales, en nuestro caso el metamodelo de requisitos, se deben evaluar mostrando ejemplos de su uso en problemas reales.

En la Tabla 8 podemos encontrar de manera resumida la estrategia de investigación seguida para el desarrollo de esta tesis doctoral.

Problema a resolver:	Integración de los requisitos textuales en un enfoque de DSDM	
Resultado de la investigación:	Método de investigación	Constructivo
	Producto	Proceso DSDM desde los requisitos hasta el código: • Metamodelo de requisitos • Transformaciones de modelos • Herramienta de soporte
Validación:	Métodos empíricos: caso de estudio y encuesta	

Tabla 8 Estrategia de investigación de la tesis

2.3. Método de investigación constructivo

Como adelantábamos en la sección anterior, el enfoque de investigación constructivo [197] es un procedimiento de investigación para producir artefactos (modelos, diagramas, planes, etc.) con la intención de solucionar un problema práctico relevante. Una vez construido, se debe analizar y comprobar la utilidad práctica de la solución junto con la contribución teórica al campo donde se está aplicando. Las características fundamentales de este enfoque de investigación (Figura 10) requiere que:

1) se centre en problemas del mundo real que se consideran relevantes para que sean resueltos de manera práctica, 2) produzca un artefacto innovador con el objeto de solucionar el problema detectado, 3) se incluya la implementación del artefacto desarrollado y, de este modo, una prueba de su aplicación práctica, 4) suponga una estrecha colaboración entre los investigadores y los interesados en el resultado de la investigación, 5) se ligue explícitamente al conocimiento teórico previo y que 6) se preste especial atención en hacer explícitas las aportaciones de la investigación a la teoría.

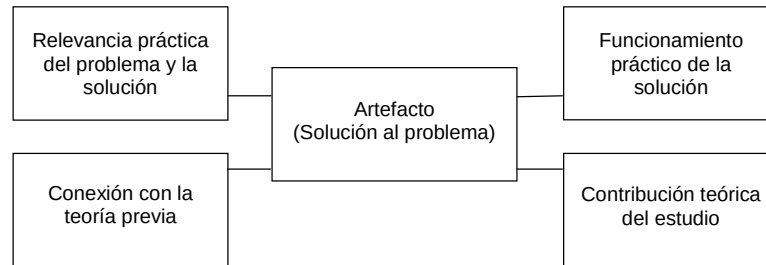


Figura 10 Características del enfoque de investigación constructivo [197].

Un proceso típico siguiendo el método de investigación constructivo consta de siete etapas, que de manera resumida se muestra en la Figura 11. En primer lugar se debe encontrar un problema práctico relevante cuya resolución también contribuya teóricamente al campo de estudio. Lo ideal es que se trate de un tema que no se haya analizado con profundidad en la literatura que demuestre la importancia práctica y teórica de ser resuelto. Este ha sido el objetivo del Capítulo 1 de esta tesis, donde se ha tratado de justificar la relevancia del tema que se va a abordar y la contribución teórica de los resultados obtenidos. La justificación se apoya en referencias a la literatura que muestran la escasez de trabajos en relación a la integración de los requisitos en el DSDM. Además, se debe conocer en profundidad las propuestas existentes en el área de nuestra investigación, en concreto debemos conocer las propuestas de integración de los requisitos textuales en el DSDM, en particular los metamodelos de requisitos existentes y las transformaciones desde especificaciones de requisitos textuales a otros modelos del desarrollo. Para ello se han elaborado sendas revisiones sistemáticas de la literatura que aportan rigurosidad al estudio del estado del arte (Sección 2.4), cuyos resultados se presentan en el Capítulo 3.

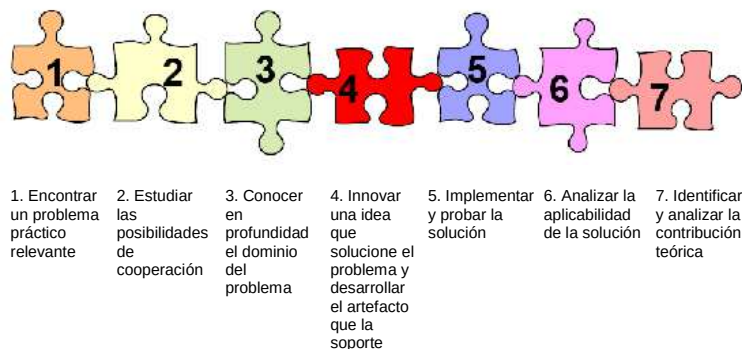


Figura 11 Proceso de investigación constructivo [197].

Una vez analizado el dominio del problema, se debe desarrollar el artefacto que soporte la idea innovadora que soluciona el problema. Una vez conocidos los modelos de referencia (metamodelos) propuestos en la literatura para la integración de los requisitos en el DSDM, se define uno que integra todos los elementos que consideramos fundamentales en un proceso de IR (Capítulo 5). Este metamodelo sirve como base para: 1) la definición de transformaciones de modelos que integre los modelos de requisitos en un proceso de DSDM, en concreto en un enfoque de modelado específico del dominio (Capítulo 6); y 2) la construcción de un prototipo de herramienta que nos permita especificar modelos de requisitos en los dos aspectos implicados en la reutilización, definición de modelos para reutilizar y definición de modelos con reutilización (Capítulo 7). La propuesta ha sido aplicada a un dominio real concreto, el de los sistemas domóticos y ha sido evaluada por los expertos tanto de la tecnología utilizada como del dominio de aplicación (Capítulo 8). Tras la validación de la propuesta, se ha analizado la aplicabilidad de la solución de manera general a otros casos de estudio y su potencial para que sirva como base a otras investigaciones más amplias. Por último, se analizan las contribuciones teóricas que quedan reflejadas en las publicaciones derivadas de los resultados de la tesis doctoral. Tanto las conclusiones como los resultados derivados de la presente tesis doctoral se presentan en el Capítulo 9. El resumen del proceso de investigación constructivo junto con la manera en la que se aborda cada etapa en esta tesis y los capítulos donde se puede encontrar los resultados de cada una de ellas se muestra en la Tabla 9.

2.4. Revisión sistemática de la literatura

Observamos que cada vez más autores utilizan revisiones sistemáticas de la literatura (RSL) para evaluar e interpretar el estado del arte en relación a un tema particular en un área de interés. Por ejemplo, en la revista *Information and Software Technology* animan a que se envíen RSL y cuenta con B. Kitchenham como editora, quien ha adaptado las guías para el desarrollo de una RSL a la investigación en IS [164]. Aunque el desarrollo de una RSL es considerablemente más costoso que una revisión tradicional, el seguir un proceso sistemático le confiere rigurosidad al estudio y valor a las conclusiones extraídas del mismo. Así, nosotros utilizaremos este método científico para explorar el estado del arte de las áreas de interés que confluyen en la temática de la tesis (Capítulo 3). La descripción de este método de investigación esta basada en guías propuestas por Kitchenham [164].

Una RSL se puede abordar para resumir las evidencias que existen en relación a alguna cuestión, por ejemplo, un tratamiento, una tecnología o una metodología. También resulta de utilidad para identificar vacíos en algún área de investigación que sugiera la oportunidad para iniciar nuevas investigaciones o para proporcionar un marco en el que situar nuevas actividades de investigación. Además también se pueden acometer para evaluar hasta que punto las evidencias empíricas soportan o contradicen hipótesis teóricas o incluso asiste en la generación de nuevas hipótesis.

El proceso de investigación constructivo	Encontrar un problema práctico relevante	Conocer en profundidad el dominio del problema	Desarrollar el artefacto que solucione el problema	Implementar y probar la solución	Analizar la aplicabilidad de la solución	Identificar y analizar la contribución teórica
Etapas del proceso aplicadas a la tesis	Revisión bibliográfica para justificar: • Interés práctico de la integración de los requisitos en un enfoque DSDM • Carencia de enfoques que lo soporten que motiva el interés teórico	Revisión sistemática de la literatura sobre la integración de requisitos en un enfoque DSDM: • Metamodelos de requisitos • Transformaciones a partir de modelos de requisitos	En base al conocimiento adquirido: • Definir el proceso de IR dirigido por modelos centrado en reutilización • Definir el metamodelo de requisitos • Definir las transformaciones desde los modelos de requisitos.	• A partir del modelo de requisitos construir un prototipo de herramienta que lo soporte • Validar la propuesta aplicándolo a un caso de estudio	Discutir cómo de general es la propuesta: • ¿Se puede aplicar a otros casos de estudio? • Posibilidad de que sea utilizada como base para otras investigaciones	Contrastar los resultados obtenidos con la revisión bibliográfica previa para destacar las contribuciones.
Capítulo de la tesis donde se discute el resultado de cada etapa.	Capítulo 1	Capítulo 3	Capítulo 4 Capítulo 5 Capítulo 6	Capítulo 7 Capítulo 8	Capítulo 9	Capítulo 9

Tabla 9 Aplicación del proceso de investigación constructivo a esta tesis

La RSL consta de tres etapas: *planificación*, *ejecución* e *informe* (ver Tabla 10). Aunque estas etapas podrían parecer secuenciales, es importante destacar que muchas de ellas se tienen que abordar de manera iterativa. En particular, muchas actividades se inician durante la etapa de desarrollo del protocolo y se refinan cuando se lleva a cabo la revisión propiamente dicha, por ejemplo, la selección de estudios primarios se realiza en base a los criterios de inclusión y exclusión definidos en el protocolo de la revisión, pero podría refinarse después de definir los criterios de calidad.

1. Planificación
1.1 Identificación de la necesidad de la revisión
1.2 Encargar la revisión (opcional)
1.3 Especificar la(s) pregunta(s) de investigación
1.4 Desarrollo del protocolo de la revisión
1.5 Evaluar el protocolo de la revisión (opcional)
2. Ejecución
2.1 Identificación de los estudios relevantes sobre el tema
2.2 Selección de estudios primarios
2.3 Evaluación de la calidad de los estudios primarios
2.4 Extracción de datos de los estudios primarios
2.5 Síntesis de datos
3. Informe
3.1 Especificar los mecanismos de diseminación de los datos
3.2 Formatear el informe principal
3.3 Evaluar el informe (opcional)

Tabla 10 Etapas del proceso de una RSL.

Una de las características fundamentales que diferencian una revisión sistemática de una revisión de la literatura convencional es que debe comenzar con la definición de un protocolo que especifica la pregunta de investigación que se va a abordar y los métodos que se van a utilizar para ejecutar la revisión. La definición de un protocolo es necesaria para reducir la posibilidad de sesgos por parte de los investigadores, por ejemplo, evita que la selección de los estudios o el análisis esté guiada por las expectativas de los investigadores. El protocolo incluye: 1) la justificación para acometer la RSL, 2) la(s) pregunta(s) de investigación que se pretende(n) resolver, 3) las fuentes y el método de búsqueda de los estudios preliminares, 4) los criterios y procedimiento de selección de los estudios preliminares y 5) los criterios de calidad que serán aplicados para seleccionar los estudios de los cuales vamos a analizar la información. En cierto modo, los criterios de calidad que se establezcan servirán como marco de referencia para comparar los estudios finalmente seleccionados. El protocolo es un elemento crítico en una RSL y por tanto, los investigadores deben establecer un procedimiento para revisar el protocolo, por ejemplo, solicitando la revisión a un grupo de expertos independientes.

Una vez establecido el protocolo en la fase de planificación, puede comenzar la RSL. Se buscarán los estudios preliminares en las fuentes determinadas en el protocolo siguiendo el procedimiento establecido y se obtendrá una primera lista de estudios tras aplicar los criterios de selección. Esta lista inicial será refinada aplicando los criterios de calidad establecidos a priori. De esta forma, finalmente seleccionamos el conjunto de

trabajos de los cuales vamos a extraer la información de acuerdo con el objetivo de la RSL.

Por último, una vez extraída la información de los estudios seleccionados, se debe resumir y analizar. Los resultados se deben presentar en tablas para favorecer su análisis puesto que permite clasificar los estudios atendiendo a distintos criterios y organizarlos de acuerdo a diferentes perspectivas. Es importante comunicar los resultados de una RSL de forma efectiva bien mediante un informe técnico o como sección de una tesis doctoral, como es nuestro caso, o bien como artículo en una conferencia o revista.

En esta tesis doctoral se ha utilizado la RSL para conocer el estado del arte en cuanto a los metamodelos de requisitos y las propuestas de transformación desde modelos de requisitos a otros modelos del desarrollo. Los resultados de los procesos de revisión se presentan en el Capítulo 3.

2.5. Métodos empíricos para la evaluación de resultados

Un estudio empírico es simplemente una prueba que compara lo que creemos con lo que observamos [266]. Sin embargo, cuando esta prueba se construye y se ejecuta con prudencia y cuando se utiliza para soportar el método científico, juega un papel fundamental en la ciencia moderna. Concretamente, ayuda a entender cómo y por qué las cosas funcionan, y nos permite utilizar este conocimiento para alterar materialmente nuestro mundo. La esencia de un estudio empírico es tratar de aprender algo útil comparando la teoría y la realidad, y mejorar nuestras teorías como resultado.

Según Wohlin *et al.* [361], para una disciplina como la IS son cruciales los métodos empíricos puesto que permiten incorporar el comportamiento humano en el enfoque de investigación que se haya tomado. No es suficiente con medir el software, sino que son necesarios métodos empíricos como *experimentos controlados*, *casos de estudio*, *encuestas* o *análisis post-mortem*, para ayudar a evaluar y validar los resultados de la investigación. Con estos métodos se puede establecer científicamente si una cosa es mejor que otra. Por tanto, los métodos empíricos proporcionan una importante base científica para la IS.

Existen fundamentalmente dos paradigmas de investigación que toman diferentes enfoques para los estudios empíricos: investigación *cuantitativa* e investigación *cualitativa*. La investigación cualitativa se ocupa de estudiar los objetos en su entorno natural. Un investigador cualitativo trata de interpretar un fenómeno en base a las explicaciones que la gente le da de él. La investigación cualitativa comienza aceptando que existe un abanico de formas diferentes de interpretación y se ocupa de descubrir las causas observadas por los sujetos del estudio y entender su punto de vista sobre el problema. El sujeto es la persona que está participando en el estudio para evaluar un objeto.

Por su parte, la investigación cuantitativa fundamentalmente trata de cuantificar una relación o comparar dos o más grupos. El objetivo es identificar una relación causa-efecto. La investigación cuantitativa se lleva a cabo con frecuencia mediante experimentos controlados o recogiendo datos a través de casos de estudio.

Es posible que el mismo hecho se investigue utilizando métodos cualitativos y cuantitativos, pero cada uno de ellos abordarán un tipo de pregunta diferente. Las investigaciones cuantitativas son adecuadas para comprobar el efecto de alguna manipulación o actividad, mientras que un estudio cualitativo de las creencias o entendimiento es apropiado para encontrar por qué los resultados de una investigación cuantitativa son los que son.

En general, un estudio empírico comprende los siguientes pasos: definición, planificación, operación, análisis e interpretación, conclusiones y presentación. El trabajo que se realice en cada uno de estos pasos depende del tipo de estudio empírico seleccionado.

Según Perry *et al.* [266], aunque la IS empírica ha madurado en los últimos años, existen serios problemas derivados de malentendidos sobre qué son los estudios empíricos y por qué los utilizamos. Nótese que cuando hablamos de IS empírica nos referimos a aplicar los métodos empíricos para validar los resultados de las investigaciones en IS. Entre los principales problemas señalados por los autores encontramos que demasiados estudios empíricos: 1) simplemente confirman lo que es obvio de manera intuitiva; 2) sólo presentan datos que no ayudan a entender por qué los datos son como son, los datos deben servir para responder preguntas, no sólo para rellenar gráficos; 3) carecen de hipótesis, por tanto, al final del estudio sólo se pueden presentar observaciones sobre los datos o 4) no se pueden extraer conclusiones que relacionen los resultados con la teoría o la práctica. Perry *et al.* esbozan la estructura general de los estudios de software empíricos, proporcionando pasos concretos para conseguir diseñar mejores estudios, de los que se puedan extraer conclusiones creíbles, obtener los datos e implicar a otros en nuestros estudios empíricos.

En esta línea, Kitchenham *et al.* [168] proporciona guías para mejorar la planificación, implementación y el informe de los resultados de una investigación empírica. Estas guías están basadas en la propia experiencia de los autores, como investigadores y revisores de estudios empíricos, y en recomendaciones procedentes de otras disciplinas, como la medicina, cuyos avances requieren estudios empíricos cuidadosos. De acuerdo con los autores, estas guías son importantes porque, por un lado, los investigadores en IS con frecuencia comenten errores estadísticos y, por otro lado, los investigadores senior insisten en que se aborden más investigaciones empíricas como base para la IS.

Para validar los resultados de esta tesis doctoral se han utilizado los métodos empíricos cualitativos de caso de estudio y encuestas que pasamos a presentar en las secciones siguientes.

2.5.1. Caso de estudio

El contenido de esta sección se basa en el trabajo de Runeson y Höst [298] que han sintetizado la información general sobre los casos de estudio procedentes fundamentalmente del campo de las ciencias sociales y sistemas de información, y las han adaptado a las necesidades de la IS.

Un caso de estudio es un método de investigación empírico cuyo objetivo es investigar un fenómeno en su contexto donde, en ocasiones, el límite entre el fenómeno y el contexto podría no estar claro. Para el estudio se utilizan diferentes fuentes de

evidencias y la información se recoge de personas, grupos y organizaciones. Este tipo de estrategia se caracteriza por carecer de control experimental. Este método de investigación ha recibido críticas como que los resultados obtenidos son de menos valor, que no se pueden generalizarse y que están sesgados por los investigadores. Estas críticas se pueden paliar aplicando una metodología de investigación adecuada.

Los casos de estudio se han utilizado habitualmente en disciplinas como la Psicología, Sociología, Ciencias Políticas, Trabajo Social y Ciencias Empresariales. En estas disciplinas, los casos de estudios se abordan con el objetivo de incrementar el conocimiento sobre individuos, grupos y organizaciones, y sobre fenómenos sociales y políticos. Las características de la disciplina de IS hace que sea razonable aplicar esta técnica de la misma forma que se viene haciendo en estas otras disciplinas. La IS implica el desarrollo, operación y mantenimiento del software y los artefactos relacionados. Por tanto, la investigación en IS, en gran medida, tiene como objetivo estudiar cómo los ingenieros de software y otros stakeholders conducen estos desarrollos, operaciones y mantenimientos bajo diferentes condiciones. El desarrollo de software lo llevan a cabo individuos, grupos y organizaciones, por tanto, cuestiones sociales y políticas son importantes para este desarrollo. Esto es, la IS es un área multidisciplinaria que incluye disciplinas donde se realizan normalmente casos de estudio. Esto significa que muchas preguntas de investigación en IS son adecuadas para la investigación mediante casos de estudio.

No obstante, las características de los objetos de estudio de IS son diferentes tanto de las ciencias sociales como de los sistemas de información. Los objetos de estudio son 1) corporaciones privadas o unidades de agencias públicas que desarrollan software en lugar de agencias públicas o corporaciones privadas que utilizan sistemas software; 2) orientado al proyecto en lugar de orientado a línea o función; y 3) el trabajo estudiado es un trabajo de ingeniería avanzado llevado a cabo por gente altamente preparada en lugar de trabajo rutinario. Además, la comunidad de investigadores en IS tienen una perspectiva pragmática y orientada a resultados de la metodología de investigación en lugar de una postura filosófica.

Las características claves de un caso de estudio son: 1) se trata de un método flexible, esto es, los parámetros del estudio pueden cambiar durante el curso del estudio, haciendo frente a las características complejas y dinámicas de un fenómeno del mundo real como la IS; 2) sus conclusiones se basan en una cadena clara de evidencias, ya sea un estudio cualitativo o cuantitativo; y 3) aumenta el conocimiento existente basándose en teorías previamente establecidas, si existen, o construyendo teorías.

El desarrollo de un caso de estudio comprende fundamentalmente cinco pasos:

1. *Diseño del caso de estudio*, que implica la definición de los objetivos y la planificación del caso de estudio
2. *Preparación para la recolección de datos*, que incluye la definición de los procedimientos y protocolos para la recopilación de los datos.
3. *Recopilación de evidencias*, que tiene que ver con la ejecución del caso de estudio con la colección de datos.
4. *Análisis de los datos recopilados*, que se realiza de forma diferente para datos cuantitativos y cualitativos. En el primer caso, normalmente el análisis incluye estadísticas descriptivas (por ejemplo, media, desviación estándar, histogramas), análisis de correlación, desarrollo de modelos predictivos y prueba de hipótesis.

Por otro lado, el objetivo del análisis de datos cualitativos es derivar conclusiones a partir de los datos, manteniendo una clara cadena de evidencias, esto es, el lector debería ser capaz de seguir la derivación de los resultados y conclusiones a partir de los datos recopilados. Esto significa que se debe presentar información suficiente acerca de cada paso del estudio y cada decisión que se haya tomado.

5. *Informe*, tanto para comunicar las conclusiones del estudio como para evaluar la calidad del mismo. Los informes podrían tener diferentes audiencias (investigador, patrocinador de la investigación, profesional de la industria, político) y por tanto, podríamos vernos obligados a escribir diferentes informes para diferentes audiencias.

Este proceso es prácticamente el mismo para cualquier tipo de estudio empírico. Sin embargo, puesto que el caso de estudio es una estrategia de diseño flexible, se lleva a cabo una cantidad significativa de iteraciones sobre estos pasos. La recopilación y análisis de los datos se debe abordar incrementalmente. Si los datos recopilados son insuficientes para el análisis, se puede planear recopilar más datos. Sin embargo, debe haber un límite a esta flexibilidad que viene dado por los objetivos establecidos desde el principio. Si cambian los objetivos, entonces estamos frente a un nuevo caso de estudio en lugar de un cambio en el actual.

En esta tesis doctoral se ha utilizado el dominio de los sistemas domóticos como dominio del caso de estudio. Se ha definido un modelo de requisitos reutilizables que constituye un catálogo de requisitos para el dominio de los sistemas domóticos. Reutilizando los requisitos de este catálogo se pueden definir los requisitos de una aplicación domótica concreta que incluya los elementos más comunes: seguridad, iluminación, clima, persianas y toldos. Una vez definidos los requisitos de la aplicación domótica se puede generar automáticamente el modelo de la aplicación que incluye los conceptos propios de los sistemas domóticos. El caso de estudio se presenta en el Capítulo 8.

2.5.2. Encuesta

El contenido de este apartado está extraído de la serie de artículos publicados por Kitchenham y Pfleeger [165-167, 267] sobre los principios de la investigación mediante el uso de encuestas publicados con el objeto de mejorar este tipo de investigación en el campo de la IS.

Una encuesta no es simplemente el instrumento que se utiliza para obtener la información, el cuestionario, sino que es un sistema completo para la recogida de información que permita describir, comparar o explicar conocimiento, actitudes y comportamiento. Por tanto, el cuestionario es parte de un proceso más amplio con actividades claramente definidas, como son:

1. Establecer objetivos concretos y medibles. Cada objetivo es simplemente una declaración de los resultados esperados de la encuesta. Es muy importante que la declaración de objetivos incluya la definición de los términos potencialmente ambiguos puesto que estas definiciones dirigen no sólo el diseño y desarrollo del cuestionario sino también la comprensión del encuestado sobre la terminología.
2. Planificar y programar la encuesta.

3. Asegurar que los recursos adecuados están disponibles.
4. Diseñar la encuesta. El proceso de diseño comienza revisando los objetivos, se debe examinar la población destino identificada por los objetivos y decidir cuál es la mejor manera de obtener la información necesaria para cumplir con los objetivos. Además también hay que tener en cuenta otros factores como el tamaño apropiado de la muestra, para que realmente la muestra sea representativa de la población, y el ratio de respuesta, que debemos asegurar que sea el más alto posible para que los resultados de la encuesta sean significativos.
5. Preparar el cuestionario para la recogida de datos. Para preparar el cuestionario se debe comenzar buscando en la literatura otros estudios realizados sobre el tema que se está investigando y analizar qué cuestionarios se utilizaron. Una vez conocidos los antecedentes, los objetivos definen el contexto de la encuesta y determinan qué áreas se deben cubrir y qué tipo de información necesitamos. Es esencial que las preguntas de la encuesta estén directamente relacionadas con los objetivos establecidos. Además, existen otros factores que se tienen que tener en cuenta a la hora de formular las preguntas: 1) las preguntas se deben formular de manera que el encuestado las entienda fácilmente y con exactitud; 2) el número de preguntas debe ser adecuado para el tiempo que el encuestado va a dedicar a la encuesta; 3) utilizar un formato de respuesta estandarizado (por ejemplo, encuesta tipo Likert [191] con cinco respuestas: Muy en desacuerdo, en desacuerdo, neutral, de acuerdo, muy de acuerdo) ayuda a reducir el tiempo de respuesta; 4) cada pregunta debe ser precisa y sin ambigüedades; 5) evitar el sesgo del investigador definiendo preguntas neutrales que no influya en la manera en la que el encuestado piensa sobre la problema y prestando atención al orden de las preguntas de manera que la respuesta de una no influya en la respuesta de la siguiente; 6) valorar qué tipo de pregunta es más adecuada: abierta o cerrada, teniendo en cuenta que las preguntas cerradas (en las que están restringidas las respuestas) son más fáciles de analizar; 7) hay que cuidar tanto el formato del cuestionario como el formato de las instrucciones.
6. Validar el cuestionario. El objetivo de la evaluación del cuestionario es comprobar que las preguntas son entendibles, evaluar el ratio de respuesta probable y la efectividad de los siguientes procedimientos, evaluar la fiabilidad y validez del cuestionario y asegurar que las técnicas de análisis de datos se ajustan a las respuestas esperadas.
7. Seleccionar a los participantes. En esta actividad se trata de seleccionar una muestra válida, esto es, representativa, de la población objetivo. Para obtener la muestra se pueden utilizar métodos de muestreo probabilísticos (como el muestreo aleatorio simple o el muestreo aleatorio estratificado) o no probabilísticos (como el muestreo por conveniencia o el muestreo de bola de nieve). Estos últimos se utilizan cuando se seleccionan los encuestados porque son de fácil acceso o los investigadores consideran que son representativos. Este tipo de muestreo corre el riesgo de no ser objetivos de manera que es peligroso extraer cualquier conclusión firme a partir de los datos. Otro factor importante es el tamaño de la muestra. Un tamaño inadecuado nos puede llevar a

resultados que no son estadísticamente significativos o a que no podamos hacer comparaciones o contrastar datos entre diferentes subconjuntos de la población.

8. Administrar y anotar el cuestionario
9. Analizar los datos
10. Hacer el informe con los resultados

En esta tesis doctoral se ha utilizado una encuesta tipo Likert de cinco respuestas para evaluar el grado de aceptación de la propuesta de transformación de modelos de requisitos en modelos de una aplicación domótica, en el marco de un proceso completo de DSDM (desde los requisitos hasta el código). Los detalles de la encuesta y los resultados se presentan en el Capítulo 8.

Capítulo 3

Análisis del estado del arte

3.1. Introducción

Resulta imprescindible realizar un estudio exhaustivo de las propuestas existentes en la literatura en relación con el tema que vamos a abordar en nuestra investigación. El objetivo de este estudio será justificar tanto el interés de la investigación, como las aportaciones con las que se van a contribuir en una determinada área de conocimiento.

Dado que el objetivo marcado para la presente tesis doctoral es la integración de los requisitos textuales en el Desarrollo de Software Dirigido por Modelos (DSDM) utilizando las técnicas que nos ofrece este enfoque (metamodelado y transformaciones de modelo) se nos abren dos vías de investigación que abordaremos realizando sendas Revisiones Sistemáticas de la Literatura (RSL) siguiendo las recomendaciones de Kitchenham [164] (el lector puede encontrar un resumen del proceso para abordar una RSL en la Sección 2.4). Por un lado, debemos estudiar qué metamodelos basados en requisitos textuales han sido propuestos, cuáles son sus características, si soportan o no las buenas prácticas de ingeniería de requisitos (IR) y si se integran en un proceso de DSDM. Por otro lado, debemos analizar qué tipo de transformaciones se han definido para obtener otros modelos del desarrollo (cómo modelos de análisis) a partir de los modelos de requisitos, prestando especial atención a si estos modelos de partida están basados o no en requisitos textuales y en caso de no estarlo si están ligados de algún modo a la especificación de requisitos textual. Las conclusiones de estos estudios nos van a servir para encontrar aspectos que necesitan ser investigados y que están dentro de los objetivos marcados en esta tesis doctoral.

Recientemente se han publicado diferentes trabajos relacionados con estas dos vías de estudios secundarios que vamos a abordar: 1) se ha elaborado una RSL respecto al uso de las técnicas de IR en el DSDM [194]; 2) se ha analizado la integración de los requisitos y los modelos en el DSDM [374] y 3) se ha desarrollado una RSL de los enfoques de transformación entre requisitos de usuario y modelos de análisis [368]. Las revisiones realizadas en esta tesis en relación a los metamodelos de requisitos y las transformaciones desde modelos de requisitos, que se presentan en las Secciones 3.2 y 3.3, respectivamente, aportan nuevos puntos de vista que complementan a los que ya han sido publicados. En cuanto a la revisión de los metamodelos de requisitos, se

proporciona un análisis más técnico de los conceptos y relaciones de traza que incluyen los metamodelos desde el punto de vista de las buenas prácticas de la IR que el ofrecido en [194] y [374]. En relación al estudio de las transformaciones, la revisión que hemos realizado, y que se presenta en el apartado 3.3, se centra en los enfoques de transformación en el marco del DSDM que parten de modelos de requisitos (no desde requisitos de usuario en general como en [368]) para analizar hasta qué punto los requisitos textuales son tenidos en cuenta en estos enfoques. A continuación se resumen brevemente las principales conclusiones extraídas de estos estudios y las contribuciones de esta tesis para cubrir las deficiencias detectas.

3.1.1. Uso de técnicas de Ingeniería de Requisitos en el DSDM

La revisión sistemática realizada por Loniewski *et al.* [193] trata de analizar qué técnicas de IR se han empleado en los enfoques de DSDM y cuál es el nivel actual de automatización. Se analizaron 72 artículos de los 884 seleccionados inicialmente en busca de: 1) tipos de requisitos (requisitos software o del negocio), 2) estructura de los requisitos (modelos o lenguaje natural), 3) transformaciones de modelos que proporciona, 4) nivel de la transformación (endógena o exógena, de acuerdo con la taxonomía propuesta por Mens y van Gorp [213]), 5) lenguaje de transformación utilizado, 6) nivel de automatización de la transformación, 7) trazabilidad post-RS (*post Requirements Specification*, trazabilidad de los requisitos hacia otros artefactos del desarrollo de acuerdo a la clasificación propuesta por Gotel and Finkelstein [111]), 8) automatización de la trazabilidad, 9) herramienta de soporte, 10) técnica de validación (encuesta, caso de estudio o experimento) y 11) contexto de uso actual (industrial o académico). Las principales conclusiones a las que llegan los autores de este estudio son:

- C1) Todavía no existe una solución completa que integre los modelos de requisitos como parte del proceso de DSDM.
- C2) El soporte de herramientas para la gestión de modelos de requisitos y su uso posterior de manera automática es escaso.
- C3) Las actividades de IR casi nunca se incluyen en la mayoría de los procesos de transformación de modelos y generación de código.
- C4) Las transformaciones automáticas de modelos se utilizan de forma escasa dado que la mayoría de las propuestas establecen algún tipo de mapeo utilizando algún lenguaje no estándar.
- C5) Los modelos no se utilizan con la frecuencia esperada, sino que los requisitos expresados en lenguaje natural son también muy importantes. Los autores distinguen entre modelos *estructurales*, de *comportamiento*, *funcionales* y *otros*, expresados en un lenguaje estándar (UML) o en cualquier otro lenguaje.
- C6) La trazabilidad post-RS no se define correctamente. No existe una propuesta completa que esté bien documentada y evaluada.
- C7) No existen estudios empíricos en entornos de DSDM que muestren los beneficios del uso de técnicas de IR como parte de un proceso de transformaciones automáticas.

La propuesta que se presenta en esta tesis doctoral trata de cubrir los problemas detectados por Loniewski *et al.* [193], puesto que trata de proporcionar un metamodelo de requisitos basado en lenguaje natural (soluciona C5) que incluya todas las relaciones de

traza tanto post-RS como pre-RS (soluciona C6) de acuerdo a las buenas prácticas de IR y la definición de reglas de transformación automáticas (soluciona C4) desde los modelos de requisitos a modelos de análisis conforme a un Lenguaje Específico de Dominio (DSL, *Domain Specification Language*). El metamodelo de requisitos y las reglas de transformación son las piezas claves para la integración de los requisitos textuales en un enfoque completo de DSDM (soluciona C1 y C3) que será soportado por una herramienta (soluciona C2) y validado utilizando métodos de validación empíricos (soluciona C7).

3.1.2. Integración de los requisitos y modelos en el DSDM

De acuerdo con Zikra *et al.* [374] a pesar de la relación estrecha que existe entre los modelos y los requisitos, sólo unos pocos enfoques de DSDM proporcionan las guías metodológicas necesarias y la herramienta de soporte que faciliten dicha relación. Los autores analizan los enfoques existentes en cuanto a la integración de los requisitos con modelos en un enfoque de DSDM utilizando como marco de comparación las propiedades generales que, a juicio de los autores, tiene que cumplir cualquier enfoque que tenga en cuenta la integración de requisitos y modelos. Zikra *et al.* clasifican los enfoques estudiados en: 1) *enfoques de integración basados en lenguaje natural*, donde los requisitos se capturan en lenguaje natural y se utilizan técnicas de procesamiento de textos (análisis sintáctico y semántico) para obtener los modelos; 2) *enfoques de integración basados en guías*, estos enfoques proporcionan guías para la construcción de los modelos de requisitos a partir de requisitos textuales que puedan servir como entrada para un proceso de DSDM, sin dar detalles sobre este proceso (*enfoques de integración de extremo*), para la definición del modelo de requisitos y la transformación (manual o automática) en el modelo inicial del DSDM (*enfoques de integración parcial*) o bien proponen un enfoque completo de DSDM que incluye la integración de los requisitos (*enfoques de integración total*), y 3) *enfoques de integración basados en trazabilidad*, en los que simplemente se establecen los enlaces entre los requisitos y los modelos posteriores de desarrollo una vez que los requisitos han sido definidos y los modelos han sido creados. El marco de comparación de las propuestas incluye: 1) soporte a los requisitos que definen tanto los aspectos dinámicos como estáticos del sistema de información, 2) soporte a los aspectos intencionales de los requisitos (objetivos o *goals*), 3) guías de diseño para la obtención del modelo inicial del DSDM a partir de los requisitos, 4) técnicas de verificación de la completitud de los modelos, 5) soporte a los aspectos de la arquitectura en los requisitos, 6) propagación de cambios entre requisitos y modelos de DSDM, 7) reutilización de modelos de DSDM y 8) herramienta de soporte.

Las principales conclusiones que se extraen del estudio realizado por Zikra *et al.* [374] son que: 1) no existe consenso en cuanto a las bases de la integración de los requisitos y los modelos, 2) se necesitan metamodelos de requisitos más elaborados para diseñar guías de transformación explícitas y 3) es limitado el soporte que proporcionan las herramientas.

El trabajo de esta tesis doctoral se encuadraría dentro de los enfoque de integración basados en guías, en concreto, en los enfoques de integración total puesto que se pretende integrar los requisitos en un enfoque de desarrollo específico de dominio

(DSM, *Domain Specific Modeling*) que cuente con un editor de modelos y las reglas de transformación para generar el código a partir de los modelos conformes a un DSL. De esta forma, los requisitos se trazarían de forma automática a los modelos de dominio y a partir de estos se generaría el código de la aplicación, proporcionando un enfoque completo de DSDM desde los requisitos hasta el código. De esta forma, esta tesis contribuye a solucionar los problemas detectados en el estudio realizado por Zikra *et al.* [374] proporcionando un proceso de DSDM basado en un metamodelo de requisitos elaborado que de soporte a las buenas prácticas de IR y permita la transformación de los modelos de requisitos en modelos conformes a un DSL. El proceso contará con una herramienta de soporte que permita tanto la definición de modelos como las transformaciones a partir de estos.

3.1.3. Enfoque de transformación entre requisitos de usuario y modelos de análisis

De acuerdo con Yue *et al.* [368], dado que los requisitos se especifican principalmente de forma textual y no mediante un modelo que pueda ser entendido fácilmente por un ordenador, las transformaciones automáticas a partir de los requisitos son difíciles de conseguir. A juicio de los autores, este puede ser uno de los motivos por los que la transformación desde los requisitos a los modelos de análisis no forma parte del ciclo de vida de la propuesta para el DSDM del OMG MDA (*Model Driven Architecture*) [170]. Los autores llevan a cabo una RSL para determinar las propuestas utilizadas para transformar requisitos en modelos de análisis (como diagramas UML o modelos entidad relación) con el objeto de conocer cuáles son las limitaciones actuales de dichos enfoques y las cuestiones abiertas que pueden dar lugar a futuras investigaciones. De forma más detallada, la información que se quiere obtener del estudio incluye: 1) representación de los requisitos (por ejemplo, casos de uso o especificaciones textuales), 2) modelo de análisis generado (como diagrama UML o diagrama entidad relación), 3) modelo intermedio utilizado en la transformación, 4) automatización del proceso (automático, semi-automático, manual), 5) pasos que se siguen en el proceso, 6) gestión de la trazabilidad y 7) validación de la propuesta.

Finalmente, los autores analizaron 16 enfoques (20 estudios primarios) de 361 artículos encontrados inicialmente, y fueron evaluados en términos de sus entradas (dificultad para documentar los requisitos), sus salidas (completitud del modelo de análisis generado) y el enfoque de transformación desde los siguientes aspectos: automatización, eficiencia, evaluación, capacidad de trazabilidad, estructuración de las reglas de transformación y completitud de las reglas de transformación. En base a los resultados obtenidos, desde el punto de vista de los autores, *un enfoque deseable para la transformación de requisitos en modelos de análisis es aquel que permita transformar de manera automática y eficiente un modelo de casos de uso utilizando restricciones razonables sobre el lenguaje natural, con o sin información específica de dominio proporcionada en un glosario, en modelos UML correctos y consistentes que comprendan tanto los aspectos estructurales como de comportamiento utilizando un modelo intermedio y técnicas de pre-procesamiento de requisitos totalmente automatizables.*

Los resultados del estudio muestran que los trabajos analizados utilizan cuatro tipos de enfoques de transformación: basadas en reglas, basadas en ontologías, basadas en patrones y transformaciones de identidad. Los autores sugieren que se utilicen las técnicas de transformación modelo a modelo (como ATL, *Atlas Transformation Language* [1]) para implementar el enfoque de transformación de requisitos a modelos de análisis de manera que los algoritmos y reglas de transformación se estructuren y especifiquen claramente.

De la revisión también se concluye que la mayoría de las propuestas no abordan la trazabilidad. Los autores sugieren que se debe proponer un mecanismo de trazabilidad para crear y mantener los enlaces entre los requisitos y los elementos del modelo de análisis. Por último, los autores proponen que las propuestas evalúen su calidad mediante estudios empíricos.

Si bien Yue *et al.* [368] apuntan al modelo de casos de uso como modelo inicial a partir del cual se puedan generar los modelos de análisis, el objetivo de la presente tesis doctoral es más ambicioso, en el sentido de que trata de definir modelos de requisitos basados en lenguaje natural. A partir de estos modelos, y utilizando las técnicas de transformación de modelos que proporciona el marco de DSDM, obtendremos el modelo de análisis, que no será un modelo UML como proponen los autores de la RSL, sino un modelo conforme a un DSL. En el mismo proceso de transformación se generarán las relaciones de traza entre los requisitos y los elementos del modelo de análisis, tal y como se sugiere como resultado del estudio. La viabilidad de la propuesta será validada mediante métodos de evaluación empíricos.

3.2. RSL metamodelos de requisitos

Las diferentes propuestas de metamodelos de requisitos existentes presentan una gran disparidad en el nivel de abstracción, aspectos que cubren, el modelo de trazabilidad que soporta y la semántica. Como apuntábamos en el Capítulo 1 (Introducción), según Weber y Weisbrod [357], los distintos tipos de requisitos pueden llegar a ser confusos y, por tanto, hay una necesidad apremiante de establecer un metamodelo de requisitos que defina formalmente los conceptos y relaciones implicadas en el proceso de IR.

A la vista de esta situación, se ha llevado a cabo una RSL en el marco de esta tesis doctoral, con el objetivo de identificar y analizar los metamodelos de requisitos más relevantes, que permitan representar gráficamente las especificaciones de requisitos en lenguaje natural mediante modelos de requisitos. En la revisión se analizarán y se compararán los conceptos y relaciones propuestos en los metamodelos, así como su capacidad para integrar los modelos de requisitos en el enfoque DSDM. Nos centramos en modelos que representen las especificaciones de requisitos textuales porque es el formato más común en la industria [63] y, además, pensamos que utilizando este tipo de modelos se evita posibles inconsistencias que podrían aparecer entre los requisitos y los modelos especificados en un formalismo diferente, por ejemplo, modelos UML.

Las bondades de los metamodelos se estudiarán utilizando un marco de comparación basado en la trazabilidad de requisitos. Una de las razones que nos han llevado a establecer un patrón de modelo de trazabilidad es que las relaciones de traza requeridas,

entre los artefactos implicados en la especificación de requisitos (tales como requisito, stakeholder o caso de prueba), determinan, indirectamente, los conceptos que estas trazas enlazan. Así, analizando el marco de trazabilidad, se examinan al mismo tiempo las relaciones y los conceptos que forman parte del metamodelo. Por otro lado, dada la importancia de la trazabilidad de requisitos en el proceso de desarrollo de software [185], proporcionar un metamodelo de requisitos que defina un modelo de trazabilidad efectivo, representa un factor de éxito clave para asegurar una implementación del software de calidad. En consecuencia, es necesario definir cuidadosamente qué significa un modelo de trazabilidad *efectivo*. Este punto se aborda en la Sección 3.2.1.

Tras la presentación del marco de comparación de los metamodelos de requisitos, pasaremos a detallar la RSL. En primer lugar, se presenta el diseño del estudio (Sección 3.2.2). A continuación, se muestran los resultados de la RSL (Sección 3.2.3), tanto el resumen de los trabajos candidatos y finalmente seleccionados (Sección 3.2.3.1) como el resultado de la evaluación de la calidad de los trabajos (Sección 3.2.3.2). Después, se discuten las preguntas de investigación planteadas de acuerdo con los resultados obtenidos y las limitaciones del estudio (Sección 3.2.4). Por último, se destacan las principales conclusiones derivadas de los resultados obtenidos (Sección 3.2.5).

3.2.1. Marco de comparación de los metamodelos de requisitos

Gotel y Finkelstein [111] definen la trazabilidad de requisitos (TR) como *la habilidad de describir y seguir la vida de un requisito, tanto hacia delante como hacia atrás*. Ellos distinguen entre: 1) trazabilidad previa a la especificación de requisitos (pre-RS, *pre-Requirements Specification*), que se refiere a aquellos aspectos de la vida de un requisito previa a su inclusión en la especificación de requisitos, y 2) trazabilidad posterior a la especificación de requisitos (post-RS, *post-Requirements Specification*), que se refiere a aquellos aspectos de la vida del requisito que resultan de su inclusión en la especificación de requisitos. Estos autores demostraron empíricamente que la mayoría de los problemas atribuidos a una TR pobre eran debidos a la ausencia de (o una inadecuada) trazabilidad pre-RS. En consecuencia, Gotel y Finkelstein establecen que es necesario dedicar esfuerzos de investigación para proporcionar, a los desarrolladores, técnicas que les permitan registrar y trazar la información relativa a la producción y revisión de la especificación de requisitos. Hasta donde nosotros sabemos, desde que este hecho se puso de relieve, no se ha dedicado mucho esfuerzo a resolver el problema de la trazabilidad pre-RS, hasta ahora. Algunas excepciones incluyen los trabajos de Pohl [273], Gotel y Finkelstein [112], Ravichandar *et al.* [285] y Stone y Sawyer [334]. Desde nuestro punto de vista, un *modelo de trazabilidad de requisitos efectivo* se refiere a un modelo de trazabilidad que cubra tanto la trazabilidad pre como post-RS siguiendo las buenas prácticas de la IR. De acuerdo con las buenas prácticas de IR propuestas por Sommerville y Sawyer [329], los tipos de trazas se pueden clasificar como se muestra en la Tabla 11.

Los dos primeros tipos de relaciones de traza de la taxonomía de Sommerville y Sawyer (Tabla 11), *requisitos-fuente* y *requisitos-razones*, claramente se corresponden con información relacionada con la *trazabilidad pre-RS*, y están en la línea del descubrimiento de Gotel y Finkelstein [111] en cuanto a que la información más útil en relación a la trazabilidad pre-RS es: (a) la fuente del requisito, y (b) las personas

implicadas en las actividades que conducen a su inclusión y refinamiento en la especificación de requisitos. Además, dado que la trazabilidad pre-RS se centra en proporcionar un mejor entendimiento de los requisitos, podría ser útil trazar el requisito a algún material adicional que complemente la especificación del requisito, puesto que, de acuerdo con Robertson y Robertson [293], no toda esta información debería formar parte de la especificación de requisitos, sino que se debe referenciar convenientemente.

Tipo de traza	Descripción
<i>Requisitos-fuentes</i>	Enlaza el requisito con las personas o los documentos a partir de los cuales se obtiene la especificación del mismo.
<i>Requisitos-razones</i>	Enlaza el requisito con una descripción que explique por qué se tiene que especificar.
<i>Requisitos-requisitos</i>	Enlaza el requisito con otros requisitos que, en cierto modo, dependen o están relacionados con él.
<i>Requisitos-arquitectura</i>	Enlaza el requisito con otros artefactos del desarrollo (análisis o diseño) que representan o modelan la especificación de requisitos. Nosotros vamos a agrupar estos tres tipos en una única categoría que vamos a denominar <i>Requisitos-artefacto</i>
<i>Requisitos-diseño</i>	
<i>Requisitos-interfaz</i>	

Tabla 11 Taxonomía de relaciones de traza según Sommerville y Sawyer.

Por otro lado, las relaciones de traza *requisitos-requisitos* y *requisitos-artefacto* (que engloba *requisitos-arquitectura*, *requisitos-diseño* y *requisitos-interfaz*, Tabla 11), tienen que ver con la trazabilidad post-RS. Por un lado, en cuanto a las interdependencias entre requisitos, Dahlstedt y Persson [67] han identificado los tipos de interdependencias fundamentales presentes en la literatura, compilando tipos de interdependencias que se solapan o que son similares, en tipos más genéricos, para mantener el número de interdependencias lo más bajo posible. La taxonomía propuesta por Dahlstedt y Persson, tras su estudio, ha sido adoptado en nuestro marco de comparación para evaluar si los metamodelos propuestos en la literatura contienen las relaciones entre requisitos consideradas como fundamentales. Por otro lado, dado que las relaciones de traza post-RS tratan de asegurar que el sistema satisface todos los requisitos que se han especificado, a lo largo del diseño y la implementación, también tendríamos que tener en cuenta las relaciones de los requisitos con los casos de prueba que deberían utilizarse para asegurarse que el requisito se satisface [67].

El marco de comparación basado en la trazabilidad de requisitos integra todos los aspectos que hemos ido presentando en esta sección. El marco de trazabilidad que proponemos, en cierto modo, ordena los conceptos que deberían definirse en cualquier metamodelo de requisitos que dé soporte a una especificación de requisitos siguiendo las buenas prácticas de IR [329] y las recomendaciones acerca de la trazabilidad [67, 111]. La descripción completa del marco de comparación se muestra en la Tabla 12.

Trazabilidad pre-RS	
<i>Requisito-fuente</i>	Se desglosa en:
<i>Requisito-stakeholder</i>	Enlaza el requisito con el stakeholder que lo propone. El stakeholder se debe caracterizar no sólo por su nombre sino también por el puesto que ocupa en la empresa [111].
<i>Requisito-documento</i>	Enlaza el requisito con el documento legal, manual de la empresa, etc., de donde se ha extraído.
<i>Requisito-razón</i>	Enlaza el requisito con los motivos para su inclusión en la especificación de requisitos.
<i>Requisito-suplemento</i>	Enlaza el requisito con información adicional (por ejemplo, documentos) que complementan su especificación.
Trazabilidad post-RS	
<i>Requisito-Requisito</i>	Enlaza un requisito con otros con los que existe algún tipo de relación. De acuerdo con el estudio de Dahlstedt y Persson [67] las relaciones fundamentales son:
<i>refina</i>	Esta relación se utiliza para describir estructuras jerárquicas (padre-hijo), donde requisitos más detallados se relacionan con sus requisitos fuente. En este sentido, estos requisitos proporcionan una explicación adicional, detallan o clarifican los requisitos fuente. También incluye las dependencias donde un requisito se divide en varias partes (composición, parte_de).
<i>influye</i>	Un requisito influye o depende de otro requisito sin que lo requiera, lo refine o esté en conflicto con él.
<i>requiere</i>	El cumplimiento de un requisito depende del cumplimiento de otro. Este tipo de relación se utiliza para especificar que si un requisito se incluye en la especificación de requisitos de un sistema, eso implica que otro requisito se tiene que incluir también.
<i>conflicto_con</i>	Un requisito está en conflicto con otro requisito si no pueden existir los dos al mismo tiempo, esto es, es imposible implementar los dos requisitos. Existe en este caso una relación de exclusión entre los requisitos. Además, un requisito puede estar en conflicto con otro si incrementando la satisfacción de un requisito, decrementa la satisfacción de otro requisito.
<i>coste/valor</i>	Tiene que ver con el coste implicado en la implementación de un requisito en relación al valor que el cumplimiento de ese requisito reportará al cliente.
<i>Requisito-caso de prueba</i>	Enlaza el requisito con el caso de prueba que se va a utilizar para su validación en el sistema.
<i>Requisito-artefacto</i>	Enlaza un requisito con otro artefacto del desarrollo. De acuerdo con Paige <i>et al.</i> [263] estas trazas pueden ser: <i>explícita</i> , si la traza que liga el requisito con el artefacto se establece explícitamente en el metamodelo de acuerdo con una sintaxis concreta adecuada, o <i>implícita</i> , si los requisitos se transforman automáticamente en otros artefactos del desarrollo.

Tabla 12 Marco de comparación de los metamodelos de requisitos basado en la trazabilidad de requisitos.

3.2.2. Diseño de la revisión sistemática en relación a los metamodelos de requisitos

En esta Sección 3.2.1 se resume el protocolo de la RSL. En la Sección 3.2.2.1 se listan las cuestiones que guían la revisión. En la Sección 3.2.2.2 se realiza la planificación del proceso de búsqueda. En la Sección 3.2.2.3 se definen los criterios de inclusión y exclusión. En la Sección 3.2.2.4 se explican los criterios de calidad que se van a utilizar para evaluar los trabajos seleccionados. En la Sección 3.2.2.5 se muestran los datos recogidos en los estudios seleccionados en la RSL. En la Sección 3.2.2.6 se presenta un análisis de los datos recogidos. Finalmente, en la Sección 3.2.2.7 se comentan las desviaciones producidas respecto a este protocolo.

3.2.2.1. Cuestiones de investigación

Las preguntas de investigación que se pretenden contestar en la RSL son:

- PI1. ¿Cuáles son los conceptos y relaciones de traza que incluyen los distintos metamodelos de requisitos propuestos en la literatura?
- PI2. ¿Cómo se integran los modelos de requisitos en procesos DSDM, esto es, cuáles son las técnicas del DSDM utilizadas (implementación de los metamodelos y transformaciones de modelos)?

3.2.2.2. Proceso de búsqueda

El proceso de búsqueda ha sido automático, utilizando para ello los motores de búsqueda basados en palabras clave de las bibliotecas electrónicas siguientes:

- IEEE Digital Library (www.computer.org/portal/site/csdl/index.jsp)
- ACM Digital Library (portal.acm.org)
- ScienceDirect (www.sciencedirect.com)
- MetaPress (Kluwer + Springer) (www.metapress.com)
- Wiley InterScience (www.interscience.wiley.com)
- Google Scholar (scholar.google.es)

De acuerdo con Zhang *et al.* [370] IEEE, ACM y ScienceDirect son los motores de búsqueda más utilizados en las RSL relacionadas con la Ingeniería del Software. Además, a partir de las fuentes seleccionadas se tiene acceso a las principales revistas y eventos de las comunidades de requisitos (por ejemplo, las revistas *Requirements Engineering Journal*, *Information and Software Technology* o *Journal of Systems and Software*, y las conferencias y *workshops* RE, REFSQ, AWRE y WER) y del DSDM (por ejemplo la conferencia MODELS). Google Scholar también ha sido incluido, a pesar de que dada su amplia cobertura [59] se repiten prácticamente todos los trabajos previamente seleccionados en otros motores de búsqueda. La razón que nos ha llevado a incluir Google Scholar es encontrar “literatura gris” (por ejemplo, informes técnicos) relacionada con las preguntas de investigación.

Se han utilizado dos cadenas de búsqueda diferentes para tratar de encontrar todas las propuestas que definan metamodelos de requisitos en cualquier contexto del desarrollo (C1) y aquellas que traten de los modelos de requisitos explícitamente en el contexto del DSDM (C2). De este modo, las cadenas de búsqueda utilizadas en esta RSL son:

C1) “requirements metamodel” OR “requirements meta-model” OR “metamodel for requirements” OR “meta-model for requirements” OR “metamodel of requirements” OR “meta-model of requirements”

C2) “model-based requirements” OR “model-driven requirements” OR “model-driven approach for requirements” OR “model-based approach for requirements”

Para cada uno de los trabajos obtenidos como resultado de las búsquedas, era suficiente con la lectura del título y el resumen para seleccionarlo, o no, como trabajo candidato, si tenían relación con las preguntas de investigación. Posteriormente, para cada uno de los trabajos candidatos, fue necesaria una lectura más cuidadosa de la introducción y las conclusiones para poder aplicar los criterios de inclusión y exclusión (Sección 3.2.2.3) y seleccionarlos como trabajos relevantes.

3.2.2.3. Criterios de inclusión y exclusión

Para seleccionar un trabajo candidato definitivamente, para su estudio detallado en la RSL, tiene que describir un metamodelo que permita representar la especificación textual de requisitos. El artículo debe proporcionar una representación gráfica de dicho metamodelo, utilizando algún formalismo, de manera que queden explícitos los conceptos y relaciones que incluye. En consecuencia, el metamodelo contendrá el concepto “Requisito” (con este o cualquier otro nombre) que se caracterice por una descripción textual.

Por tanto, quedan excluidos del estudio:

- El metamodelo de requisitos aparece en la propuesta como un componente tangencial y no con el objetivo de dar soporte a la especificación de requisitos. Por ejemplo, un artículo en el que se presenta un mecanismo para detectar inconsistencias entre modelos y proponen un metamodelo de requisitos como ejemplo, pero no pretendiendo contribuir al dominio de la gestión de requisitos.
- Propuestas en las que los requisitos textuales no estén representados en un modelo conforme a un metamodelo.
- Los trabajos que describan textualmente el metamodelo pero no proporcionan una representación gráfica del mismo o la que se de sea incompleta.
- Trabajos que solamente traten la especificación de requisitos de un problema muy concreto (por ejemplo, requisitos para un problema de computación científica, o los requisitos para la definición de una transformación de modelos) y no la especificación de una aplicación de desarrollo de software.
- Trabajos que propongan una extensión a otros modelos generales de IR que dan soporte a la especificación de requisitos. Por ejemplo, una extensión para dar soporte a los requisitos de seguridad definidos en otro metamodelo. En este caso se incluirá en la RSL el metamodelo general.

- Trabajos duplicados del mismo estudio. En el caso de que existan versiones diferentes de un mismo enfoque se toma la más completa o bien la que incluya el metamodelo de requisitos, si en la versión completa no se detalla.
- Trabajos de menos de cuatro páginas.

3.2.2.4. Criterios de calidad

Los criterios de calidad que se han establecido para evaluar los trabajos seleccionados están inspirados en el trabajo de Fernández *et al.* [88] e incluyen:

- CC1) **Solución al problema de trazabilidad pre-RS.** Dado que Gotel y Finkelstein [111] destacan como fundamental abordar el problema de la trazabilidad pre-RS, aquellos trabajos que definan en sus metamodelos todas las relaciones de traza pre-RS establecidas en el marco de comparación, un total de cuatro (Sección 3.2.1), serán valorados con un 2. Los trabajos que incluya sólo parte de las relaciones de traza serán valorados con un 1, y los trabajos que no tengan en cuenta las relaciones pre-RS serán valorados con un -1.
- CC2) **Riqueza del modelo de traza post-RS.** De acuerdo con el marco de comparación de los metamodelos (Sección 3.2.1), hemos definido tres tipos de traza post-RS. Los trabajos que defina relaciones de trazas dentro de las tres categorías serán valorados con un 2, aquellos trabajos que sólo tengan en cuenta dos de las tres categorías serán valorados con un 1, las propuestas que sólo tengan en cuenta las relaciones entre requisitos serán valorados con un 0, y, finalmente, se valorará con un -1 los trabajos que no tengan en cuenta la trazabilidad post-RS.
- CC3) **Validación de la propuesta.** El trabajo propuesto puede haber sido validado 1) utilizando algún método empírico en un entorno industrial, mediante un caso de estudio real, o bien, 2) utilizando un caso de estudio académico para ilustrar la propuesta. En el primer caso, la propuesta será valorada con un 1, mientras que en segundo caso será valorada con un 0. En el caso de que la propuesta no haya sido validada será valorada con un -1.
- CC4) **Lugar de publicación.** Tendremos en cuenta la calidad de la conferencia o revista donde se ha publicado el trabajo. Para ello haremos uso del ranking de conferencias de ciencias de la computación (ERA, *Excellence in Research for Australia*, antiguo CORE) y el listado JCR (*Journal Citation Reports*), respectivamente. Así, los valores asignados a cada trabajo en función de su relevancia son los definidos en la Tabla 13.

	Conferencia (ERA)	Revista (JCR)	Valor
Muy relevante	---	Primer cuartil (Q1)	2
Relevante	A	Q2	1
No muy relevante	B	Q3, Q4	0
No relevante	C, no listada	No indexada	-1

Tabla 13 Criterio de calidad para el lugar de publicación.

- CC5) **Citado por otros.** Consideramos que se debe tener en cuenta como indicador de la calidad de un trabajo las veces que ha sido referenciado por otros autores. No obstante, no se debe penalizar a los trabajos más recientes, publicados en el

mismo año en el que está siendo redactado este documento, puesto que no han tenido las mismas oportunidades que trabajos anteriores. Así, los trabajos de 2012 no referenciados por nadie se valorará con un 0, los trabajos anteriores a esta fecha sin citas serán valorados con un -1. En el caso de que el trabajo tenga de 1 a 5 citas será valorado con un 1, y en el caso de que tenga más de 5 citas, será valorado con un 2. Para conocer las veces que un trabajo ha sido citado utilizaremos la cuenta que nos proporciona la base de datos SCOPUS y Google Scholar. La diferencia fundamental entre estas dos fuentes es que en la base de datos SCOPUS podemos eliminar las auto-referencias (como así hemos hecho), mientras que con Google Scholar no. En el caso de discrepancia entre los valores reportados por ambas fuentes se ha considerado el valor medio.

Aplicando estos criterios de calidad a los trabajos obtenidos en el proceso de búsqueda y selección, la valoración final de cada uno variará entre -5 y 11.

3.2.2.5. Recogida de datos

Para cada uno de los trabajos seleccionados se extrajeron los datos necesarios para poder comparar los metamodelos de acuerdo al marco de comparación establecido (Sección 3.2.1), y poder responder a las preguntas de investigación planteadas (Sección 3.2.2.1). También se recogió la información requerida para poder evaluar la calidad del estudio de acuerdo a los criterios de calidad establecidos en el protocolo (Sección 3.2.2.4). Estos datos incluyen:

- Origen del estudio: fuente a partir de la cual se ha obtenido el estudio
- Referencia bibliográfica completa
- Relaciones de traza pre y post-RS soportadas por los metamodelos
- Objetivo del trabajo, enfoque de IR que soporta y dominio de aplicación.
- Cuestiones relacionadas con las técnicas del DSDM (representación del metamodelo, integración en un proceso de DSDM y soporte a las transformaciones de modelos a partir de los requisitos).
- Aspectos relacionados con la evaluación de la calidad que no han sido incluidos en los puntos anteriores: método de validación utilizado, número de veces referenciado por otros autores.

El proceso de recogida de datos fue llevado a cabo por esta doctoranda. Para garantizar la consistencia de los datos extraídos, de acuerdo con las guías de Kitchenham [164] para los estudiantes de doctorando, se llevó a cabo un proceso test-retest de manera que se extrajeron los datos por segunda vez de un conjunto aleatorio de trabajos.

3.2.2.6. Análisis de datos

Los datos recogidos han sido tabulados para mostrar:

- El contenido de los metamodelos de acuerdo al marco de comparación basado en la trazabilidad de requisitos (en relación a la pregunta de investigación PI1, Sección 3.2.2.1). Esta información se resume en la Tabla 20.
- El identificador asignado al estudio, fuente, autor/es, referencia y año de publicación.

- El motivo por el que los autores proponen la definición de un metamodelo de requisitos.
- Enfoque de IR: tradicional, orientado a objetivos (GORE), orientado a aspectos (AORE) o un enfoque propio definido *ad-hoc* para la propuesta que se presenta.
- Aspectos relacionados con las técnicas del DSDM:
 - o Representación del metamodelo: modelo de clases, modelo MOF, perfil UML.
 - o Integración en el DSDM: ninguna, si la propuesta se centra únicamente en la etapa de IR y el trabajo no se encuadra en el DSDM; parcial, si proponen el soporte a la etapa de IR como primer paso para la integración de los requisitos en el DSDM; y total, si la etapa de IR se enmarca en un proceso completo de DSDM.
 - o Soporte a las transformaciones de modelos desde los requisitos a otros modelos del desarrollo, en cuyo caso se estudia el tipo de la transformación, el lenguaje utilizado y los modelos fuente y destino.
- Herramienta que dé soporte a la propuesta.
- El método de validación y el dominio de aplicación.

3.2.2.7. Desviaciones del protocolo

La sección de trabajos relacionados de cada uno de los artículos incluidos en la RSL ha sido revisada para detectar propuestas que no hubieran sido encontradas por los motores de búsqueda. Así, se analizaron siete trabajos (ver Tabla 14), de los cuales sólo uno se seleccionó finalmente. Es el caso del trabajo de Navarro *et al.* [243], referenciado desde el trabajo de los mismos autores [242] encontrado en la base de datos Metapress, donde se detalla el metamodelo de requisitos que sirve como base para la implementación de la herramienta MORPHEUS [242]. La inclusión de este trabajo supone una desviación respecto al protocolo definido inicialmente.

También hemos estudiado el modelo de requisitos del lenguaje estándar del OMG SysML [257], puesto que es el modelo base de otras propuestas [77, 145, 199] y era necesario conocer su modelo de traza para poder inferir el de los trabajos incluidos en la revisión.

3.2.3. Resultados

En esta sección se resumen los resultados del proceso de búsqueda incluyendo tanto los estudios candidatos como los que finalmente han sido seleccionados y analizados (Sección 3.2.3.1). Además, también se evalúa la calidad de los trabajos seleccionados (Sección 3.2.3.2) de acuerdo a los criterios de calidad establecidos en la Sección 3.2.2.4.

3.2.3.1. Resultados de las búsquedas

El número de trabajos obtenido como resultado de la búsqueda de cada una de las cadenas en las bases de datos electrónicas (Sección 3.2.2.2) se puede ver en la Tabla 14. La cadena C1 tenía que ver con la búsqueda de metamodelos de requisitos y la cadena C2 con enfoques de requisitos dirigidos o basados en modelos. En esta tabla se muestra

como de los 91 trabajos relevantes, finalmente 22 fueron seleccionados (aplicando los criterios de inclusión y exclusión) para un estudio en profundidad. Estos trabajos serán presentados con más detalle en la Sección 3.2.4. Siguiendo las recomendaciones de Kitchenham [164] los trabajos seleccionados y rechazados fueron discutidos con el director de esta tesis doctoral para favorecer la fiabilidad de las decisiones tomadas por esta doctoranda. Además, los trabajos han sido re-evaluados tras su inclusión en la RSL durante la elaboración de este documento lo que garantiza la consistencia de las decisiones acordadas.

	C1	C2	Relevantes	Seleccionados
ACM	20	22	13	1
IEEE	221	70	18	6
MetaPress (MP)	64	45	20	4
ScienceDirect (SD)	10	11	6	1
Wiley	4	3	2	1
Google Scholar (GS)	333	287	25	8
Desviaciones			7	1
TOTAL	652	438	91	22

Tabla 14 Resultados de las búsquedas de metamodelos de requisitos.

Trabajos excluidos de la RSL

En el proceso de búsqueda de trabajos, todo aquél que incluyera el término “metamodelo de requisitos”, o una figura con este título, era seleccionado como trabajo candidato, de ahí que el número de trabajos analizados para determinar su inclusión, o no, en la RSL haya sido tan elevado (un total de 91, Tabla 14). Dado que el listado de los trabajos potencialmente relevantes, que finalmente fueron rechazados, es amplio, esta información se encuentra como anexo (ANEXO I). En este listado también se incluye el motivo de exclusión, de acuerdo a los criterios establecidos (Sección 3.2.2.3).

Entre los trabajos excluidos, cabe destacar dos líneas de investigación interesantes: 1) las propuestas que se centran en la definición de un metamodelo para la especificación de las sentencias de requisitos, como medio para formalizar y restringir la forma en la que se escriben los requisitos [39, 40, 74, 149, 321, 351], y 2) las propuestas definidas en el marco del DSDM, que parten de modelos de requisitos, como primer modelo para el desarrollo de un aplicación, pero estos modelos son conformes a metamodelos que no incluyen la especificación de requisitos textuales, sino que representan otros artefactos, no ligados a los requisitos expresados en lenguaje natural, como son: casos de uso [37], diagramas de actividades [119], modelos de requisitos basados en tareas [346], modelos de requisitos para el web [81] o modelos i^* [15]. A continuación, vamos a presentar brevemente los trabajos relacionados con la primera línea de investigación puesto que tratan, de algún modo, con especificaciones de requisitos textuales, que es el formato en el que estamos interesados en esta tesis doctoral.

El enfoque ProjectIT [321] integra un conjunto de lenguajes y herramientas para dar soporte a todas las actividades del desarrollo del software para sistemas interactivos en el marco del DSDM. Entre estos lenguajes se incluye *ProjectIT-RSL* [351], un lenguaje natural controlado para la especificación de requisitos. El lenguaje se ha definido después de analizar los patrones lingüísticos más comunes que aparecen en los

documentos de especificación de requisitos, y es soportado por un editor que proporciona la visión del modelo como si fuera un procesador de texto. El metamodelo que define el lenguaje ProjectIT-RSL, especifica conceptos tales como Entidad, Actor, Operacion, y relaciones como, “actor *ejecuta* una operación sobre una entidad”. El uso de este lenguaje restringido, favorece la definición de transformaciones modelo a modelos de las especificaciones de requisitos a otros modelos, estableciendo las siguientes correspondencias: (1) las entidades se transforman en clases UML; (2) las propiedades de las entidades son los atributos de las clases y (3) los actores se transforman en actores UML. Estos elementos, generados automáticamente, se pueden utilizar posteriormente en la fase de diseño.

Brottier *et al.* [39] proponen un mecanismo, dirigido por modelos, para manipular un conjunto de requisitos textuales parciales, producidos por distintos stakeholders, para generar un modelo de requisitos global. Este mecanismo se integra en la *plataforma R2A (Requirements to Analysis)*, implementada utilizando la tecnología DSDM. La pieza clave de esta plataforma es el metamodelo de requisitos que se ha definido para poder representar el modelo de requisitos global (metamodelo *R2A*). Este mismo metamodelo se utiliza en el trabajo que estos mismos autores presentan posteriormente [40].

Para la especificación textual de requisitos se utiliza lenguaje natural restringido denominado RDL (*Requirements Description Lenguaje*), para el que se ha definido un metamodelo semánticamente equivalente, denominado RDM (*Requirements Description Metamodel*). El proceso para generar el modelo de requisitos global, a partir de la especificación textual de requisitos escrita utilizando el lenguaje RDL, consta de tres pasos: 1) analizar sintácticamente cada especificación de requisitos textual para producir un modelo de sintaxis abstracta, conforme al metamodelo RDM; 2) transformar cada modelo de sintaxis abstracta en un modelo conforme al metamodelo R2A; y 3) unir estos modelos intermedios en un modelo único (también conforme al metamodelo R2A). La generación de modelos R2A a partir de modelos RDM supone la generación tanto de la parte de análisis (entidades) como funcional (casos de uso) del modelo R2A.

La idea de fondo de las propuestas anteriores es definir un metamodelo que represente un lenguaje natural restringido para la especificación de requisitos, de manera que sea factible definir transformaciones de modelos a partir de las especificaciones de requisitos textuales. En cambio, el objetivo del trabajo de Denger *et al.* [74] es mejorar la calidad de las especificaciones de requisitos en lenguaje natural para el dominio de los sistemas embebidos. Para ello, proponen utilizar patrones de lenguaje natural que guíen al analista en la formulación de las sentencias de requisitos, que serán menos ambiguas, más completas y más precisas. Los patrones, que describen todas las posibles sentencias de requisitos de los sistemas embebidos, se derivan a partir de un metamodelo, que representa los elementos que componen estas sentencias. Este metamodelo se ha desarrollado analizando la literatura acerca de los sistemas embebidos, y las especificaciones de requisitos existentes para este tipo de sistemas, con el objeto de determinar los tipos de comportamientos que forman parte de los sistemas embebidos y los elementos del lenguaje que se necesitan para especificar cada uno de estos comportamientos. Estos elementos son los que se representan en el metamodelo.

Así, el metamodelo incluye conceptos (metaclases) tales como, Comportamiento Discreto, Evento, Reaccion, entre otros.

En esta misma línea de definir un metamodelo para mejorar la calidad de las especificaciones de requisitos, se enmarca el trabajo de Jungbai *et al.* [149], en el que proponen un metamodelo para requisitos no funcionales basado en ontologías, lo que permite inducir los requisitos no funcionales, en dominios específicos, a partir del conocimiento y las reglas definidos en la ontología. De esta forma, la ontología del dominio guía la adquisición y el diseño de los requisitos no funcionales. La propuesta se enmarca en el dominio de las aplicaciones de red (*Networked Software*).

Trabajos incluidos en la RSL

Los trabajos que finalmente fueron incluidos en la RSL aplicando los criterios de inclusión y exclusión definidos en el protocolo (Sección 3.2.2.3) son los que se muestran en la Tabla 15, Tabla 16 y Tabla 17. En la columna *Integración MDSD* los símbolos utilizados significan: □ no se encuadra la propuesta en el DSDM, ◻ propuesta basada en modelos, ▣ se encuadra en el DSDM pero sólo trata la etapa de IR, y ■ propone un proceso completo DSDM. Un enfoque de IR *basado en modelos*, al igual que la tecnología *dirigida por modelos*, se apoya en la definición de un metamodelo para especificar los elementos del proceso de IR, pero, a diferencia con la tecnología dirigida por modelos, no integra la etapa de IR en un proceso que siga la filosofía del DSDM, en el que se van generando los modelos de las distintas etapas mediante transformaciones de modelos. En las tablas aparecen en blanco las casillas para las que no se encontraba explícitamente la información asociada en el artículo referenciado.

El primer resultado interesante que se desprende del estudio es que las primeras propuestas, que hemos encontrado, que presentan un metamodelo de requisitos en el marco del DSDM son del año 2006 [243, 355]. Debemos tener en cuenta que la primera versión del estándar del OMG para el DSDM, MDA, se publicó en el año 2001, y que la primera versión del lenguaje SysML, que extiende UML con nuevos modelos, entre los que incluye el modelo de requisitos, fue publicada en el año 2006. Cercano a estas fechas es cuando se presenta la primera versión del metamodelo REMM [234, 348], pieza clave de esta tesis doctoral, y publicamos nuestra hipótesis de trabajo en relación a las ventajas de integrar la IR con el DSDM [233]. Así, nuestra línea de investigación comienza justo cuando empieza a cobrar interés la idea de modelar los requisitos para poder trazarlos e integrarlos con el resto de artefactos del desarrollo en el marco de un proceso DSDM. Los trabajos anteriores al 2006 [52, 189, 224, 281] se centran en: 1) el problema de la trazabilidad de los requisitos con el resto de elementos implicados en el desarrollo, tanto pre-RS como post-RS, como son los trabajos de Ramesh y Jarke [281] y Letelier [189], ambos muy referenciados (ver Tabla 18); y 2) el modelado de la variabilidad en la especificación de requisitos para familias de sistemas o Líneas de Producto Software (SPL, *Software Product Line*) [52, 224].

Fuente	Autor [ref.]	Año	Propósito del metamodelo	Enfoque IR	Representación del metamodelo	Integración MDS	Herramienta	Dominio de aplicación	Validación	Id.
MP	Cerón <i>et al.</i> [49]	2005	Modelado de requisitos	Nuevo proceso basado en CMMi		☐	ENAGER Eclipse-based	Familias de Sistemas	Caso de estudio académico (sistemas domóticos y automoción)	Cerón05
IEEE	Dubois <i>et al.</i> [74]	2010	Trazabilidad		Perfil UML que extiende SysML	☐	Papyrus-based	Sistemas Embebidos	Caso de estudio académico (Aplicación automoción)	Dubois10
GS	Geisberger <i>et al.</i> [99]	2007	Modelado de requisitos, fácil transición al diseño	GORE		☐	AutoRAID, AutoFocus	Sistemas Embebidos	Caso de estudio Industrial	Geisberger07
MP	Goknil <i>et al.</i> [104]	2008	Modelo de requisitos base extensible		Ontología (OWL)	☒	OWL-based	PG	Caso de estudio Industrial (Servicio de móvil)	Goknil08
GS	Ji <i>et al.</i> [141]	2011	Especificación de requisitos para aplicaciones industriales		Perfil UML que extiende SysML	☒	Si	Sistemas de automatización industrial	Caso de estudio académico (Sistema de presión hidroestática)	Ji11
IEEE	Kaindl <i>et al.</i> [150]	2009	Modelado de objetivos y sus relaciones	GORE	MOF	☐	Eclipse-based	PG	Caso de estudio académico (Club de fitness)	Kaindl09

Tabla 15 Trabajos incluidos en la RSL de metamodelos de requisitos (1/3). Leyenda: GORE = Goal Oriented Requirement Engineering; PG = Propósito General.

Fuente	Autor [ref.]	Año	Propósito del metamodelo	Enfoque IR	Representación del metamodelo	Integración MDSD	Herramienta	Domnio de aplicación	Validación	Id.
IEEE	Kassab <i>et al.</i> [157]	2009	Trazabilidad entre requisitos funcionales y no funcionales			☐		PG	Caso de estudio académico (Sistema de <i>e-mail</i>)	Kassab09
GS	Letelier [185]	2002	Trazabilidad		Perfil UML	☐		PG	Proyecto con el Centro de Operaciones de la Agencia Espacial Europea	Letelier02
GS	Mannion y Kaindl [195]	2007	Modelado de la variabilidad para la reutilización de requisitos	MRAM [196]	Perfil UML que extiende SysML	☐		SPL		Mannion07
IEEE	Marschall y Schoenmakers [201]	2003	Enlazar documentos de requisitos con el modelo conceptual			☒	Si	Sistemas web B2B	Caso de estudio académico (sistema de pedidos B2B)	Marschall03
ACM	Merschen <i>et al.</i> [211]	2011	Importar requisitos de DOORS y trazarlos		MOF	☒	Eclipse-based	SPL automoción embebido	Caso de estudio en Daimler AG	Merschen11
Wiley	Molina <i>et al.</i> [215]	2010	Especificación de requisitos medibles	GORE	Perfil UML	☒	Eclipse-based	PG	Caso de estudio académico (sistema de venta de entradas online)	Molina10
MP	Monperrus <i>et al.</i> [220]	2011	Métricas sobre especificación de requisitos		MOF	☒	Eclipse-based	PG		Monperrus11
GS	Monzón y Dueñas [221]	2004	Reutilización de requisitos			☐	DOORS	SPL Aircraft	Aplicación Industrial	Monzón04

Tabla 16 Trabajos incluidos en la RSL de metamodelos de requisitos (2/3).

Leyenda: MRAM = *Method for Requirements Authoring Management*, SPL = *Software Product Line*.

Fuente	Autor [ref.]	Año	Propósito del metamodelo	Enfoque IR	Representación del metamodelo	Integración MSD	Herramienta	Dominio de aplicación	Validación	Id.
IEEE	Moon <i>et al.</i> [222]	2007	Trazar la variabilidad entre requisitos y arquitecturas	Propuesto en [223]		□		SPL		Moon07
GS	Vicente-Chicote <i>et al.</i> [346]	2007 2008	Modelo de requisitos <i>para y con</i> reutilización	SIREN [340]	MOF	▣	REM-Studio Eclipse-based	PG	Caso de estudio académico (sistema de iluminación automático)	Vicente-Chicote07
	Navarro <i>et al.</i> [239, 240]	2006 2009	Integración de técnicas de IR	ATRIUM [238]		■	MORPHEUS [239]	PG	Caso de estudio EFTCoR	Navarro06
IEEE	Ramesh y Jarke [278]	2001	Trazabilidad			□	ConceptBase herramientas comerciales de trazabilidad	PG	Casos de estudio industriales	Ramesh01
SD	Sánchez <i>et al.</i> [298]	2011	Trazar requisitos reutilizables con diseño		MOF	■	Eclipse editor	Sistemas domóticos	Encuesta	Sánchez11
GS	Schwarz <i>et al.</i> [305]	2008	Trazabilidad			□		PG		Schwarz08
GS	Vogel y Mantell [352]	2006	Integración de requisitos en MDA		Perfil UML	■	Mantis, RSA	PG	Experimentos Industriales	Vogel06
MP	Zikra <i>et al.</i> [372]	2011	Integrar modelos de requisitos con diseño	GORE		▣		Modelado de Empresa	Caso de estudio académico	Zikra11

Tabla 17 Trabajos incluidos en la RSL de metamodelos de requisitos (3/3). Leyenda: RSA = *Rational Software Architecture*.

Si nos fijamos en el propósito para la construcción de los metamodelos de requisitos en los diferentes trabajos, destacan dos aspectos fundamentalmente: 1) establecer un modelo de referencia para los productos de IR [52, 102, 107, 145, 154, 243, 348, 355, 375], o bien, 2) el problema de la trazabilidad de los requisitos con otros artefactos, como arquitecturas o diseños [77, 161, 189, 204, 214, 281, 301, 308]. Dentro de los trabajos que se encuadran en el primer grupo, Goknil *et al.* [107] proponen lo que, a su juicio, podría constituir un modelo base (*core metamodel*), que luego podría extenderse para dar soporte a distintos enfoques de IR. En cierto modo, estos dos aspectos están solapados puesto que definir un metamodelo implica la definición de conceptos y las relaciones entre ellos, esto es, las relaciones de traza. Sin embargo, hay propuestas que se centran en definir los elementos implicados en la especificación de requisitos, de manera que los modelos de requisitos sirvan para representar la especificación textual de requisitos, mientras que en otros trabajos el énfasis se hace en utilizar los modelos de requisitos para poder trazarlos. Por ejemplo, en el trabajo de Merschen *et al.* [214], los modelos de requisitos se utilizan para importar las especificaciones de requisitos definidas en DOORS y trazarlas con modelos correspondientes del programa de simulación que se utiliza para el diseño de los sistemas embebidos (Simulink [11]).

Encontramos además, otros dos aspectos minoritarios como propósito de la definición de los metamodelos de requisitos, como son: el modelado de la variabilidad, para dar soporte a la reutilización de requisitos [199, 224, 234], y la idea de medición de los requisitos [218, 223]. En este último caso, las aproximaciones de los trabajos son diferentes, por un lado Molina *et al.* [218] integra en el metamodelo el concepto de requisitos y las medidas que sirven para evaluar ese requisito, en este sentido se dice que se propone un metamodelo de requisitos “medibles”. Por otro lado, Monperrus *et al.* [223] han compilado un conjunto amplio de métricas que permiten medir, automáticamente, la calidad de la especificación de requisitos, representada en un modelo de requisitos. Por ejemplo, número de requisitos totales o número de requisitos asignados a un componente software.

Hemos de señalar que, si bien hemos intentado clasificar los metamodelos atendiendo a su propósito, las categorías utilizadas en la clasificación no son disjuntas. Por ejemplo, Moon *et al.* [225] modelan la variabilidad de una SPL a nivel de requisitos y arquitecturas y utilizan los metamodelos para poder trazar la variabilidad entre ambos niveles. Por otro lado, en el trabajo de Molina *et al.* [218], proponen un metamodelo de requisitos medibles, estableciendo previamente los conceptos, que a juicio de los autores, debían formar parte de un metamodelo para IR básico (*core*, al estilo de la propuesta de Goknil *et al.* [107]). Cabe destacar, que ambos trabajos, el de Molina *et al.* y el de Goknil *et al.*, referencian a REMM, el metamodelo de requisitos que proponemos en esta tesis doctoral (Capítulo 5), a la hora de establecer los elementos básicos de un metamodelo de requisitos, y comparten muchos de los conceptos que se propusieron inicialmente en REMM.

La mayoría de las propuestas no establecen explícitamente el enfoque de IR que se debe seguir para abordar la etapa de IR, y obtener la especificación de requisitos, que se representa en los modelos. Entendemos que, en estos casos, los autores deben considerar que sea cual sea el enfoque utilizado, el resultado del proceso de IR, que es la especificación de requisitos, se puede representar en los modelos que ellos proponen. En los casos en los que si se guía el proceso a seguir en la etapa de IR, se reparten más o

menos por igual, los trabajos que recomiendan seguir un enfoque orientado a objetivos (GORE) [102, 154, 218, 375] y los que definen un proceso *ad-hoc* para esta etapa [52, 199, 225, 243, 348]. Bajo nuestro punto de vista, una propuesta de IR debe cubrir no sólo la tecnología utilizada para dar soporte a la especificación de requisitos, sino también de un proceso que guíe esta etapa crucial en el desarrollo del software.

En casi la mitad de los trabajos (45,5%) no se especifica la tecnología utilizada para la representación o implementación de los metamodelos. En estos casos, los metamodelos se muestran gráficamente al estilo de los diagramas de clases de UML. En algunas ocasiones, la implementación del metamodelo se puede inferir de la herramienta de soporte utilizada, puesto que si se utiliza Eclipse, los metamodelos se habrán especificado utilizando el lenguaje MOF. Así, cinco de las propuestas utilizan este lenguaje para la implementación del metamodelo (23%), y seis son las propuestas (27%) que especifican el metamodelo como un perfil de UML, en concreto, en tres de ellas, el metamodelo de requisitos extiende el perfil de SysML definido para representar los modelos de requisitos [77, 145, 199]. Solamente en la propuesta de Goknil *et al.* [107] se utiliza el lenguaje para definir ontologías OWL [31] para la especificación del metamodelo de requisitos (4,5%).

Analizando la integración de los metamodelos de requisitos en el contexto del DSDM, vemos que más de la mitad de los trabajos estudiados (54,5%) no enmarcan sus propuestas en este tipo de desarrollos, sino que utilizan este formalismo (el metamodelo) para representar los conceptos implicados en el proceso de IR y las relaciones de traza. Entre estas propuestas se encuentra las presentadas por Dubois *et al.* [77], Geisberger *et al.* [102] y Marschall y Schoenmakers [204], que siguen un enfoque de IR *basada en modelos*. Por otro lado, el 32% de los trabajos proponen soluciones para el modelado de los requisitos dentro del marco del DSDM. Este grupo de trabajos los hemos clasificado como que tienen una integración parcial, en el sentido de que la aproximación que presentan se centra sólo en la etapa de IR o, como mucho en la relación de la etapa de IR con el siguiente modelo del desarrollo. Sólo tres de los trabajos (13,5%) proponen un proceso completo de DSDM (desde los requisitos hasta el código) [241, 301, 355], en todos ellos los modelos de requisitos se utilizan para mantener las relaciones de traza entre la especificación de requisitos y el resto de artefactos del desarrollo, pero no para transformar los modelos de requisitos en los modelos del siguiente nivel, como se propone en la filosofía de DSDM. En esta línea es donde queremos contribuir con el trabajo de esta tesis doctoral, en la definición de un proceso completo de DSDM en el que se puedan obtener los modelos de diseño de la aplicación (conformes a un DSL) a partir de los modelos de requisitos.

En relación a las herramientas de soporte, el 27% de las propuestas no hacen referencia, ni muestran, ninguna herramienta que permita automatizar las aproximaciones que presentan. Del resto, cabe destacar que el 44% de las herramientas utilizadas se han desarrollado en el marco del entorno de desarrollo Eclipse [52, 154, 214, 218, 223, 301, 348].

La mitad de las propuestas se definen de manera general para cualquier dominio de aplicación, mientras que otros trabajos proponen soluciones para dominios particulares tales como: sistemas embebidos [77, 102, 214], sistemas de automatización industrial [145], sistemas B2B [204], sistemas domóticos [301], modelado de empresa (*Enterprise Modeling*) [375] y líneas de producto software [52, 199, 224, 225].

Por ultimo, son pocas las propuestas que han sido validadas en entornos reales o industriales (36%). Cabe destacar el trabajo de Ramesh y Jarke [281], en el que abordaron estudios empíricos en un gran número de organizaciones (un total de 26, incluyendo defensa, gubernamentales, aeroespaciales, desarrollo de hardware, farmacéuticas, utilidades, integración de sistemas, electrónica y telecomunicaciones), para elaborar un modelo de referencia que recoja las necesidades de información (en relación a la trazabilidad) de los diferentes stakeholders implicados en el proceso de desarrollo de software. También destaca el trabajo de Mannion y Kaindl [199] validado en el contexto de un proyecto con el Centro de Operaciones de la Agencia Espacial Europea, o la propuesta de Merschen *et al.* [214] desarrollado conjuntamente con ingenieros de Daimler AG.

3.2.3.2. Evaluación de la calidad

Los datos correspondientes a los criterios de calidad, establecidos en la Sección 3.2.2.4, son los que se muestran en la Tabla 18. En esta tabla se indica con el símbolo # el hecho de que el trabajo no se encuentre en la base de datos o el listado donde se ha buscado. En la columna de “Validación” los acrónimos significan: CEA, caso de estudio académico; CER, caso de estudio real o industrial; Survey, encuesta y “x” no validado.

A partir de los datos recogidos en en la Tabla 18, y aplicando las puntuaciones establecida para cada uno de los criterios de calidad, la valoración final de cada trabajo es la que se muestra en la Tabla 19. En esta tabla podemos ver que el trabajo que alcanza la máxima puntuación es el clásico de Ramesh y Jarke [281] sobre un modelo de referencia para la trazabilidad de requisitos, que proporciona un marco extenso para la definición de relaciones de traza basado en estudios empíricos con empresas.

Al trabajo de Ramesh y Jarke le siguen en valoración las propuestas de Cerón *et al.* [52] y Vicente-Chicote *et al.* [348]. Estos trabajos, a pesar de no estar validados en casos de estudios reales, proporcionan un modelo de trazabilidad amplio, cercano al establecido en el marco de comparación.

Algunos de los trabajos han obtenido una valoración negativa [154, 223, 225, 375], estos trabajos tienen en común, fundamentalmente, que en sus modelos de trazabilidad de requisitos solamente tienen en cuenta las relaciones de traza entre requisitos, ignorando las relaciones de traza pre-RS y las relaciones de traza entre los requisitos y otros artefactos.

3.2.4. Discusión

Después de haber recogido los datos y de haber analizado los trabajos seleccionados, a continuación se responde a las cuestiones de la investigación presentadas en la Sección 3.2.2.1. En primer lugar, en la Sección 3.2.4.1 compararemos los metamodelos propuestos en los trabajos seleccionados, para establecer si cubren los aspectos relacionados con la trazabilidad pre-RS y post-RS que hemos considerado fundamentales. A continuación, en la Sección 3.2.4.2 discutiremos las características de las propuestas en cuanto a su integración en el DSDM.

	Pre-RS (4)	Post-RS (3)	Validación	Publicación	Nivel	Citas-SCOPUS	Citas-GS
Cerón05	3	3	CEA	<i>Conference on Product Focused Software Process Improvement</i>	B	3	10
Dubois10	0	3	CEA	<i>IEEE International Conference on Engineering of Complex Computer Systems</i>	A	3	5
Geisberger07	0	2	CER	<i>Dagstuhl Seminar 06351 "Methods for Modelling Software Systems (MMOSS)"</i>	#	#	1
Goknil08	3	2	CER	<i>European Conference on Model Driven Architecture</i>	#	8	26
Ji11	0	3	CEA	<i>International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools</i>	#	#	0
Kaindl09	0	1	CEA	<i>IEEE International Conference on Software Engineering Advances</i>	C	1	1
Kassab09	1	2	CEA	<i>2009 WRI World Congress on Computer Science and Information Engineering</i>	#	1	2
Letelier02	2	3	x	<i>International Workshop on Traceability in Emerging Forms of Software Engineering</i>	#	#	84
Mannion07	0	3	CER	SYSTEMS ENGINEERING	Q4	3	13
Marschall03	0	2	CEA	<i>IEEE International Conference and Workshop on the Engineering of Computer-Based Systems</i>	B	#	6
Merschen11	0	3	CER	<i>Workshop on Variability Modeling of Software-Intensive Systems</i>	#	0	5
Molina10	2	2	CEA	INTERNATIONAL JOURNAL OF INTELLIGENT SYSTEMS	Q2	1	1
Monperrus11	0	2	x	SOFTWARE QUALITY JOURNAL	Q4	0	0
Monzón04	3	1	CER	<i>International Workshop on Requirements Reuse in System Family Engineering</i>	#	#	1
Moon07	0	2	x	<i>IEEE International Conference on Computer and Information Technology</i>	C	3	5
Vicente-Chicote07	4	2	CEA	TOOLS Europe - JOT	B	7	28
Navarro06	0	1	CER	JOURNAL OF COMPUTER INFORMATION SYSTEMS	Q3	7	15
Ramesh01	4	3	CER	IEEE TRANSACTIONS ON SOFTWARE ENGINEERING	Q1	297	666
Sánchez11	0	2	Survey	JOURNAL OF SYSTEMS AND SOFTWARE	Q2,Q3	4	4
Schwarz08	0	3	x	<i>Workshop Software Reengineering</i>	#	#	8
Vogel06	1	2	CER	<i>European Conference on Model Driven Architecture</i>	#	#	7
Zikra11	0	1	CEA	<i>Exploring Modelling Methods in Systems Analysis and Design (EMMSAD)</i>	C	0	2

Tabla 18 Criterios de calidad de los trabajos que proponen metamodelos de requisitos.

3.2.4.1. PI1 ¿Cuáles son los conceptos y relaciones de traza que incluyen los distintos metamodelos de requisitos propuestos en la literatura?

La comparación de los metamodelos de requisitos es difícil, porque no existe un modelo de referencia o una ontología de requisitos ampliamente aceptada, con el que poder contrastarlos. Además, cada uno de ellos define conceptos y relaciones de traza diferentes, para los que no se proporciona una semántica formal de dichos términos (salvo en el caso de Goknil *et al.* [107]), y por tanto, la interpretación de si los elementos de los metamodelos son equivalentes, o no, se basa simplemente en si los nombres, que les han dado los autores, parece que son semánticamente equivalentes o no. En consecuencia, hemos fijado un marco de comparación (Sección 3.2.1), que se ha definido

atendiendo a las recomendaciones de estudios acerca de la trazabilidad de requisitos [67, 111] y las buenas prácticas en IR [329].

	CC1	CC2	CC3	CC4	CC5	VALOR
Cerón05	1	2	0	0	2	5
Dubois10	-1	2	0	1	1	3
Geisberger07	-1	1	1	-1	1	1
Goknil08	1	1	1	-1	2	4
Jill	-1	2	0	-1	0	0
<i>Kaindl09</i>	-1	0	0	-1	0	-2
Kassab09	1	1	0	-1	1	2
Letelier02	1	2	-1	-1	2	3
Mannion07	-1	2	1	0	2	4
Marschall03	-1	1	0	0	1	1
Merschen11	-1	2	1	-1	1	2
Molina10	1	1	0	1	1	4
<i>Monperrus11</i>	-1	1	-1	0	0	-1
Monzón04	1	0	1	-1	0	1
<i>Moon07</i>	-1	1	-1	-1	1	-1
Vicente-Chicote07	2	1	0	0	2	5
Navarro06	-1	0	1	0	2	2
Ramesh01	2	2	1	2	2	9
Sánchez11	-1	1	1	1	1	3
Schwarz08	-1	2	-1	-1	1	0
Vogel06	1	1	1	-1	1	3
<i>Zikra11</i>	-1	0	0	-1	1	-1

Tabla 19 Valoración de los trabajos sobre metamodelos de requisitos.

El análisis de cada trabajo, en base al marco de comparación establecido, se muestra en la Tabla 20. A la vista de los datos recogidos en esta tabla, la primera conclusión que podemos extraer es que, a las relaciones de traza pre-RS se les presta menos atención que a las relaciones de traza post-RS. Mientras que todos los metamodelos incluye, al menos, algún tipo de traza entre requisito, el 59% de las propuestas ignoran completamente las relaciones de traza pre-RS. Estas relaciones de traza son las que, a juicio de Gotel y Finkelstein [111], era más importantes. Por tanto, la necesidad de dedicar esfuerzos a las relaciones de traza pre-RS, sigue vigente casi veinte años después.

En todos los metamodelos que tienen en cuenta las relaciones de traza pre-RS [52, 107, 161, 189, 218, 224, 281, 348, 355], se incluye el concepto Stakeholder. La relación entre los requisitos y los stakeholders que los proponen resuelve uno de los problemas más comunes identificados por Gotel y Finkelstein [111], la incapacidad de acceder a la fuente de los requisitos. No obstante, no es posible conocer las propiedades que caracterizan al concepto Stakeholder en todas las propuestas, sólo se hacen explícitas estas propiedades en los trabajos de: Goknil *et al.* [107], que caracterizan al stakeholder por un *nombre* y una *descripción*, Vogel y Mantell [355] que describen al stakeholder por su *interés* y la *información de contacto*, y en REMM [348], metamodelo propuesto en esta tesis doctoral (Capítulo 5), donde se recoge el *nombre* y el *puesto* que desempeña.

Identificador [ref]	Trazabilidad Pre-RS					Trazabilidad Post-RS					
	Requisito-fuente		Requisito-razón	Requisito-suplemento	Requisito-Requisito					Requisito-Caso de prueba	Requisito-artefacto
	Req-stakeholder	Req-documento			requiere	refina	conflicto_con	influye	coste/valor		
Cerón05[49]	✓	✓	Atributo		✓	✓				✓	✓
Dubois10[74]						✓				✓	✓
Geisberger07[99]						✓					✓
Goknil08[104]	✓		Atributo	✓	✓	✓				✓	
J111 [141]							✓	✓		✓	✓
Kaindl09 [150]							✓	✓			
Kassab09 [157]	✓					✓		✓			✓
Letelier02 [185]	✓		✓			✓		✓		✓	✓
Mannion07 [195]							✓	✓		✓	✓
Marschall03 [201]					✓	✓		✓		✓	✓
Merschen11 [211]						✓		✓		✓	✓
Molina10 [215]	✓	✓					✓			✓	
Monperrusi1 [220]							✓	✓		✓	✓
Monzón04 [221]	✓	✓	✓				✓	✓			✓
Moon07 [222]							✓	✓			✓
Navarro06 [240]					✓	✓	✓	✓			
Ramesh01 [278]	✓	✓	✓	✓			✓	✓		✓	✓
Sánchez11 [298]					Type: String						✓
Schwarz08 [305]						✓				✓	✓
Vicente-Chicote07 [346]	✓	✓	Atributo	✓	✓	✓	✓	✓	✓	✓	
Vogel06 [352]	✓										
Zikra11 [372]						✓	✓	✓			

Tabla 20 Comparativa de los metamodelos de requisitos. Con un ✓ se representa la existencia de una relación que liga dos conceptos en el metamodelo. Cuando la información se recoge mediante un atributo en lugar de una traza se indica explícitamente en la casilla.

La necesidad de conocer la información sobre el puesto que ocupa el stakeholder en la organización la pone de relieve Sommerville y Sawyer [303], pues consideran que si el stakeholder que propuso el requisito ya no está en la empresa, la persona que desempeña el mismo puesto puede participar en la discusión acerca del requisito. Esta información podría recogerse en la *descripción* que proponen Goknil *et al.*, pero no está claro si el resto de propuestas, salvo REMM, tienen en cuenta las recomendaciones de las buenas prácticas de IR.

Mientras que todas las propuestas que modelan las relaciones de traza pre-RS coinciden en especificar el stakeholder como fuente del requisito, no en todos los casos tienen en cuenta que un requisito se puede extraer a partir de la información recogida en algún documento, como la legislación en relación a la seguridad de los sistemas de información, políticas o normas de la empresa o del sector. Este aspecto sólo lo tienen en cuenta el 55,5% de los metamodelos que tratan las relaciones pre-RS (que en realidad representan, aproximadamente, el 23% sobre el total de trabajos) [52, 218, 224, 281, 348].

En cuanto a la razón que justifica la existencia de un requisito, en los metamodelos que lo tienen en cuenta [52, 107, 189, 224, 281, 348], se modela por igual como una relación de traza entre el concepto Requisito y el concepto Razon (*rationale*), o simplemente como una propiedad que caracteriza al requisito. Realmente, bajo nuestro punto de vista, como la justificación se puede expresar como una cadena de texto, no sería necesario definir un concepto en el metamodelo sino que es suficiente con incluir este detalle como parte de la información (o mejor dicho, de la meta-información) del requisito. Se implemente de una forma o de otra, lo realmente importante es que esta información se tenga en cuenta, puesto que puede ser necesaria en la fase de negociación de requisitos, y vemos que sólo un 27% del total de trabajos, indican su uso explícitamente.

La relación de traza pre-RS menos utilizada es aquella que liga los requisitos con información que complementa la especificación (requisito-suplemento). Sólo tres de los trabajos [107, 281, 348] destacan la necesidad de recoger este tipo de información, lo que representa el 14% del total de los trabajos analizados, aproximadamente.

En cuanto a las relaciones de traza post-RS, todas las propuestas coinciden en modelar las relaciones de traza entre requisitos, reconociendo, por tanto, que los requisitos no se especifican de forma aislada sino que dependen unos de otros de alguna manera. El número de relaciones definidas varía de un trabajo a otro pero, de nuevo, existe un consenso en identificar las relaciones de traza que se encuadrarían en la categoría *refina* del marco de comparación (Tabla 12). En cada metamodelo se utilizan nombres diferentes para las trazas como, *parte_de* [189, 214, 224, 225, 281], *padre-hijo* [77, 348], *proviene_de* [77, 145, 199, 281, 355] o *contiene* [107, 145, 199], que, en ocasiones, son semánticamente equivalentes, a pesar de tener nombres distintos. Por ejemplo, Dubois *et al.* [77] explican que la relación *proviene_de* se utiliza para conectar un requisito *hijo* con su fuente, el requisito *padre*, por tanto, es equivalente a la relación padre/hijo especificada en otros metamodelos. Esta situación nos hace ver, una vez más, la importancia de establecer un modelo de referencia común para IR, en el que se unifique no sólo el nombre, sino la semántica, de las relaciones de traza entre-requisitos. También, lo que está claro, es que los autores están de acuerdo en que es frecuente que un requisito forme parte, se derive o refine la definición dada en otro requisito.

También existe cierto grado de consenso (73%) en el uso de una relación de traza entre requisitos genérica (*influye* en el marco de comparación, Tabla 12), en el sentido de

que simplemente establece que los requisitos están relacionados, sin que uno requiera a otro, estén en conflicto o uno forme parte de otro. En realidad, esta es una relación de traza común, disponible en las herramientas de gestión de requisitos comerciales, que, por lo general, soportan un modelo conceptual simple. Un ejemplo de esto son las relaciones *trace_to* y *trace_from* de RequisitePro.

Por otro lado, casi la mitad de los metamodelos analizados (45,5%) representan la existencia de algún tipo de conflicto entre los requisitos, aunque la naturaleza de este conflicto puede ser diferente. En algunos casos se especifica que un requisito *excluye* a otro requisito [52, 243, 348], que lo *contradice* [204], que *restringe* [199, 308] o, simplemente, que existe algún *conflicto* [107, 225, 375]. Bajo nuestro punto de vista, el modelado de la relación de exclusión entre requisitos tiene sentido en el modelado de una SPL, para especificar posible puntos de variación. En la especificación de requisitos de un producto concreto dentro de la familia esta relación no debe existir, dado que sólo uno de los dos requisitos habrá sido elegido. La representación de conflictos entre requisitos puede tener sentido en las primeras etapas del proceso de IR, pero tras la fase de negociación, los conflictos han debido resolverse y no deben estar presentes en la especificación final de los requisitos de una aplicación.

No es común la definición de la relación de traza *requiere* (Tabla 12), tan sólo el 27% de los trabajos la incluyen [52, 107, 204, 243, 301, 348]. Tampoco la relación catalogada como *coste/valor*, pues el único metamodelo que la especifica es REMM [348]. Como explicamos más adelante, en la Sección 5.3.3, nuestra decisión de modelar esta relación de traza se basa en el estudio, no sólo del trabajo de Dahlstedt y Persson [67], sino también en el realizado por Carlshamre *et al.* [48], en el que concluyen que este tipo de traza es la más común (del 59% al 90% de todas las relaciones identificadas) en tres de los cinco casos de estudios que abordaron. También Alan M. Davis [71] destaca la utilidad de este tipo de trazas, que pueden servir como apoyo en la fase de negociación de los requisitos.

Merece una mención aparte el trabajo de Sánchez *et al.* [301], en el que proponen un metamodelo de requisitos sencillo, que se puede resumir en que: un catálogo contiene requisitos conectados mediante relaciones. El problema de este metamodelo es que no define los tipos de relaciones que pueden existir entre los requisitos, sino que estos tipos de relaciones se especifican en el momento en el que se crean los modelos de requisitos. La ventaja de este diseño es que se puede especificar cualquier tipo de relación entre requisitos, pero el principal inconveniente es que no garantiza que haya consistencia entre las relaciones definidas en distintos modelos de requisitos. Una discusión más detallada de este problema se presenta en el Capítulo 8.

No existe consenso, ni siquiera entre las propuestas que están más centradas en la definición de un metamodelo como modelo de referencia para la especificación de requisitos, acerca de si los casos de prueba deben formar parte, o no, de la especificación de requisitos. Por ejemplo, Kaindl *et al.* [154] y Monzón y Dueñas [224] no incluyen esta relación de traza. De acuerdo con las buenas prácticas de IR, recomendadas por Sommerville y Sawyer [329], la definición de los casos de prueba al mismo tiempo que se especifican los requisitos puede servir para detectar problemas tales como requisitos ambiguos o incompletos. El 41% de los trabajos no siguen esta recomendación.

Todos los trabajos que establecen como objetivo la trazabilidad de requisitos (ver Sección 3.2.3.1, Tabla 15, Tabla 16 y Tabla 17), incluyen explícitamente la relación que

liga el requisito con otro artefacto del desarrollo que lo implementa, o lo representa a nivel de análisis o diseño (traza *requisito-artefacto* en el marco de comparación, Tabla 12). En particular, las propuestas que extienden el metamodelo de SysML para el modelado de requisitos [77, 145, 199], heredan las relaciones definidas en este estándar, que incluyen la traza que liga el requisitos con el caso de prueba (o cualquier otro elemento de un modelo) que lo *verifica* y la que describe cómo un modelo de diseño o implementación *satisface* uno o más requisitos.

Únicamente el 32% de los trabajos no modelan las relaciones de traza entre los requisitos y otros artefactos del desarrollo [107, 154, 218, 243, 348, 355, 375]. Sin embargo, hay que tener en cuenta algunas salvedades:

- Goknil *et al.* [107] plantean como trabajo futuro la definición de modelos de traza que ligue los modelos de requisitos con los modelos de diseño, que constituirían las trazas *requisito-artefacto* definidas en el marco de comparación.
- El trabajo publicado por Vicente-Chicote *et al.* [348], constituye la primera versión del metamodelo de requisitos que se presenta en esta tesis doctoral. En esta primera versión las trazas *requisito-artefacto* se presentaban como trabajo futuro y se han concretado en la versión actual (Capítulo 5).
- El metamodelo de requisitos propuesto por Navarro *et al.* [243] forma parte de una metodología de DSDM [241, 242] que, según los autores, *ha sido definida para guiar la definición de forma concurrente de los requisitos y la arquitectura del software, prestando especial atención a la trazabilidad entre ellos* [242]. Por tanto, aunque no se vea expresado explícitamente en el metamodelo de requisitos propuesto en el trabajo analizado [241], es evidente que la metodología debe soportar la relación de traza *requisito-artefacto*.
- En la propuesta de Vogel y Mantell [355] mantienen el modelo de trazabilidad independiente del metamodelo de requisitos, lo que permite trazar cualquier artefacto creado durante el proceso DSDM con cualquier otro, en particular los requisitos. En consecuencia, la relación de traza *requisito-artefacto* es soportada, aunque no directamente en el metamodelo de requisitos.
- Zikra *et al.* [375] presentan un metamodelo que integra modelos de empresa y requisitos con modelos de diseño en el marco del DSDM. Los conceptos del metamodelo se agrupan en seis modelos, entre los que se encuentran el modelo de requisitos y el modelo de arquitectura. Los autores establecen relaciones entre modelos que incluye la relación de que el modelo de requisitos se implementa en el modelo de arquitectura. Sin embargo, esta relación entre-modelos, que debería traducirse en alguna relación que ligue los conceptos definidos en ambos modelos no se detalla. Podríamos decir que, potencialmente, el metamodelo que proponen soporta la traza *requisito-artefacto*.

En consecuencia, podemos decir que sólo dos de los trabajos revisados no tienen en cuenta las relaciones de traza *requisito-artefacto*. Así, se puede concluir, que existe cierto grado de consenso entre los autores acerca de la conveniencia de modelar este tipo de relación de traza en un metamodelo que aporte una solución a la trazabilidad de requisitos. Muy posiblemente este consenso esté motivado porque se trata de una exigencia de los estándares de calidad, como CMMi [309].

3.2.4.2. PI2 ¿Cómo se integran los modelos de requisitos en procesos DSDM, esto es, cuáles son las técnicas del DSDM utilizadas?

En esta sección vamos a presentar los aspectos más relevantes de las propuestas que se enmarcan en el DSDM (Tabla 15, Tabla 16 y Tabla 17).

De acuerdo con Goknil *et al.* [107], los desarrolladores necesitan contar con un metamodelo de requisitos para mantener una continuidad en el proceso DSDM, tratando todos los artefactos que se producen en cada etapa como modelos. Dado que existen diferentes aproximaciones para abordar la etapa de IR, los autores proponen un metamodelo base (*core metamodel*) que se puede adaptar para dar soporte a distintas técnicas de especificación de requisitos. En este trabajo, el metamodelo se expresa como una ontología (utilizando el lenguaje OWL [31]), de esta forma se pueden utilizar las capacidades de razonamiento de las herramientas de soporte a la ontología. El objetivo es especificar reglas de inferencia genéricas para el metamodelo base, y aplicarlas para los metamodelos adaptados, formalizando la semántica de las relaciones del metamodelo. Estas reglas de inferencia permiten derivar nuevas relaciones entre los requisitos a partir de las establecidas en el modelo de requisitos. Goknil *et al.* plantean la necesidad de definir modelos de traza que ligen los requisitos con el diseño de la arquitectura. Estos modelos de traza permitirán determinar el impacto de los cambios en los modelos de requisitos en los modelos de diseño. No obstante, este aspecto queda pendiente como trabajo futuro en este artículo.

Ji *et al.* [145] señalan a la IR dirigida por modelos (MDRE, *Model Driven Requirements Engineering*) como un enfoque prometedor, y bien soportado, para abordar la complejidad de los sistemas de automatización industrial, representando las especificaciones de requisitos como modelos formales que son correctos, completos, consistentes, no ambiguos y fáciles de leer y de mantener. Sin embargo, a juicio de los autores, no existe un lenguaje de modelado estandarizado y universal que cubra el proceso completo de IR desde la especificación de requisitos, la asignación en los modelos de diseño, hasta los modelos de verificación. Los autores explican las carencias del modelo de requisitos propuesto por el OMG en el lenguaje SysML y proponen una extensión que sirva para aplicaciones industriales reales, implementada como un nuevo perfil UML denominado MDRE4BR (*MDRE for Bosch Rexroth*). El metamodelo permite establecer relaciones *satisfice* entre los requisitos y los objetos de diseño y *verifica* para ligar el resultado de los casos de prueba con el estado del requisito relacionado. Aunque esta propuesta se centra en presentar el enfoque MDRE, los autores señalan la posibilidad de generar los modelos para la simulación de sistemas, aplicando los resultados de estudios recientes que integran SysML con Modelica [6].

La propuesta de Merschen *et al.* [214] se encuadra en el análisis de líneas de productos para los sistemas embebidos. En este trabajo, se aplican las técnicas del DSDM para abordar los retos relacionados con: 1) la trazabilidad de las señales de los modelos Simulink [11], 2) los aspectos de evolución de la variabilidad y 3) la trazabilidad entre los artefactos, por ejemplo, identificar qué requisito afecta a qué bloque (en un modelo Simulink) y qué señal. El modelo Simulink representa la implementación de la funcionalidad. Después de pequeños cambios estos modelos se utilizan para generar código que luego se ejecuta en coches reales. Si bien el metamodelo de requisitos propuesto es sencillo, el aspecto interesante que se pone de relieve en esta aproximación,

es la necesidad de representar la especificación de requisitos como un modelo de manera que se puede generar automáticamente, mediante transformaciones de modelos, las dependencias entre los artefactos. A partir de este modelo de traza, también mediante transformaciones de modelos, se puede filtrar sólo las dependencias (subsistemas del modelo Simulink y casos de prueba) que afectan a un requisito dado.

Merschen *et al.* destacan que aunque tanto IBM Rational DOORS [123] como Matlab/Simulink soportan la trazabilidad, este soporte no es suficiente en la práctica porque se tiene que especificar durante la etapa de elicitación de requisitos los subsistemas relacionados con un requisito. Además, el autor del modelo de requisitos suele ser diferente del autor del modelo Simulink, y es posible que no tenga la formación técnica adecuada. Por tanto, el desarrollador tendrá que decidir si un nuevo subsistema es necesario y entonces adaptar adecuadamente el documento de requisitos.

Molina *et al.* [218] proponen la integración de modelos de requisitos dirigidos por objetivos, métricas de software y el proceso DSDM con capacidades de automatización, mediante la definición de un metamodelo que soporte la definición de requisitos medibles. Este metamodelo, que enlaza explícitamente objetivos, requisitos y métricas, se expresa mediante un perfil UML (basado en el lenguaje de requisitos orientado a objetivos GRL [12]). Los motivos para la definición de un perfil son, desde el punto de vista de los autores, 1) favorecer su adopción a los desarrolladores acostumbrados a UML y 2) facilitar la inclusión de los requisitos medibles en enfoques DSDM existentes, lo que por su parte facilita la validación de los requisitos respecto al resto de artefactos del desarrollo. El trabajo que se presenta se centra únicamente en la definición del metamodelo de requisitos y no en su integración en un proceso DSDM.

Como vimos en la Sección 3.2.3.1, en esta misma línea de requisitos medibles se enmarca el trabajo de Monperrus *et al.* [223], aunque con una aproximación diferente. Mientras que el objetivo del metamodelo de Molina *et al.* es modelar requisitos medibles, el metamodelo propuesto por Monperrus *et al.* se ha definido expresamente para que soporte las métricas que miden la calidad de las especificaciones de requisitos. Para implementar las métricas es para lo que se utiliza el DSDM, porque permite crear una descripción formal y ejecutable de las métricas de requisitos. Por tanto, los autores no marcan como objetivo final la integración de los requisitos en un proceso de DSDM, sino que utilizan la tecnología de este tipo de desarrollo con otro fin.

Zikra *et al.* [375] abordan el modelado de empresa (objetivos y procesos del negocio) en el contexto del DSDM. Según los autores, la mayoría de los enfoques DSDM asumen implícitamente que la creación del modelo inicial del proceso es responsabilidad del modelador y proporcionan pocas guías, en unos casos, o ninguna, en otros, para iniciar el desarrollo. Este modelo inicial debe representar requisitos y diseños organizacionales, a menudo representado por un modelo de empresa. Para cubrir este hueco, Zikra *et al.* proponen EKD (*Enterprise Knowledge Development*) como enfoque para el modelado de empresa que se integre en el DSDM. EKD está compuesto por seis modelos relacionados, que incluyen: modelo de objetivos, modelo de reglas de negocio, modelo conceptual, modelo de procesos de negocio, modelo de requisitos y modelo de la arquitectura. El trabajo se centra en la descripción de los metamodelos para el modelado de la empresa, pero no en su integración en un proceso DSDM.

También destacamos en la Sección 3.2.3.1, que sólo en tres trabajos se presenta una visión completa de un proceso de desarrollo DSDM en el que integran los modelos de

requisitos. Por un lado, Sánchez *et al.* [301] proponen un metamodelo de requisitos sencillo, puesto que su único objetivo es definir un catálogo de requisitos reutilizables, que estén trazados explícitamente con modelos de diseño. De esta forma, al seleccionar un requisito del catálogo, se estaría reutilizando, al mismo tiempo, el modelo de diseño con el que está relacionando (esta propuesta se estudia con más detalle en el Capítulo 8). Componiendo los modelos de diseño, relacionados con los requisitos reutilizados, se obtendría el modelo de diseño de la aplicación, a partir del cual se puede generar el código. Por su parte, Navarro *et al.* [241-243] incluyen en el metamodelo de requisitos el concepto de *operationalization*, esto es, el modo en el que se va a abordar el objetivo marcado (*goal*) en el sistema. Este concepto es el que sirve para mantener las relaciones de traza entre el modelo de requisitos y el modelo de la arquitectura. Cada *operationalization* se desarrolla mediante un diagrama de secuencia (*scenarío model*) que muestra cómo los elementos de la arquitectura colaboran para llevar a cabo la *operationalization*. Los diagramas de secuencia son la entrada de las transformaciones de modelos que terminan en la generación de código. Por otro lado, Vogel y Mantell [355] se centran en la definición de un proceso de desarrollo para la adopción de MDA en PYMES (*Pequeñas y Medianas Empresas*). Uno de los elementos principales es el metamodelo de requisitos, expresado mediante un perfil UML, necesario para formalizar los requisitos, mejorar su calidad, facilitar su validación y la trazabilidad hacia otros artefactos del desarrollo. En este trabajo se da sólo una visión general del marco completo de desarrollo, que se detalla en Mansell *et al.* [200].

Por último, nos queda por evaluar hasta qué punto se implementan transformaciones de modelos a partir de los requisitos. De los pocos trabajos que proponen algún tipo de transformación (Tabla 21), éstas tienen que ver con: 1) transformaciones modelo a texto (m2t) [348, 355], para generar los documentos de requisitos (SRS, *Software Requirements Specification*) a partir de la especificación representada en los modelos, en formato ODF (*Open Document Format*) [355] o HTML [348], 2) transformaciones texto a modelo (t2m) [214], para representar la especificación de requisitos en modelos, o bien 3) transformaciones modelo a modelo (m2m) [214] para generar el modelo de traza que ligue los requisitos con el modelo de diseño y las pruebas. Insistimos de nuevo en el hecho de que en ninguno de los trabajos se definen transformaciones m2m, tomando como modelo fuente el modelo de requisitos.

Id.	Integración DSDM	Representación del metamodelo	Req- artefacto		Transformaciones			
			I	E	Tipo	Modelo fuente	Modelo destino	Lenguaje de transformación
Vogel06 [355]	■	Perfil UML	□	■	m2t	Modelo de requisitos	ODF SRS	MOFScript
Moros0708 [234, 348]	▣	MOF	□	■	m2t	Modelo de requisitos	HTML SRS (IEEE-830)	MOFScript
Merschen11 [214]	▣	MOF	□	■	t2m	Doc. requisitos DOORS	Modelo de requisitos	Xtext
					m2m	Modelos de: 1. requisitos 2. Simulink 3. Pruebas	Modelo de Trazabilidad	ATL

Tabla 21 Propuestas de transformaciones a partir de los requisitos.

Aunque en las versiones iniciales de este trabajo sólo se abordaban las transformaciones modelo a texto (como se puede ver en la Tabla 21), en la versión final, que forma parte de esta tesis doctoral, se han definido también transformaciones modelo a modelo que permiten generar los modelos conformes a un DSL a partir del modelo de requisitos (Capítulo 6).

3.2.4.3. Limitaciones del estudio

Por un lado, es posible que hayamos podido dejar fuera algún trabajo en el que se presente un “modelo de referencia” para IR pero que no lo nombre o lo represente explícitamente como “metamodelo” de requisitos. No obstante, dado el volumen de trabajos analizados, pensamos que no hemos omitido ningún trabajo relevante. Por otro lado, precisamente debido al gran número de propuestas encontradas, ha sido una labor ardua de análisis y selección, en la que hemos podido cometer algún error sólo por el hecho de ser únicamente dos personas, la doctoranda y el director de tesis (como revisor del proceso), las implicadas en el estudio. Son muchas las aproximaciones que incluyen la definición de un metamodelo de requisitos, pero son difíciles de catalogar y de organizar, puesto que cada una de ellas tiene una motivación diferente y se aplica a dominios y problemas distintos. A pesar de esto, pensamos que los trabajos finalmente seleccionados sirven como muestra del estado del arte en relación a los metamodelos de requisitos y nos revelan los aspectos en los que tiene sentido dedicar esfuerzos en nuevas investigaciones.

3.2.5. Conclusiones

Los resultados de la RSL revelan que es necesario definir un metamodelo de requisitos único, y ampliamente consensuado, que formalice los conceptos y relaciones implicados en el proceso de IR. Los modelos de requisitos, conformes a este metamodelo, deben representar la especificación de requisitos de una aplicación, de manera que pueda trazarse hacia otros artefactos del desarrollo e integrarse en un proceso de DSDM. En este tipo de desarrollos, los artefactos que se maneja son modelos y, por tanto, los requisitos deben modelarse para que haya una continuidad en el proceso.

Al no existir un modelo de referencia para IR comúnmente aceptado, cada autor propone los elementos que, bajo su punto de vista, deberían formar parte de un metamodelo de requisitos. A nuestro juicio, la definición de este metamodelo debería hacerse de acuerdo a las recomendaciones extraídas de estudios empíricos y las buenas prácticas de IR, que hemos recogido en un marco de comparación. Del análisis de los metamodelos, tomando como referencia el marco de comparación, podemos concluir que existe cierto grado de acuerdo en: 1) modelar la relación de los requisitos con el stakeholder que los propone, para dar soporte a las relaciones de traza pre-RS, 2) definir trazas entre requisitos, dado que, los requisitos no se definen de forma aislada sino que están relacionados unos con otros, mediante relaciones que pueden ser de distinta naturaleza (en esto es en lo que difieren unas propuestas de otras) y 3) trazar los requisitos hacia otros artefactos del desarrollo. El metamodelo propuesto en esta tesis doctoral (Capítulo 5), coincide con estas valoraciones y además cumple con todas las recomendaciones fijadas en el marco de comparación.

En cuanto a la integración de los modelos de requisitos en el DSDM, en muchas propuestas asumen que el hecho de modelar los requisitos, a partir de un metamodelo, ya permite la integración de las especificaciones de requisitos en un proceso de DSDM. Los requisitos, por tanto, se trazan explícitamente con el resto de artefactos del desarrollo, pero no se definen transformaciones modelo a modelo a partir de los modelos de requisitos. Esta es la línea de investigación central de esta tesis doctoral, encontrar las herramientas necesarias para poder transformar los modelos de requisitos en modelos conformes a un DSL (modelo de la aplicación). De esta forma, siguiendo el enfoque de modelado específico de dominio, a partir de los modelos de la aplicación, se generaría el código de la aplicación. En consecuencia, en este proceso de DSDM el modelo inicial sería el modelo de requisitos, a partir del cual, mediante transformaciones de modelos, se obtendría el código de la aplicación.

3.3. RSL transformaciones de modelos de requisitos a otros modelos del desarrollo

Para establecer el contexto de la RSL parafraseamos a Kleppe *et al.* [170] que nos define los elementos implicados en las transformaciones de modelos. Una *transformación* es la generación automática de un modelo destino a partir de un modelo fuente conforme a una definición de transformación. Una *definición de transformación* es un conjunto de reglas de transformación que de forma conjunta describen cómo un modelo en el lenguaje origen puede transformarse en un modelo del lenguaje destino. Una *regla de transformación* es una descripción de cómo uno o más constructores del lenguaje fuente se pueden transformar en uno o más constructores del lenguaje destino.

El objetivo general de la RSL es el estudio de las propuestas existentes en la literatura en relación a las transformaciones automáticas (o semiautomáticas) desde modelos de requisitos hacia otros modelos del desarrollo con el objeto de evaluar hasta qué punto los requisitos textuales son tenidos en cuenta en estas aproximaciones. Para ello vamos a seguir el mismo esquema de la Sección 3.2, en primer lugar, se presenta el diseño del estudio (Sección 3.3.1) A continuación, se muestran los resultados de la RSL (Sección 3.3.2), tanto el resumen de los trabajos candidatos y finalmente seleccionados (Sección 3.3.2.1) como el resultado de la evaluación de la calidad de los trabajos (Sección 3.3.2.2). Después, se discuten las preguntas de investigación planteadas de acuerdo con los resultados obtenidos y las limitaciones del estudio (Sección 3.3.3). Por último, se destacan las principales conclusiones derivadas de los resultados obtenidos (Sección 3.3.4).

3.3.1. Diseño de la revisión sistemática en relación a las transformaciones a partir de modelos de requisitos

En esta sección se presenta el protocolo de la RSL que incluye: las cuestiones de investigación que guían la RSL (Sección 3.3.1.1), la planificación del proceso de búsqueda (Sección 3.3.1.2), la definición de los criterios de inclusión y exclusión (Sección 3.3.1.3), los criterios utilizados para evaluar la calidad de los trabajos (Sección 3.3.1.4), el proceso de recogida de datos (Sección 3.3.1.5) y de análisis de los datos (Sección 3.3.1.6) y, por último

las desviaciones producidas durante el proceso respecto al protocolo establecido (Sección 3.3.1.7). De acuerdo a los consejos de Kitchenham [164] para la elaboración de una RSL por parte de un estudiante de doctorado, este protocolo fue desarrollado por la doctoranda y presentado al director de esta tesis doctoral para su revisión y crítica.

3.3.1.1. Cuestiones de investigación

Las preguntas de investigación que se pretenden contestar en la RSL son:

PI1: ¿Cuáles son los modelos de requisitos a partir de los cuáles se pueden generar otros modelos? ¿Tienen estos modelos en cuenta los requisitos textuales?

PI2: ¿Cuáles son los modelos de desarrollo posteriores (modelos de análisis, por ejemplo) generados a partir de los modelos de requisitos?

3.3.1.2. Proceso de búsqueda

El proceso de búsqueda ha sido automático, utilizando para ello los motores de búsqueda basados en palabras clave de las bibliotecas electrónicas siguientes:

- IEEE Digital Library (www.computer.org/portal/site/csdl/index.jsp)
- ACM Digital Library (portal.acm.org)
- ScienceDirect (www.sciencedirect.com)
- MetaPress (Kluwer + Springer) (www.metapress.com)
- Wiley InterScience (www.interscience.wiley.com)

Como ya se apuntaba en la Sección 2.2.2, IEEE, ACM y ScienceDirect son las bibliotecas electrónicas más utilizadas en las RSL relacionadas con la Ingeniería del Software. Además, a partir de estas fuentes seleccionadas se tiene acceso a las principales revistas y eventos de las comunidades de requisitos (por ejemplo, la revista REJ) y del DSDM (por ejemplo la conferencia MODELS). En esta ocasión GoogleScholar no ha sido incluido puesto que dada su amplia cobertura [59] se repiten prácticamente todos los trabajos previamente seleccionados en otros motores de búsqueda.

Se han utilizado dos cadenas de búsqueda diferentes para poder abarcar todos los trabajos encuadrados en enfoques generales de DSDM (C1) como en las propuestas específicas en el marco de MDA (C2)—el lector puede encontrar una breve introducción a MDA en la Sección 1.1.3.2—. De este modo, las cadenas de búsqueda utilizadas en esta RSL son:

C1) “requirements model” **AND** “model transformation” **AND**
 (“model driven” **OR** “model based”)

C2) “from CIM to PIM”

Para cada uno de los trabajos obtenidos como resultado de las búsquedas, era suficiente con la lectura del título y el resumen para seleccionarlo, o no, como trabaja candidato dependiendo de si hacían referencia a transformaciones entre los requisitos y

cualquier otro modelo del desarrollo. Posteriormente, para cada uno de los trabajos candidatos, fue necesaria una lectura más cuidadosa de la introducción y las conclusiones para poder aplicar los criterios de inclusión y exclusión (Sección 3.3.1.3) y seleccionarlos como trabajos relevantes.

3.3.1.3. Criterios de inclusión y exclusión

Dado que en el DSDM el proceso de transformación implica que se parte de un modelo origen, conforme a un metamodelo, para, aplicando unas reglas de transformación generar automáticamente un modelo destino, conforme al mismo u otro metamodelo (ver Sección 3.3), en la RSL se incluirán aquellos trabajos que tengan claramente diferenciados estos componentes: modelo de requisitos como origen, otro modelo del desarrollo como destino y reglas de transformación automáticas (o semi-automáticas) que se apliquen para generar dichos modelos destino. De acuerdo con la clasificación de Mens y van Gorp [213], la RSL la vamos a centrar en *transformaciones verticales*, esto es, en transformaciones en las que el modelo de origen y destino pertenece a distintos niveles de abstracción. La idea es que consigamos “avanzar” en el proceso de desarrollo tomando como modelo de partida el modelo de requisitos.

Por tanto, quedan excluidos del estudio:

- Propuestas en las que los requisitos no estén representados en un modelo conforme a un metamodelo.
- Los trabajos que presenten únicamente guías metodológicas para la obtención de un modelo a partir de los requisitos que no estén automatizadas.
- Trabajos que propongan transformaciones entre modelos que se encuentren en el mismo nivel de abstracción, esto es, que propongan transformaciones entre modelos de requisitos. Siendo rigurosos, una propuesta de transformación entre requisitos de sistema y requisitos software realmente implica dos niveles de abstracción diferentes, pero en aras de la simplificación, equiparamos los niveles de abstracción con los niveles del desarrollo del software (requisitos, análisis, diseño e implementación).
- Trabajos duplicados del mismo estudio. En el caso de que existan versiones diferentes de un mismo enfoque se toma la más completa o bien la que incluya el proceso de transformación de los modelos de requisitos si en la versión completa no se detalla.
- Trabajos que no alcancen una valoración de calidad positiva de acuerdo a los criterios de calidad que se han definido (Sección 3.3.1.4).

3.3.1.4. Criterios de calidad

Al igual que los criterios de calidad establecidos para la RSL de los metamodelos de requisitos (Sección 3.2.2.4), se han fijado distintos criterios para evaluar los trabajos seleccionados, inspirados en el trabajo de Fernández *et al.* [88], e incluyen:

- CC1) ***Conexión con los requisitos textuales***. Dada la importancia de los requisitos textuales en el proceso de IR consideramos que toda propuesta que de soporte a la definición de un modelo de requisitos debe incluir los requisitos textuales o

deben mantener relaciones de traza a ellos de forma explícita o implícita. De forma explícita significa que la relación de traza está definida en el metamodelo, mientras que de forma implícita implica que el modelo de requisitos se ha obtenido a partir de los requisitos textuales mediante una transformación de modelos. Por tanto, los posibles valores son: ***Si***, se valorará con un 1, o ***No***, se valorará con un -1.

- CC2) ***Integración en un proceso completo de DSDM*** La solución que se propone en los trabajos puede ser ***parcial***, si sólo presentan la transformación de los modelos de requisitos al siguiente modelo del desarrollo, o puede ser ***total***, si la transformación de requisitos a otros modelos forma parte de una propuesta completa de DSDM desde los requisitos hasta el código. Las soluciones parciales serán valoradas con un 0, mientras que las soluciones totales serán valoradas con un 1.
- CC3) ***Grado de automatización***. Una transformación puede ser automática, si no es necesaria la intervención del analista en el proceso de transformación, o puede ser semi-automática, en caso contrario. Si la solución propuesta es automática será valorada con un 1, mientras que si es semi-automática será valorada con un 0.
- CC4) ***Validación de la propuesta***. Igual que en la Sección 3.2.2.4, la propuesta será valorada con: un 1, si se ha utilizado para su validación algún método empírico aplicado en un entorno industrial o en un contexto real; un 0, si se ha aplicado algún método de validación en un entorno académico; y un -1 en el caso de que la propuesta no haya sido validada.
- CC5) ***Lugar de publicación***. Igual que en la Sección 3.2.2.4, tendremos en cuenta la calidad de la conferencia o revista donde se ha publicado el trabajo, utilizando el ranking de conferencias ERA y el listado JCR, respectivamente. Así, los valores asignados a cada trabajo en función de su relevancia son los definidos en la Tabla 13.
- CC6) ***Citado por otros***. Igual que en la Sección 3.2.2.4, los trabajos que no hayan sido referenciados por nadie serán valorados con un 0, si se publicaron en 2012, y un -1 si se publicaron en un año anterior. En el caso de que el trabajo tenga de 1 a 5 citas será valorado con un 1, y en el caso de que tenga más de 5 citas, será valorado con un 2.

Aplicando estos criterios de calidad a los trabajos obtenidos en el proceso de búsqueda y selección, la valoración final de cada uno variará entre -4 y 8.

3.3.15. Recogida de datos

Para cada uno de los trabajos seleccionados se extrajeron los datos necesarios tanto para dar respuesta a las preguntas de investigación que se han planteado en la RSL (Sección 3.3.1.1), como para poder evaluar la calidad del estudio de acuerdo a los criterios de calidad establecidos en el protocolo (Sección 3.3.1.4). Estos datos incluyen:

- Origen del estudio: fuente a partir de la cual se ha obtenido el estudio
- Referencia bibliográfica completa
- Dominio de aplicación, con el objeto de clasificar los trabajos.

- Respuestas a las preguntas de investigación: modelo de requisitos origen y modelo de requisitos destino
- Cuestiones relacionadas con la integración en un proceso DSDM: si la propuesta se enmarca en el DSDM y si se integra de manera parcial o en un proceso completo de desarrollo. En el caso de que el metamodelo se integre en el DSDM, si los requisitos se trazan implícita o explícitamente con otros artefactos del desarrollo, tipo de transformación, lenguaje utilizado y herramienta de soporte.
- Aspectos relacionados con la evaluación de la calidad que no han sido incluidos en los puntos anteriores: método de validación utilizado, número de veces referenciado por otros autores.

El proceso de recogida de datos fue llevado a cabo por esta doctoranda. Para garantizar la consistencia de los datos extraídos, de acuerdo con las guías de Kitchenham [164] para los estudiantes de doctorando, se llevó a cabo un proceso test-retest de manera que se extrajeron los datos por segunda vez de un conjunto aleatorio de trabajos.

3.3.16. Análisis de datos

Los datos recogidos han sido tabulados para mostrar:

- El identificador asignado al estudio, fuente, autor/es, referencia y año de publicación.
- La consideración o no de los requisitos textuales en la propuesta (en relación a la pregunta de investigación PI1).
- Los modelos origen y destino implicados en la transformación (en relación a las preguntas de investigación PI1 y PI2).
- Aspectos técnicos del proceso de transformación como: si se encuadra o no en el marco de MDA, el tipo de transformación, automática (A) o semi-automática (S-A), el lenguaje utilizado para la transformación y si el proceso está soportado o no por una herramienta.
- El método de validación y el dominio de aplicación.

3.3.17. Desviaciones del protocolo

Tras una serie de búsquedas tentativas con la cadena C1 decidimos incluir la cadena de búsqueda C2 (ver Sección 3.3.1.2) puesto que algunos trabajos que proponían soluciones en el ámbito de MDA y que eran conocidos previamente no eran encontrados. Si bien las cadenas “*model-driven*” y “*model-based*” son bastante generales y deberían incluir cualquier aproximación en el marco del DSDM, hablar de transformaciones de modelos CIM (nivel de requisitos) a modelos PIM (siguiente nivel en el proceso de desarrollo MDA) es la terminología más común en los trabajos propuestos en el marco de MDA. La inclusión de esta cadena de búsqueda se puede considerar como una desviación del protocolo establecido inicialmente.

La sección de trabajos relacionados de cada uno de los artículos seleccionados ha sido revisada cuidadosamente para detectar referencias relacionadas con las preguntas de investigación que no hubieran sido encontradas por los motores de búsqueda. Sin

embargo, ninguna de las referencias que a priori podían parecer interesantes han sido finalmente incluidas en la RSL y no se ha producido ninguna desviación en este sentido.

3.3.2. Resultados

En esta sección se resumen los resultados del proceso de búsqueda incluyendo tanto los estudios candidatos como los que finalmente han sido seleccionados y analizados (Sección 3.3.2.1). Además, también se evalúa la calidad de los trabajos seleccionados (Sección 3.3.2.2) de acuerdo a los criterios de calidad establecidos en la Sección 3.3.1.4.

3.3.2.1. Resultados de las búsquedas

El número de trabajos obtenido como resultado de la búsqueda de cada una de las cadenas en las bases de datos electrónicas (Sección 3.3.1.4) se puede ver en la Tabla 22.

	C1	C2	Relevantes	Seleccionados
ACM	26	11	6	1
IEEE	100 ¹	83	10	6
ScienceDirect (SD)	46	10	9	5
Wiley	7	1	0	0
MetaPress (MP)	101	30	18	2
Total	154	135	43	14

Tabla 22 Resultados de las búsquedas de transformaciones a partir de modelos de requisitos.

En esta tabla se muestra como de los 43 trabajos relevantes, finalmente 14 fueron seleccionados (aplicando los criterios de inclusión y exclusión) para un estudio en profundidad. Estos trabajos serán presentados con más detalle en la Sección 3.3.3. Siguiendo las recomendaciones de Kitchenham [164] los trabajos seleccionados y rechazados fueron discutidos con el director de esta tesis doctoral para favorecer la fiabilidad de las decisiones tomadas por esta doctoranda. Además, los trabajos han sido re-evaluados tras su inclusión en la RSL durante la elaboración de este documento lo que garantiza la consistencia de las decisiones acordadas.

Trabajos excluidos de la RSL

La Tabla 23 y la Tabla 24 muestran el listado de los trabajos potencialmente relevantes que fueron rechazados y el motivo que nos llevó a tomar esta decisión. Merece la pena destacar los trabajos que proponen procesos sistemáticos para la obtención de modelos a partir de los requisitos pero que han sido excluidos porque no están automatizados [50, 109, 278, 372]. En algunos casos, como en [50], la formalización de las transformaciones con un lenguaje de transformación de modelos (ATL, en este caso) se presenta como un trabajo futuro.

¹ De 5333 de los resultados limitados a: conference proceeding, IEEE transactions on Software Engineering, IEEE transactions on Computers, Software Magazine, Computer Magazine y Computing in Science and Engineering Magazine. Y la cadena de búsqueda sólo con “model-driven”

Fuente	Autores	Título	Año	Motivo de exclusión
ACM	N. Koch	<i>Transformation techniques in the model-driven development process of UWE</i>	2006	Presenta el proceso completo, la transformación CIM2PIM se detalla en el otro trabajo del mismo año.
	N. Koch <i>et al.</i>	<i>Model transformations from requirements to web system design</i>	2006	Misma tecnología que el trabajo seleccionado más reciente Escalona08
	M. Lucena <i>et al.</i>	<i>Stream: a strategy for transition between requirements models and architectural models</i>	2011	Más reciente el trabajo de Castro <i>et al.</i> de 2012
	D. Merschen <i>et al.</i>	<i>Experiences of applying model-based analysis to support the development of automotive software product lines</i>	2011	Las transformaciones de modelos se emplean para generar trazas no para transformar los modelos
	W. Zhang <i>et al.</i>	<i>Transformation from CIM to PIM: a feature-oriented component-based approach</i>	2005	Proceso guiado no automático
IEEE	D. de Almeida Ferreira y Alberto Rodrigues da Silva	<i>A Controlled Natural Language Approach for Integrating Requirements and Model-Driven Engineering</i>	2009	Plantean la generación de modelos de requisitos a partir de lenguaje natural para integrarlos en el DSDM pero no dicen cómo transformar los modelos de requisitos. (<i>Doctoral proposal</i>)
	S. Kherraf <i>et al.</i>	<i>Transformation from CIM to PIM Using Patterns and Archetypes</i>	2008	La formalización y automatización de las transformaciones lo plantean como trabajo futuro.
	C. Ponsard y M. Delehay	<i>Towards a Model-Driven Approach for Mapping Requirements on AADL Architectures</i>	2009	Mapeo entre el modelo de objetivos y la arquitectura en el lenguaje AADL no automatizado.
	Siqueira y Silva	<i>Mapping the WRSMP Model to Model-Driven Architecture Models</i>	2011	Establece un modelo de referencia para RE y mapean cada uno de estos elementos en los modelos de MDA. No propone ninguna transformación a partir de modelos de requisitos.
SD	J. Castro <i>et al.</i>	<i>Changing attitudes towards the generation of architectural models</i>	2012	Proceso guiado no automatizado. Plantean como trabajo futuro automatizar las transformaciones con ATL.
	Liu y Zhu	<i>Construct Aspectual Models from Requirement Documents for Model-driven Development of Automotive Software</i>	2011	La propuesta trata de definir un modelo de requisitos único pero no trata la transformación de modelos a partir del modelo de requisitos.
	Maté y Trujillo	<i>A trace metamodel proposal based on the model driven architecture framework for the traceability of user requirements in data warehouses</i>	2012	Este trabajo se centra en la definición de un metamodelo de trazabilidad. La transformación CIM2PIM se presenta en el trabajo seleccionado Mazón07
	Rodríguez <i>et al.</i>	<i>Secure business process model specification through a UML 2.0 activity diagram profile</i>	2011	Se centran en la definición de los modelos de proceso seguros. Las transformaciones las presentan en el trabajo seleccionado Rodríguez10

Tabla 23 Artículos candidatos no seleccionados (1/2).

Fuente	Autores	Título	Año	Motivo de exclusión
MP	Alencar <i>et al.</i>	<i>From i* Requirements Models to Conceptual Models of a Model Driven Development Process</i>	2009	Más reciente el trabajo de Castro <i>et al.</i> de 2012
	Cao <i>et al.</i>	<i>Transformation from computation independent model to platform independent model with pattern</i>	2008	Solicitado a los autores pero no enviado.
	Castro <i>et al.</i>	<i>F-STREAM: A Flexible Process for Deriving Architectures from Requirements Models</i>	2011	Más reciente el trabajo de Castro <i>et al.</i> de 2012
	Donins y Osis	<i>Reconciling software requirements and architectures within MDA</i>	2009	No define claramente un enfoque de transformación en el contexto del DSDM en el que hay que establecer los metamodelos de los modelos origen y destino y el lenguaje de transformación que se utiliza.
	González <i>et al.</i>	<i>Systematic Derivation of Class Diagrams from Communication-Oriented Business Process Models</i>	2011	Proceso sistemático pero no automatizado
	Lucena <i>et al.</i>	<i>A Model Transformation Approach to Derive Architectural Models from Goal-Oriented Requirements Models</i>	2009	Más reciente el trabajo de Castro <i>et al.</i> de 2012
	Machado <i>et al.</i>	<i>Engineering and Managing Software Requirements</i>	2005	Propone un enfoque basado en heurísticas para transformar los requisitos de usuario en requisitos del sistema. Modelos en el mismo nivel de abstracción no automatizado.
	Martínez <i>et al.</i>	<i>From early to late requirements: a goal-based approach</i>	2008	Transformación en el mismo nivel de abstracción.
	Murenecs y Asnina	<i>Automated Derivation of Use Case Model from Topological Functioning Model</i>	2011	Transformaciones en el mismo nivel de abstracción.
	Navarro <i>et al.</i>	<i>MORPHEUS: A Supporting Tool for MDD</i>	2010	Las transformaciones se hacen desde los modelos PIM no desde los requisitos.
	Pastor <i>et al.</i>	<i>Model-Driven Development</i>	2008	Este trabajo establece un mapping entre OO-Method y MDA
	Pimentel <i>et al.</i>	<i>Deriving software architectural models from requirements models for adaptive systems: the STREAM-A approach</i>	2011	Más reciente el trabajo de Castro <i>et al.</i> de 2012
	Sánchez <i>et al.</i>	<i>Towards MDD transformations from AO requirements into AO architecture</i>	2006	Más reciente el trabajo seleccionado en la RSL con id Sanchez10
	Valderás <i>et al.</i>	<i>Transforming web requirements into navigational models: AN MDA based approach</i>	2005	Más reciente el trabajo seleccionado en la RSL con id Valderás09
	Valderás <i>et al.</i>	<i>A transformational approach to produce web application prototypes from a web requirements model</i>	2006	Más reciente el trabajo seleccionado en la RSL con id Valderás09

Tabla 24 Artículos candidatos no seleccionados (2/2).

La definición de buenas arquitecturas software, que proporcionen un plan de diseño de un sistema y que ayude a manejar la complejidad de dicho sistema, ha sido reconocida como un factor crítico para el éxito de un producto [121]. Estas arquitecturas deben derivarse a partir de las especificaciones de requisitos del sistema de manera que el producto desarrollado cumpla con los requisitos establecidos. A pesar de la importancia de estas dos actividades (especificación de requisitos y diseño de arquitecturas software) el conocimiento sobre la forma en la que ambas se relacionan es bastante limitado y constituye una línea de investigación activa. Las investigaciones en esta línea tratan de derivar, de forma más o menos guiada, las arquitecturas del software a partir de las especificaciones de requisitos utilizando diferentes aproximaciones como por ejemplo partiendo de modelos de características [372] o bien partiendo de modelos de objetivos [50, 278].

En esta línea, y en el contexto de MDA, la propuesta de Zhang *et al.* [372] trata de cubrir el hueco que existe entre modelos CIM y PIM, dado que, aunque las transformaciones de modelos es una parte esencial de MDA, la mayoría de las investigaciones se centran en las transformaciones de modelos PIM a PSM. El enfoque propuesto por Zhang *et al.* utiliza el *modelo de características* para estructurar los requisitos en el nivel CIM y la *arquitectura software* (que consiste en un conjunto de componentes e interacciones entre componentes) para organizar los elementos a nivel PIM. Los autores definen un proceso disciplinado, pero no automático, para construir arquitecturas software en base al modelo de características utilizando las *responsabilidades* como el conector entre características y componentes. La idea es que generalmente una característica puede ser implementada en base a un conjunto de responsabilidades y que un componente puede ser visto como un contenedor de responsabilidades, puesto que a un componente se le puede asignar varias responsabilidades. De esta forma, el proceso para trazar las características a los componentes está formado básicamente por dos pasos: 1) encontrar una forma programable de implementar los requisitos denotados por cada característica, esto es, descomponer cada característica en un conjunto de responsabilidades que la implementa, y 2) asignar responsabilidades a componentes. El análisis de las interacciones entre las características se utiliza como guía para definir las interacciones entre componentes.

Los trabajos de Ponsard *et al.* [278] y Castro *et al.* [50] abordan la derivación de las arquitecturas software a partir de los modelos de objetivos. Ponsard *et al.* [278] proponen un mapeo entre los modelos: KAOS [70] y AADL (*Architecture Analysis and Design Language*) [85] en el dominio de los sistemas embebidos. El modelo de requisitos de KAOS está compuesto por cuatro submodelos: el modelo de objetivos, el modelo de objetos, el modelo de agentes y el modelo de operación. Por su parte, el modelo AADL incluye un nivel estructural, donde se definen componentes y los conectores que representan los flujos de datos y control entre componentes, y un nivel de comportamiento expresado como una secuencia de llamadas a subprogramas. De acuerdo a esta estructura, el proceso de mapeo está compuesto por tres pasos: 1) producir la descripción estructural AADL a partir, fundamentalmente, del modelo de agentes y complementado por los modelos de objetivos y de objetos; 2) obtener el modelo de comportamiento AADL a partir del modelo de operación y 3) refinar la arquitectura inicial inyectando los requisitos no funcionales en base a una librería de patrones. En

cuanto al trabajo de Castro *et al.* [50], los autores proponen STREAM (*Strategy for Transition between REquirements models and Architectural Models*) un proceso para transformar el modelo de requisitos inicial, expresado utilizando el lenguaje de modelado orientado a objetivos i* [365], en un modelo inicial de la arquitectura, expresado utilizando el lenguaje Acme [101]. El enfoque consiste en: 1) un conjunto de reglas de transformación para preparar los modelos de requisitos de manera que se balancee las responsabilidades de los actores software, delegándolas a otros actores; 2) un conjunto de reglas de transformación para generar un conjunto de soluciones de diseño arquitecturales en Acme; y 3) un proceso sistemático para asistir al desarrollador en la selección de la arquitectura que mejor se ajuste a los requisitos no funcionales.

Por otro lado, de acuerdo con la propuesta MDA, a nivel CIM se modela el negocio. Algunos autores reconocen la importancia de modelar los procesos de negocio (BPM, *Business Process Modeling*) como paso previo para la especificación de requisitos. En esta línea de modelado del negocio, aunque González *et al.* [109] no encuadran su propuesta explícitamente en el contexto de MDA sino en un proceso de DSDM general, los autores proponen una técnica sistemática para derivar diagramas de clase a partir de los modelos de proceso de negocio. Para ello integran los métodos de Análisis de Comunicación [83] (método de IR y BPM orientado a la comunicación que incluye el Diagrama de Eventos Comunicativos y las Plantillas de Especificación de Eventos) y OO-Method [265] (un método de desarrollo de software orientado a objetos y dirigido por modelos que incluye los modelos de Objetos -diagrama de clases extendido-Funcional, Dinámico y de Presentación para el modelado conceptual). Tanto el Diagrama de Eventos Comunicativos como las Plantillas de Especificación de Eventos contienen información relevante para la derivación del Modelo de Objetos. Aunque las guías de derivación se presentan en forma de reglas, éstas se aplican de forma manual, los autores consideran que es el primer paso para la integración del BPM en un enfoque completo de DSDM.

Trabajos incluidos en la RSL

Los trabajos que finalmente fueron incluidos en la RSL aplicando los criterios de inclusión y exclusión definidos en el protocolo (Sección 3.3.1.3) son los que se muestran en la Tabla 25 y la Tabla 26.

El resultado más significativo, y el que motiva el interés de la investigación que se presenta en esta tesis doctoral, es que, como esperábamos, la mayoría de las propuestas no tienen en cuenta los requisitos textuales. En concreto, el 71,4 % de los trabajos no tienen en cuenta la conexión (ya sea implícita o explícita) de los requisitos textuales con el resto de artefactos del desarrollo. En consecuencia, una vez que se elicitó la especificación de requisitos, es responsabilidad del analista la definición del modelo de requisitos que se ajuste a dicha especificación. El no enlazar los requisitos especificados con los modelos complica la validación de la completitud de los requisitos en los modelos, esto es, comprobar que todos los requisitos han sido tenidos en cuenta durante el modelado. Sólo [73, 157] tienen en cuenta la generación de diagramas de clases a partir de modelos de requisitos basados en lenguaje natural. Mientras que, en el trabajo presentado por Zhu *et al.* [373] consideran la transformación de requisitos textuales al lenguaje formal LOTOS, pero dicha transformación no forma parte de la propuesta.

Fuente	Autores [ref.]	Año	Req. textuales	Modelo origen	Modelo destino	MDA	Tipo	Lenguaje	Herramienta	Validación	Dominio	Id
IEEE	Escalona y Aragón [78]	2008	N	Requisitos definidos mediante patrones conforme al MM conforme al MM WebRE	Modelos UWEE (navegacional, contenido, interfaz)	N	A	QVT	NDT-Tool (j2EE) (ed+t)	11 Proyectos Reales	web	Escalona08
SD	Valderás y Pelechano [343]	2005 2009	N	Modelo de requisitos basado en tareas	Modelo navegacional OOWS	N	A	Graph transformations	AGG (t)	Desarrollo de varias aplicaciones web de tamaño pequeño-medio	web	Valderás09
SD	De Castro <i>et al.</i> [48]	2011	N	Modelo de Valor (conforme a la notación evalue) Modelo de Procesos de Negocio (conforme a BPMN)	- Modelo de casos de uso - Modelo extendido de casos de uso - Modelo de Procesos de Servicios - Modelo de Composición de Servicios	S	SA	ATL	Basada en Eclipse (ed+t)	Web real para gestión de imágenes médicas	SOA	DeCastro11
IEEE	Lazarte <i>et al.</i> [179]	2010	N	Modelo de Proceso de Negocio Colaborativo (conforme al lenguaje UPColBPiP)	Diagramas de Procesos de Negocio (BPMN)	S	A	ATL/QVT	Basada en Eclipse (ed+t)	Caso de estudio sobre la gestión integrada de una cadena de abastecimiento	Colaboraciones B2B	Lazarte10
ACM	Mosser <i>et al.</i> [235]	2011	N	Mapas de Casos de Uso orientados a Aspectos (conformes a AoURN)	Modelos de Procesos de negocio ADORE	N	SA	Lenguaje de programación	jUCMNav y ADORE (ed s)	Caso de estudio PicWeb	SOA	Mosser11
IEEE	Zhang y Jiang [368]	2008	N	VIP Framework: BYM (<i>Business Value Model</i>), BIM (<i>Business Information Model</i>) y BPM (<i>Business Process Model</i>)	BPEL (<i>Business Process Execution Language for Web Services</i>) model	S	A	Lenguaje de programación	Entorno IPRO (eds) + t	Caso de Estudio de una Agencia de Viajes electrónica	Servicios Web	Zhang08

Tabla 25 Trabajos seleccionados (1/2). Leyenda: S (SI) y N (No); A (Automático) y SA (Semi-Automático); ed (editor) y t (transformaciones).

Fuente	Autores [ref.]	Año	Req. textuales	Modelo origen	Modelo destino	MDA	Tipo	Lenguaje	Herramienta	Validación	Dominio	Id
IEEE	Escalona y Aragón [76]	2008	N	Requisitos definidos mediante patrones conforme al MM WebRE	Modelos UWE (navegacional, contenido, interfaz)	N	A	QVT	NDT-Tool (J2EE) (ed+1)	11 Proyectos Reales	web	Escalona08
SD	Valderás y Pelechano [343]	2005	N	Modelo de requisitos basado en tareas	Modelo navegacional OOWS	N	A	Graph transformations	AGG (1)	Desarrollo de varias aplicaciones web de tamaño pequeño-medio	web	Valderás09
SD	De Castro <i>et al.</i> [48]	2011	N	Modelo de Valor (conforme a la notación ervalue) Modelo de Procesos de Negocio (conforme a BPMN)	- Modelo de casos de uso - Modelo extendido de casos de uso - Modelo de Procesos de Servicios - Modelo de Composición de Servicios	S	SA	ATL	Basada en Eclipse (ed+1)	Web real para gestión de imágenes médicas	SOA	DeCastro11
IEEE	Lazarte <i>et al.</i> [179]	2010	N	Modelo de Proceso de Negocio Colaborativo (conforme al lenguaje UPOIBPIP)	Diagramas de Procesos de Negocio (BPMN)	S	A	ATL/QVT	Basada en Eclipse (ed+1)	Caso de estudio sobre la gestión integrada de una cadena de abastecimiento	Colaboraciones B2B	Lazarte10
ACM	Mosser <i>et al.</i> [235]	2011	N	Mapas de Casos de Uso orientados a Aspectos (conformes a AoURN)	Modelos de Procesos de negocio ADORE	N	SA	Lenguaje de programación	JUCMINav y ADOPRE (ed s)	Caso de estudio P4cWeb	SOA	Mosser11
IEEE	Zhang y Jiang [368]	2008	N	VIP Framework: BVM (Business Value Model), BIM (Business Information Model) y BPM (Business Process Model)	BPBL (Business Process Execution Language for Web Services) model	S	A	Lenguaje de programación	Entorno IPRO (eds) + t	Caso de Estudio de una Agencia de Viajes electrónica	Servicios Web	Zhang08

Tabla 25 Trabajos seleccionados (1/2). Leyenda: S (Si) y N (No); A (Automático) y SA (Semi-Automático); ed (editor) y t (transformaciones).

El único trabajo que tiene en cuenta el establecimiento de relaciones de traza explícitas entre los modelos de requisitos basados en lenguaje natural y el resto de modelos de desarrollo es el propuesto por Colombo *et al.* [54]. Este trabajo utiliza el lenguaje SysML [257] el cual incluye los diagramas de requisitos para la representación gráfica de los requisitos textuales y sus relaciones.

En cuanto a los modelos origen y destino de la transformación, casi son tantos los modelos como los trabajos incluidos en la RSL. Sólo parece encontrarse ciertas coincidencias en los modelos utilizados si analizamos las propuestas por dominios de aplicación, aunque, si bien pueden coincidir en el modelo obtenido mediante el proceso de transformación, el lenguaje utilizado para representar dichos modelos en la mayoría de los casos es diferente. Por ejemplo, en las propuestas relacionadas con el desarrollo de aplicaciones web el interés se centra en la generación de modelos navegaciones (conformes a UWE [81] o OOWS [346]) a partir de los modelos de requisitos expresados utilizando también diferentes modelos (conforme al metamodelo WebRE [81] o basados en tareas [346]). En el desarrollo de aplicaciones orientadas a servicios (SOA) juega un papel importante el modelado de los procesos de negocios que se pueden representar de acuerdo a la notación estándar BPMN [183] o alguna más específica como ADORE [238].

Si bien el enfoque MDA es el estándar de DSDM propuesto por el OMG, está repartido de forma bastante igualitaria los trabajos que se proponen en el marco de MDA (57,2%) y aquellos que siguen un proceso de DSDM general (42,8%). Resulta curioso, que aunque el OMG también propone QVT [259] como lenguaje de transformación de modelos, este no es el lenguaje que adoptan todas las propuestas presentadas en el contexto de MDA y, ni siquiera es el lenguaje de transformación de modelos más utilizado. De forma global, el lenguaje de transformación de modelos más utilizado es ATL con un porcentaje del 42,8%. La distribución de los lenguajes de programación utilizados en función del enfoque del proceso de desarrollo (en el contexto de MDA o un enfoque general de DSDM) se puede ver en la Figura 12. Nótese que la suma de todos los trabajos en la Figura 12 es 15, en lugar de 14 que representa el total de trabajos incluidos en la RSL (Tabla 25 y Tabla 26). Esto es debido a que en la propuesta de Lazarte *et al.* no se dice con exactitud el lenguaje utilizado para las transformaciones, sino que al presentar la arquitectura de la herramienta de soporte incluyen tanto ATL como QVT. Por tanto, hemos contado la propuesta de Lazarte *et al.* tanto en los trabajos que utilizan ATL como en la suma de los trabajos que utilizan QVT.

Dado que los trabajos incluidos en la RSL definen procesos de transformación automáticos o semi-automáticos guiados por reglas de transformación definidas en algún lenguaje de programación o de transformación de modelos (ver Figura 12), es de suponer que el proceso está soportado por alguna herramienta. Esta información queda implícita en la propuesta de Mazón *et al.* [209], donde se definen los perfiles UML para representar los modelos origen y destino y las reglas de transformación QVT, pero no se detalla la herramienta utilizada para dar soporte al proceso. En algunos casos [238, 302, 336], los autores sólo indican explícitamente los editores de modelos utilizados, quedando implícito el motor de transformación utilizado de acuerdo al lenguaje de transformación. En el trabajo de Sánchez *et al.* [302] no se concreta el editor de modelos puesto que sostienen que se puede utilizar cualquier herramienta para UML 2.0 que disponga de la utilidad de importación/exportación de modelos XMI. Otras veces ocurre al contrario, y la herramienta propuesta por los autores se limita únicamente a dar

soporte a la transformación de modelos [73, 346]. No obstante, Valderás y Pelechano [346] aunque sólo señalan el uso de la herramienta de libre distribución AGG (*Attributed Graph Grammar*) para las transformaciones de grafos, esto constituye sólo una parte de la propuesta global que se detalla en la tesis doctoral de Valderás [345] y que incluye una herramienta para la definición de requisitos y otra para la generación de código. El resto de trabajos (57,2%) proporcionan información tanto del soporte para la edición de modelos como para el proceso de transformación, aunque no siempre queda claro si estos dos procesos tienen lugar en un entorno integrado. No cabe duda de esta integración en las propuestas desarrolladas en el entorno Eclipse [50, 54, 183] y las herramientas NDT-tool [81] y BPSTool [295]. De estas dos últimas cabe destacar que han sido probadas en casos de estudios reales.

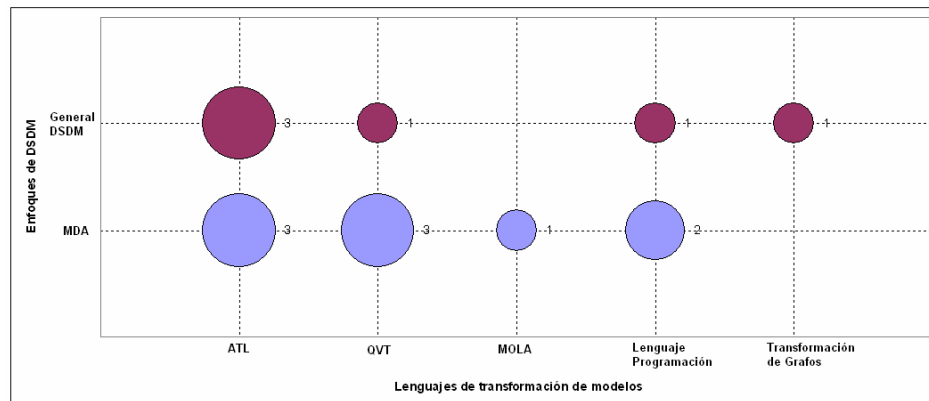


Figura 12 Lenguajes de transformación de modelos utilizados en cada uno de los enfoques de DSDM.

La mayoría de los trabajos (71,4%) no han sido validados utilizando casos de estudio reales o algún otro método de validación empírico. Las únicas excepciones son: 1) la propuesta de Escalona y Aragón [81] que ha sido aplicada a 11 proyectos reales y además, el proceso NDT ha sido adoptado por varias compañías como metodología de especificación de requisitos, 2) la aproximación de Valderás y Pelechano [346] que se ha utilizado para el desarrollo de varias aplicaciones web de pequeña-mediana envergadura (aunque los autores no detallan ni el número exacto ni la temática de dichas aplicaciones las hemos considerado como aplicaciones reales), 3) el enfoque de De Castro *et al.* [50] que se ha probado en diferentes casos de estudio incluyendo el desarrollo de una web para el procesamiento y gestión de imágenes médicas y 4) la propuesta de Rodríguez *et al.* [295] que no sólo se ha desarrollado utilizando el método de investigación en acción, sino que ha sido validado en el marco de una cooperativa cuyo principal proceso de negocio es la distribución de electricidad a áreas rurales.

Sólo tres de las propuestas se han desarrollado con propósito general (21,4%) [73, 157, 302], el resto se han definido para el desarrollo de un tipo concreto de aplicaciones (aplicaciones web [81, 346], aplicaciones orientadas a servicios [50, 183, 238, 371], sistemas de control [54], sistemas de tiempo real [373] y almacenes de datos [209]) o para cubrir un aspecto transversal del desarrollo como es la seguridad [295, 336].

3.3.2.2. Evaluación de la calidad

Los datos asociados a cada uno de los trabajos correspondientes a los criterios de calidad establecidos en la Sección 3.3.1.4 son los que se muestran en la Tabla 27.

	CC1	CC2	CC3	CC4	CC5		CC6 Scopus	CC6 GS
Escalona08	N	T	A	11 CER	IEEE TRANSACTIONS ON SOFTWARE ENGINEERING	Q1	23	44
Valderas09	N	T	A	CER	INFORMATION AND SOFTWARE TECHNOLOGY ²	Q1	7	13
DeCastro11	N	P	S-A	CER	INFORMATION AND SOFTWARE TECHNOLOGY	Q1	3	7
Jiang12	N	P	A	CEA	LECTURE NOTES IN ELECTRICAL ENGINEERING	no en JCR	0	0
Lazarte10	N	T	A	CEA	<i>IEEE International Enterprise Distributed Object Computing Workshop</i>	CORE B	1	1
Mosser11	N	P	S-A	CEA	<i>10th International Conference on Aspect-Oriented Software Development</i>	CORE A	1	6
Zhang08	N	P	A	CEA	<i>1st IEEE International Workshop on Semantic Computing and Systems</i>	no en CORE	5	9
Debnath08	S	P	A	CEA	<i>International Conference on Information Technology: New Generations</i>	no en CORE	6	9
Kalnins10	S	T	A	CEA	LU (Latvia University) Press	no en JCR	---	---
Sánchez10	N	P	S-A	3 CEA	INFORMATION AND SOFTWARE TECHNOLOGY	Q1	7	12
Rodríguez10	N	P	A	I-A CER	INFORMATION AND SOFTWARE TECHNOLOGY	Q1	3	4
Sun08	N	P	A	CEA	<i>International Conference on Quality Software</i>	CORE B	1	5
Colombo12	S	P	A	CEA	THE JOURNAL OF SYSTEMS AND SOFTWARE	Q2	0	0
Mazón07	N	P	A	CEA	<i>ER</i>	CORE A	3	51
Zhu10	N	P	A	CEA	<i>International Conference on Quality Software</i>	CORE B	0	1

Tabla 27 Criterios de calidad de los trabajos seleccionados. Los acrónimos utilizados son:
1) Conexión con requisitos textuales: S (Sí) y N (No); **2) Integración en el DSDM:** T (Total) o P (Parcial); **3) Grado de Automatización:** A (Automático) y S-A (Semi-Automático); **4) Validación:** CEA (Caso de Estudio Académico), CER (Caso de Estudio Real) o I-A (Investigación en Acción).

Atendiendo a estos datos, la valoración de cada trabajo, aplicando los criterios de calidad, es la que se muestra en la Tabla 28. Podemos ver que los dos trabajos relacionados con el desarrollo de aplicaciones web [81, 346] son los que obtienen una evaluación más alta. Estos estudios consiguen la máxima puntuación en todos los criterios establecidos salvo en la conexión con la especificación de requisitos textual. En estos enfoques, como veremos en la sección siguiente (Sección 3.3.3 Discusión) el modelo de requisitos contiene la información necesaria para poder generar los modelos navegacionales pero la creación de estos modelos depende de la experiencia y la habilidad del analista que debe ser capaz de crear los modelos que se correspondan con la especificación de requisitos del usuario.

² En la última revisión de 2011 se ha modificado el índice de impacto de esta revista, no obstante mantenemos la consideración de Q1 que era el valor que tenía en el momento del estudio.

	CC1	CC2	CC3	CC4	CC5	CC6	VALOR
Escalona08	-1	1	1	1	2	2	6
Valderas09	-1	1	1	1	2	2	6
DeCastro11	-1	0	0	1	2	1	3
Jiang12	-1	0	1	0	-1	0	-1
Lazarte10	-1	1	1	0	0	1	2
Mosser11	-1	0	0	0	1	1	1
Zhang08	-1	0	1	0	-1	2	1
Debnath08	1	0	1	0	-1	2	3
Kalnins10	1	1	1	0	0	0	3
Sánchez10	-1	0	0	0	2	2	3
Rodríguez10	-1	0	1	1	2	1	4
Sun08	-1	0	1	0	0	1	1
Colombo12	1	0	1	0	1	0	3
Mazón07	-1	0	1	0	1	2	3
Zhu10	1	0	1	0	0	1	3

Tabla 28 Evaluación de la calidad de cada trabajo seleccionado.

Merece la pena también destacar el trabajo de Rodríguez *et al.* [294] que sigue en valoración a los anteriores y lo que le diferencia es que no proporcionan un enfoque total de DSDM sino únicamente la transformación de procesos de negocio seguros (nivel CIM según la terminología de MDA) a diagramas de clase y diagramas de casos de uso (nivel PIM, de acuerdo a MDA). Por último, cabe señalar que sólo el trabajo de Jiang and Ding [146] alcanza una puntuación negativa y por ese motivo ha sido excluido de la RSL.

3.3.3. Discusión

Después de haber recogido los datos y de haber analizado los trabajos seleccionados, a continuación se responde a las cuestiones de la investigación presentadas en la Sección 3.3.1.1. Dado que las dos preguntas de investigación están fuertemente relacionadas, no vamos a separar la discusión de los trabajos en función de los modelos origen (PI1) y modelos destino (PI2), sino que vamos a analizar cada una de las propuestas de forma conjunta. Hemos organizado la discusión por dominios de aplicación.

3.3.3.1. Enfoques de propósito general

Los trabajos de Debnath *et al.* [73] y Kalnins *et al.* [157] son los únicos que hemos encontrado que proponen derivar modelos de análisis, en concreto modelos de clases, a partir de modelos de requisitos basados en lenguaje natural. Por una parte, Debnath *et al.* [73] utiliza como punto de partida para la transformación los modelos de requisitos: LEL (*Language Extended Lexicon*) y Escenarios [187], que describen el vocabulario y el comportamiento del sistema, respectivamente. Aplicando las reglas de transformación (definidas formalmente en [188]) se pueden generar clases, atributos, métodos y

relaciones, teniendo en cuenta la estructura y la semántica de los modelos de requisitos. En este caso, el proceso de transformación es más o menos directo, dado que la estructura de los símbolos definidos en el modelo LEL está muy próxima a la definición de una clase. Cada símbolo especifica unas propiedades y su comportamiento. Aplicando las reglas de transformación, cada símbolo se corresponde con una clase en el diagrama de clases, todos los nombres de la sección de propiedades se convierten en atributos y todas las entradas de la sección de comportamiento se traducen en métodos de la clase. Tras el proceso de transformación, es necesario que el analista revise los modelos generados porque, por ejemplo, es muy probable que contengan muchos atributos que no son necesarios.

Por otro lado, mientras que la propuesta de Debnath *et al.* [73] se centra en las primeras etapas del desarrollo, la aproximación de Kalnins *et al.* [157] abarca todo el proceso, desde los requisitos hasta el código. El punto de partida del proceso propuesto por Kalnins *et al.*, es la especificación de requisitos semi-formal que describe el comportamiento del sistema, utilizando un lenguaje natural controlado (RSL, *Requirements Specification Language*). A partir de aquí, los autores proponen una cadena de modelos (análisis, arquitectura y diseño detallado) hasta obtener un prototipo del sistema, utilizando el lenguaje de transformación MOLA [155]. Vamos a centrarnos en la solución propuesta para la transformación de los modelos de requisitos.

El RSL utiliza casos de uso para definir de forma precisa los requisitos acerca del comportamiento del sistema. Cada caso de uso se detalla mediante uno o más escenarios descritos utilizando sentencias en un lenguaje natural controlado [323]. Estos escenarios se pueden representar gráficamente utilizando diagramas de actividades, en los que los nodos son *sentencias SVO(O)*, que consisten en *Sujeto*, *Verbo*, *Objeto directo* y, opcionalmente, *Objeto indirecto*. Estas sentencias expresan las acciones que se van a ejecutar en el escenario. Además, para la descripción de estas sentencias se ha definido un conjunto de palabras claves con semántica predefinida, por ejemplo, las palabras claves *show*, *build*, *add* y *remove*, se pueden utilizar para especificar los verbos de las sentencias. Para completar el modelo de requisitos se tienen que definir todos los conceptos (*notions*) relevantes del dominio.

A partir de la definición de los conceptos del dominio, teniendo en cuenta también algunos elementos de los escenarios, se puede generar de una forma sencilla el principal modelo de análisis, el *Modelo de Dominio (MD)*. El MD es un modelo de clases conceptuales del sistema que se va a construir. Cada clase puede tener un estereotipo para distinguir diferentes tipos de conceptos, de acuerdo al Perfil de Análisis predefinido, por ejemplo, el estereotipo «entity» representa un concepto persistente. El uso de las palabras claves facilita la transformación de los requisitos al modelo del dominio. Este es un aspecto interesante que hemos tenido en cuenta en nuestra propuesta. El uso de palabras claves, como forma de restringir la especificación de los requisitos, parece una solución viable para poder transformar los requisitos, basados en lenguaje natural, en otros modelos de análisis. Así, en nuestra propuesta, el uso de parámetros en la especificación de requisitos nos va ayudar en el proceso de transformación de modelos.

El trabajo de Sánchez *et al.* [302] se centra en los requisitos no funcionales que son gestionados como *aspectos*, dado que son cuestiones transversales (*crosscutting concerns*) que se pueden describir de manera independiente al sistema que los incorpora.

Por ejemplo, los requisitos de seguridad, se pueden describir asumiendo que algún elemento del sistema debe ser seguro. Además, a partir de estos requisitos se pueden derivar automáticamente una solución arquitectural mediante transformaciones de modelos, lo que contribuye a mejorar la trazabilidad sistemática de los requisitos con la arquitectura que los satisface.

De acuerdo con Sánchez *et al.* [302] la generación del modelo de arquitectura que se corresponde con los requisitos asociados a alguna cuestión transversal, con frecuencia consiste en aplicar unos patrones predefinidos que representan una solución que satisface los requisitos. Estos patrones especifican un conjunto de pasos y operaciones, que la mayoría de las veces se pueden definir de manera algorítmica. Por tanto, los patrones se van a encapsular en las transformaciones de modelos, de manera que permitan generar automáticamente un modelo de diseño de la arquitectura orientada a aspectos, tomando como entrada los modelos de requisitos orientados a aspectos. Además, puede haber más de un patrón que satisfaga un mismo requisito. Por ejemplo, se pueden utilizar distintas técnicas para resolver la integridad de mensajes. Por tanto, es responsabilidad del arquitecto del software seleccionar la transformación que sea más adecuada en función de las necesidades de los stakeholders. Podemos extraer como conclusión que, si existe un patrón predefinido que los expertos siguen para la generación de los modelos a partir de los requisitos, este conocimiento se pueden encapsular en la definición de la transformación de modelos.

Los modelos de requisitos se basan en los escenarios de UML 2.0 y son conformes al Perfil UML ASM (*Aspectual Scenario Modelling*), mientras que los modelos de salida de la transformación, los modelos de la arquitectura, son modelos conformes al Perfil UML CAM (*Component Aspect Model*) [270]. En los modelos expresados en ASM se puede seleccionar el patrón que se va a utilizar en el proceso de transformación. Las transformaciones de modelos se han especificado utilizando el lenguaje QVT e implementado utilizando ATL.

Una cuestión interesante de la propuesta de Sánchez *et al.* es que tienen en cuenta la reutilización de requisitos, un aspecto central en la propuesta que vamos a desarrollar en esta tesis doctoral. Considerando que muchos de los requisitos que especifican aspectos (como integridad, autenticidad, persistencia y encriptación) reaparecen en diferentes aplicaciones, se pueden recoger en una biblioteca de modelos para ser reutilizados, creando, de este modo, un catálogo de requisitos aspectuales reutilizables.

3.3.3.2. Desarrollo de aplicaciones Web

La aplicación de la tecnología del DSDM a la ingeniería Web recibe el nombre de Ingeniería Web Dirigida por Modelos (MDWE, *Model-Driven Web Engineering*). MDWE propone la representación de conceptos utilizando metamodelos y el soporte al proceso de desarrollo mediante un conjunto de transformaciones de modelos. En este contexto, Escalona y Aragón [81] presentan NDT, una metodología completamente basada en MDWE, para las primeras etapas del desarrollo. NDT propone una fase de requisitos detallada, a partir de la cual se puede obtener sistemáticamente los modelos de análisis de los sistemas Web, garantizando, de esta forma, la consistencia entre los requisitos y los conceptos de la fase de análisis. El objetivo no es proporcionar un enfoque metodológico completo para el Web, sino generar, y garantizar la calidad de,

los modelos de análisis que pueden servir como entrada para los procesos de desarrollo de otras metodologías Web. Este es nuestro objetivo también en esta tesis doctoral, proporcionar un método de IR dirigido por modelos basado en reutilización que genere los modelos conformes a un DSL, que luego puedan seguir un enfoque de modelado específico de dominio.

El metamodelo propuesto en NDT para la fase de requisitos es una *extensión de WebRE* [82]. Aunque los conceptos definidos en el metamodelo podrían representarse utilizando artefactos de UML, la técnica elegida en NDT para la definición de los requisitos son los patrones. Los patrones son tablas con campos específicos para recoger la información que requiere cada concepto definido en el metamodelo. Los requisitos se clasifican atendiendo a su naturaleza en: requisitos de almacenamiento de información, requisitos de actores, requisitos funcionales, requisitos de interacción o requisitos no funcionales.

Una vez que los requisitos han sido validados, se pueden generar mediante *transformaciones modelo a modelo en QVT*, los modelos de análisis. Estos modelos de análisis no son nuevos, sino que utilizan experiencias previas basadas en los estándares. Los modelos generados son: el *modelo de contenido*, que es un diagrama de clases UML, el *modelo navegacional* y el *modelo de interfaz abstracta*, para los que asume el metamodelo UWE (*UML-based Web Engineering*) [172].

Valderás y Pelechano [346] también proponen la generación de *modelos navegacionales* a partir de *modelos de requisitos basados en tareas* mediante transformaciones m2m, en el marco del método de ingeniería Web OOWS [93]. OOWS es una extensión del método de producción del software OO denominado OO-Method [264], que propone diferentes modelos para describir los distintos aspectos de un sistema (diagrama de clases, modelo funcional, modelo dinámico). A estos modelos OOWS añade el modelo navegacional para describir los aspectos navegacionales de una aplicación Web.

Las transformaciones m2m se han implementado mediante *transformaciones de grafos* utilizando AGG [3]. Las transformaciones se definen como un conjunto de correspondencias (*mappings*) que toman abstracciones del modelo de requisitos basado en tareas como base e indican cómo se pueden soportar a nivel conceptual mediante las primitivas del modelo navegacional de OOWS. Las transformaciones de grafos se complementan con mecanismos de trazabilidad que permiten trazar los requisitos con el modelo navegacional derivado automáticamente.

Coincidimos con Valderás y Pelechano en la idea de generar los modelos de traza en la definición de las transformaciones de manera que se puedan ligar los requisitos con otros conceptos del desarrollo que se derivan a partir de ellos. Esta es la aproximación que vamos a seguir en esta tesis, generar los informes de trazabilidad después de ejecutar las transformaciones.

3.3.3.3. Aplicaciones orientadas a servicios

En el desarrollo orientado a servicios de los sistemas de información, es crucial alinear las especificaciones del negocio de alto nivel, que entiende los analistas o gestores del negocio, con las tecnologías de la información de un nivel inferior, utilizadas por los desarrolladores del software [51]. En esta línea, existen diferentes

trabajos que proponen el uso de las transformaciones de modelos para generar modelos de diseño a partir de los modelos que describen los requisitos del negocio en diferentes formatos.

El trabajo de Zhang y Jiang [371] presenta un enfoque basado en MDA, que permite adaptar rápidamente la arquitectura de una aplicación Web a nuevos requisitos del negocio. Para ello proponen analizar y modelar los requisitos del negocio, a nivel CIM, utilizando el marco de modelado del negocio VIP [26] que está compuesto por: BVM (*Business Value Model*), que describe proposiciones de valor para el negocio, BIM (*Business Information Model*), describe la arquitectura de la información de una aplicación Web, esto es, el intercambio de información entre el sistema Web y los actores, y BPM (*Business Process Model*), que especifica la funcionalidad de la aplicación Web. El marco VIP incluye la definición de los metamodelos para la descripción de cada uno de los modelos implementados como perfiles UML 2.0. Los modelos con la lógica del negocio (BIM y BPM) representan de manera precisa los requisitos del negocio, y pueden ser transformados en modelos de diseño expresados en BPEL (*Business Process Execution Language for Web Services*) [125], a nivel PIM, aplicando transformaciones basadas en Java de acuerdo con un repositorio de reglas de correspondencia (*mapping rules*). BPEL, es un lenguaje basado en WSDL (*Web Service Definition Language*), que permite especificar los procesos del negocio utilizando servicios Web para alcanzar sus objetivos.

En la propuesta de Zhang y Jiang [371] se utiliza la *metodología e³-value* [110] para la especificación del modelo BVM, a nivel de requisitos del negocio. Esta misma metodología es la que adoptan De Castro *et al.* [51], a nivel CIM, junto con BPMN [258], en el marco de un método para el desarrollo orientado a servicios de los sistemas de información (SOD-M). En la propuesta de De Castro *et al.*, el modelo de valor se utiliza para identificar los actores del negocio y las actividades que llevan a cabo que se espera que sean rentables, mientras que el modelo de procesos de negocio se emplea para entender y describir los procesos del negocio relacionados con el entorno en el que se va a utilizar el sistema que se va a construir. A nivel PIM, De Castro *et al.* utilizan un conjunto de DSLs que han desarrollado específicamente para el desarrollo orientado a servicios de los sistemas de información, que permiten representar los modelos de: 1) casos de uso, para representar los servicios del negocio que se van a implementar en el sistema; 2) casos de uso extendidos, para representar la funcionalidad (servicios) que requiere el sistema para llevar a cabo un servicio del negocio; 3) proceso de servicio, muestra el conjunto de actividades lógicamente relacionadas que necesitan ejecutarse en el sistema para llevar a cabo un servicio del negocio (diagrama de actividades); y 4) composición de servicio, que complementa el modelo anterior y representa el flujo de trabajo detallado necesario para llevar a cabo un servicio.

En el marco del método SOD-M, han definido las reglas (algunas de ellas parcialmente automáticas) para generar: el modelo de casos de uso a partir del modelo de valor, el modelo de casos de uso extendido a partir del modelo de procesos de negocio y el modelo de casos de uso, el modelo de proceso de servicio a partir del modelo de casos de uso extendido y el modelo de composición de servicio a partir del modelo de valor, el modelo de procesos de negocio y el modelo de proceso de servicio. En el momento de la publicación de este trabajo [51] sólo la regla de transformación del modelo valor al modelo de casos de uso estaba implementada, utilizando el lenguaje ATL. Los autores

señalan que están trabajando en la implementación del resto de reglas y en la generación de código desde los modelos SOD-M a diferentes plataformas de servicios Web, con lo que definirían un proceso completo DSDM desde los requisitos hasta el código, aunque los modelos de requisitos no estarían enlazados con la especificación textual de requisitos.

Mosser *et al.* [238] se centra en la aplicación de las técnicas del desarrollo de software orientado a aspectos a lo largo del ciclo de vida, en el contexto de arquitecturas orientadas a servicios (SOA). Los autores proponen un enfoque orientado a aspectos que comprende modelos de requisitos de propósito general y modelos de diseño de procesos de negocio como primer paso hacia un proceso de ingeniería del software dirigido por *asuntos (concerns)* iterativo más amplio, que asegura la encapsulación de asuntos transversales a lo largo de todas las fases del desarrollo.

Los modelos de requisitos orientados a aspectos se expresan en AoURN (*Aspect-oriented User Requirements Notation*) [240]. AoURN soporta la elicitación, análisis, especificación y validación de requisitos, en un marco de modelado orientado a aspectos, con la ayuda de modelos de objetivos y escenarios. El modelo de objetivos describe las razones para incluir un *asunto* en la aplicación y su impacto en los objetivos de alto nivel de los stakeholders. Por otro lado, los escenarios describen los actores y entidades estructurales del sistema que colaboran para realizar un *asunto*.

A partir de los modelos de requisitos, en concreto, a partir de los escenarios, se genera automáticamente, mediante transformaciones de modelos, modelos de procesos de negocio orientados a aspectos expresados con ADORE (*A meta-moDel suppOrting oRchestration Evolution*) [237]. Utilizando ADORE se pueden modelar los procesos de negocio completos como una orquestación de servicios.

Para el caso de los modelos de negocio colaborativos y los sistemas B2B (*Business to Business*) que los implementan, a nivel de requisitos, las empresas se centran en el diseño de *procesos de negocio colaborativos* (PNC), que definen la vista global de las interacciones entre las empresas para alcanzar un objetivo de negocio común. Las empresas también tienen que diseñar la integración de los PNC con sus procesos privados que los soportan. Mientras que a nivel tecnológico, las empresas se centran en la funcionalidad e interoperabilidad de sus sistemas de información B2B para ejecutar los PNC. A nivel de arquitectura, los sistemas B2B pueden utilizar una arquitectura SOA.

En esta línea, Lazarte *et al.* [183] proponen una metodología, que explota los beneficios del DSDM, e incluye guías, lenguajes, métodos, transformaciones de modelos y herramientas para dar soporte a la representación de requisitos del negocio, la definición de modelos de proceso colaborativos independientes de la tecnología, la derivación de procesos independientes de la tecnología y arquitecturas que la empresa necesita para soportar los procesos colaborativos y también la generación de soluciones específicas de la tecnología para cada empresa. La metodología consiste en cuatro fases: *Análisis del Negocio*, *Diseño de la Solución del Negocio*, *Diseño de la Solución de Arquitectura IT* (*Information Technology*) y *Diseño de la Solución Tecnológica*. Para que los modelos obtenidos como resultado de cada fase evolucionen de forma consistente, la metodología proporciona transformaciones de modelos automáticas y semiautomáticas. Nos vamos a centrar en la transformación entre las primeras etapas del desarrollo.

El resultado de la etapa de *Análisis del Negocio* es el *modelo de requisitos del negocio* (nivel CIM) que ayuda al analista del negocio a entender el problema e identificar los PNC que se van a diseñar en la etapa siguiente. El modelo de requisitos del negocio está formado por dos vistas: Vista de Colaboración y Vista de Procesos Colaborativos. La primera captura los participantes y sus relaciones de comunicación, así como los objetivos de negocio comunes que tienen que cumplirse por las partes. La segunda vista identifica los PNC necesarios para conseguir los objetivos del negocio comunes.

El lenguaje de modelado utilizado para los requisitos del negocio y los PNC es el perfil UML UPColBPIP (*UML Profile for Collaborative Business Processes based on Interaction Protocols*) [352]. Una vez definido el modelo de requisitos del negocio, se aplica una transformación de modelos automática que genera como salida el esqueleto de modelos PNC expresados también en UPColBPIP, que serán definidos en la tarea de diseño de PNC en la fase de *Diseño de la Solución del Negocio*. A partir de los modelos de PNC, se aplica también una transformación automática que genera los diagramas de procesos de negocio para cada una de las empresas que colaboran en el PNC de acuerdo a la notación BPMN (nivel PIM).

3.3.3.4 Seguridad

Aunque los trabajos de Rodríguez *et al.* [295] y Sun *et al.* [336] se encuentran enmarcados en el campo de la seguridad, las propuestas que presentan son bien diferentes. Por un lado, Rodríguez *et al.* presentan un enfoque general para la generación automática de modelos de análisis (diagramas de clases y diagrama de casos de uso) a partir de modelos de procesos de negocio seguros. Mientras que Sun *et al.* se fijan en un aspecto muy concreto de la seguridad como es la generación de configuraciones de control de acceso para componentes J2EE (*Java 2 Platform, Enterprise Edition*) a partir de la especificación de requisitos, como medio para conseguir la seguridad de la información.

En la propuesta de Rodríguez *et al.* [295], el modelo fuente de la transformación vertical (de CIM a PIM según la propuesta MDA) es conforme al metamodelo para procesos de negocio seguros. Este metamodelo se corresponde bien con el metamodelo UML 2.0-AD (*Activity Diagram*) o bien con BPMN-BPD (*Business Process Modeling Notation - Business Process Diagram*), que han sido extendidos para poder incluir los requisitos de seguridad en la definición de los procesos del negocio. Esta extensión recibe el nombre de BPsec (*Business Process security*). Se puede utilizar cualquiera de las dos notaciones para el modelado de los procesos del negocio puesto que se ha definido la correspondencia entre los conceptos que incluyen ambos modelos mediante una transformación horizontal m2m (modelos del mismo nivel de abstracción, el nivel CIM). El modelo fuente de la transformación es BPMN-BPD y el modelo destino es UML 2.0-AD.

El modelo UML 2.0-AD se utiliza para obtener los modelos de análisis o modelos de nivel PIM (de acuerdo a la terminología MDA). El proceso de transformación tiene lugar en varias etapas y el resultado final consiste en el *diagrama de clases y el diagrama de casos de uso UML 2.0* que incluyen la información de seguridad descrita en los requisitos. Sólo en la generación de los casos de uso es necesaria la intervención

del analista dado que las transformaciones no son suficientes para obtener un modelo de casos de uso preciso y seguro, y es necesario que se refine manualmente el modelo generado de forma automática. Todas las transformaciones se han definido en QVT.

Por su parte, la propuesta de Sun *et al.* [336] se encuadra en el desarrollo de sistemas empresariales a gran escala utilizando la tecnología de desarrollo basada en componentes. En este tipo de aplicaciones se adopta el modelo de Control de Accesos Basado en Roles [90] que se implementa mediante el servicio de seguridad J2EE [144]. En consecuencia, los Requisitos de Control de Acceso (RCA) de una organización deben especificar qué roles pueden hacer qué operaciones del negocio bajo algunas condiciones. De esta forma, el desarrollador puede implementar los RCAs definiendo Configuraciones de Control de Acceso (CCA) para los componentes de forma declarativa (asignando permisos para ejecutar métodos a cada rol identificado en la organización). Según los autores, la definición correcta de CCAs de forma manual y *ad-hoc* es casi imposible y es necesario proporcionar mecanismos para poder generarlas automáticamente de acuerdo con los RCAs.

El motor de transformación toma como entradas los RCAs y el grafo de llamadas de la aplicación J2EE y genera CCAs consistentes y completas. La especificación de RCAs se basa en requisitos funcionales, asignando operaciones del negocio a roles. En concreto, la propuesta ha adoptado el *Modelo de Características* para especificar RCAs [372] y la Arquitectura Software (modelo de componentes UML 2.0) para el grafo de llamadas [29]. Los métodos e interfaces de la arquitectura tienen atributos para poder registrar información adicional, en particular los CCAs. Estos CCAs definidos en la arquitectura se pueden transformar en aquellos interpretables por el servicio de seguridad J2EE. Por tanto, el resultado de la ejecución de las reglas de transformación, implementadas en ATL, es una copia del modelo de componentes que representa la arquitectura software con la configuración del control de accesos.

3.3.3.5. Miscelánea

Para terminar el estudio de los trabajos seleccionados queda únicamente discutir tres trabajos enmarcados en dominios de aplicación diferentes: sistemas de control [54], sistemas en tiempo real [373] y almacenes de datos [209].

El objetivo del trabajo de Colombo *et al.* [54] es encontrar mecanismos efectivos y automáticos para generar el diseño del sistema a partir del análisis y la especificación de requisitos. La propuesta se basa en el enfoque para el análisis de requisitos denominado *Problem Frames* (PF) [142]. De acuerdo a este enfoque, el sistema software se especifica como una “máquina” que interacciona con el mundo exterior. Este mundo representa el espacio del problema. Los problemas se modelan en términos de dominios, que interaccionan con la máquina mediante fenómenos compartidos. Para dar soporte a las actividades de modelado del enfoque *Problem Frames* se utiliza el lenguaje SysML [257]. Este mismo lenguaje se utiliza también para la fase de diseño, para permitir la trazabilidad y favorecer la integración de los artefactos modelados. En consecuencia, la aplicabilidad de la propuesta está limitada a los problemas donde se pueda aplicar la metodología PF, esto es, problemas con requisitos de usuario funcionales donde el comportamiento de la máquina se puede definir en términos de relaciones causales entre los fenómenos observados y controlados.

El proceso de análisis comienza identificando los aspectos estructurales del problema, esto es, identificando los componentes básicos del sistema y sus interfaces. Después, el problema se descompone de forma recursiva en sub-problemas, de manera que se pueda abordar cada problema independientemente. Una vez que se ha identificado y analizado cada subproblema, y que se ha definido una especificación máquina para cada uno de ellos (expresada mediante un diagrama de máquinas de estado), las descripciones se recomponen para contribuir a la solución del problema original. El resultado del proceso de análisis es un *modelo de requisitos* elaborado manualmente, utilizando el Diagrama de Requisitos de SysML, que define las características del entorno del problema en lenguaje natural. Esta descripción informal inicial se refina en una más rigurosa y estructurada utilizando *diagramas SysML de estructura (diagramas de bloques y parámetros) y comportamiento (diagrama de máquina de estados y actividades)*.

Las decisiones de composición y descomposición tomadas durante el proceso de análisis, y los artefactos obtenidos como resultado se pueden utilizar para generar el primer modelo de diseño de acuerdo al patrón arquitectónico *blackboard* [46]. La generación se realiza de forma automática mediante transformaciones de modelos *implementadas en ATL*. A nivel de diseño, los modelos generados de manera incremental son *diagramas de bloques SysML*, que se organizan de acuerdo con el estilo definido en el patrón arquitectónico *blackboard*. El comportamiento de cada bloque se describe en términos de actividades asignadas a estos bloques.

La aproximación de Colombo *et al.* pone de relieve las ventajas de modelar las especificaciones de requisitos en lenguaje natural para poder enlazarlas con el resto de artefactos del desarrollo. Si bien los autores apuntan a que esta representación informal no se puede utilizar para derivar los artefactos del diseño, nuestra hipótesis de partida para el trabajo que vamos a desarrollar en esta tesis doctoral, es que los modelos de requisitos basados en lenguaje natural pueden utilizarse para derivar otros modelos si se utiliza un lenguaje natural restringido, como proponen Kalnins *et al.* [157].

Zhu *et al.* [373] abordan el problema de la ambigüedad e imprecisión de los modelos de la arquitectura software que se derivan a partir de especificaciones de requisitos en lenguaje natural. Para solucionar este problema, los autores proponen el uso de un lenguaje de especificación formal, LOTOS (*Language Of Temporal Ordering Specification*) [137], para la especificación de requisitos de los sistemas en tiempo real, de manera que se pueda generar, a partir de ella, los modelos de la arquitectura software (AS) expresados en UML-RT (*UML for RealTime*) [313]. En consecuencia, las especificaciones en lenguaje natural se transforman en especificaciones formales en LOTOS, y estas a su vez se transforman en los modelos AS en UML-RT. Los autores asumen en el artículo que la primera parte del proceso de transformación está hecha, y se centran en las transformaciones desde la especificación formal a los modelos AS.

Los modelos de la especificación formal y UML-RT son conformes a sendos metamodelos definidos en la plataforma AMMA (*Atlas Model Management Architecture*). A partir de estos modelos se definen las transformaciones en ATL que establecen la correspondencia entre los conceptos del metamodelo LOTOS y el metamodelo de UML-RT.

Mazón *et al.* [209] proponen el uso del marco de modelado i* [365, 366] y de MDA para describir: 1) cómo modelar objetivos y requisitos de información para almacenes de

datos, que permitan comprender las necesidades de información; y 2) cómo derivar un modelo conceptual multidimensional, que proporcione la información necesaria para soportar el proceso de toma de decisiones. Así, los autores proponen comenzar el desarrollo por una etapa de análisis de requisitos siguiendo el enfoque GORE (*Goal Oriented Requirements Engineering*), para modelar los requisitos de información de los implicados en la toma de decisiones, para después, derivar un modelo conceptual multidimensional adecuado a partir de ellos.

Los modelos de requisitos (modelos a nivel CIM según MDA) se representan mediante modelos i* conformes a un perfil UML, definido por los autores, que adapta el marco de modelado i* al dominio de los almacenes de datos. Aplicando varias reglas de transformación QVT, los modelos de requisitos se transforman en modelo multidimensional (a nivel PIM según MDA) expresados de acuerdo al Perfil UML para el modelado multidimensional. Este perfil contiene los estereotipos necesarios para representar las principales propiedades de un modelo multidimensional a nivel conceptual mediante un diagrama de clases UML en el cual la información se organiza claramente en hechos (*facts*) y dimensiones (*dimensions*). El modelo PIM generado soporta los objetivos que el responsable de la toma de decisiones planea alcanzar con el almacén de datos.

3.3.3.6. Limitaciones del estudio

Las limitaciones del estudio pueden derivarse de la rigidez de las cadenas de búsqueda utilizadas. Hemos utilizado cadenas fijas como “*requirements model*” o “*model transformation*” en lugar de expresiones equivalentes como “*model of requirements*” o “*transformation of requirements*” porque consideramos que las primeras son las expresiones comúnmente aceptadas y las más utilizadas en los artículos relacionados con la temática del DSDM. Además, consideramos que dadas las restricciones de los motores de búsqueda el uso de cadenas simples (sin muchos conectores lógicos) proporciona resultados más fiables. A la vista de los trabajos relacionados de muchos de los trabajos seleccionados, consideramos que los resultados obtenidos son representativos en relación a las preguntas de investigación objeto de esta RSL.

3.3.4. Conclusiones

La conclusión principal que se deriva de esta RSL y que valida las investigaciones que configuran esta tesis doctoral, es que son escasos los trabajos que tienen en cuenta los requisitos textuales en los primeros modelos del desarrollo de manera que puedan integrarse en un enfoque de DSDM. En aquellos enfoques donde se propone la transformación de requisitos textuales a otros modelos de desarrollo se generan los modelos de clases dado que los requisitos se definen de manera controlada pudiéndose derivar con relativa facilidad los conceptos y las relaciones implicadas en la especificación de requisitos. Sólo en uno de los enfoques, el basado en SysML, se utilizan los modelos de requisitos en los que se representan gráficamente los requisitos textuales y las relaciones entre ellos, pudiéndose trazar explícitamente estos requisitos con el resto de artefactos de desarrollo. El objetivo de esta tesis doctoral es combinar ambos enfoques, de manera que sea posible definir modelos de requisitos donde se representen

gráficamente los requisitos textuales y las relaciones de trazas entre ellos y al mismo tiempo sea posible generar otros modelos del desarrollo a partir de estas especificaciones de requisitos.

Algunas conclusiones que podemos extraer de los trabajos y que nos pueden servir para desarrollar la propuesta de esta tesis doctoral son:

1. Modelar las especificaciones de requisitos en lenguaje natural para poder enlazarlas con el resto de artefactos del desarrollo, al estilo de los diagramas de requisitos del lenguaje SysML.
2. Utilizar un lenguaje natural restringido, con palabras claves, para guiar el proceso de transformación.
3. Si existe un patrón predefinido que los expertos siguen para la generación de los modelos a partir de los requisitos, este conocimiento se pueden encapsular en la definición de la transformación de modelos.
4. Generar a partir de los modelos de requisitos el primer modelo de análisis que sirva como entrada a otro proceso de DSDM definido.
5. Generar la matriz de trazabilidad durante el proceso de transformación, que ligue los requisitos con los conceptos del modelo generado.

Capítulo 4

SIREN: un método práctico de Ingeniería de Requisitos basado en reutilización

4.1. Introducción

El propósito de la reutilización de requisitos es identificar descripciones de sistemas que puedan ser reutilizadas (en su totalidad o en parte) con un mínimo número de modificaciones, de manera que se reduzca el esfuerzo total de desarrollo [55]. Como hemos justificado en la Sección 1.1.2, creemos que es éste el nivel de reutilización que aporta mayores beneficios y ha constituido una línea de investigación central del Grupo de Investigación en Ingeniería del Software (GIIS) de la Universidad de Murcia al que pertenece esta doctoranda. El resultado más destacado es la definición del método SIREN (*Simple REuse of RequiremenNts*) [339-343], un enfoque práctico basado en reutilización y en estándares de Ingeniería del Software que ayuda a seleccionar y especificar los requisitos de un sistema software. En principio, los requisitos de SIREN se escriben en lenguaje natural, pero pueden incluir todo tipo de objetos como información complementaria del propio requisito –como por ejemplo tablas, diagramas o esquemas de cualquier tipo–.

El método SIREN incorpora un modelo de procesos en espiral, guías de aplicación de dicho proceso, un repositorio reutilizable de requisitos y una herramienta CARE (*Computer-Aided Requirements Engineering*) de soporte, denominada SirenTool [180, 249, 339]. El repositorio de requisitos reutilizables de SIREN se encuentra organizado por catálogos, que se corresponden con un conjunto de requisitos reutilizables. En SIREN estos catálogos se pueden corresponder con (1) *dominios verticales* o simplemente *dominios*, como contabilidad o finanzas; o (2) *dominios horizontales* o *perfiles*, conjuntos de requisitos aplicables de forma transversal a distintos dominios (como protección de datos de carácter personal [340, 344] o seguridad [340, 343]).

Estos catálogos, a su vez, están organizados en una jerarquía de documentos de especificación de requisitos que siguen una estructura de acuerdo a estándares del IEEE. La creación de los catálogos formaría parte del proceso *para* reutilización (SIRENp), mientras que el uso de tales catálogos formaría parte del ciclo *con* reutilización (SIRENc). En la Figura 13 se pueden ver las dos facetas del método SIREN.

Wiegers [359] afirma que más que nuevos métodos, lo que hace falta para acercar el estado de la práctica y el estado de la investigación en Ingeniería de requisitos (IR) es hacer los métodos existentes más atractivos y menos engorrosos: en definitiva, facilitar su puesta en práctica. Así, aunque existen muchas técnicas para abordar la reutilización de requisitos (véase por ejemplo la clasificación de Cybulsky y Reed [55]), SIREN se centra en la reutilización de requisitos desde un punto de vista muy práctico, intentando que su uso sea lo más sencillo posible. Más que para ser aplicado propiamente como un método completo de IR, SIREN se ha concebido para ayudar a la reutilización de requisitos en lenguaje natural en el marco de cualquier método de desarrollo de software. No obstante, SIREN también puede ser utilizado como un primer método de IR en una organización de desarrollo en la que no se haya definido un método de IR, y para ello incorpora un conjunto básico de guías generales de IR.

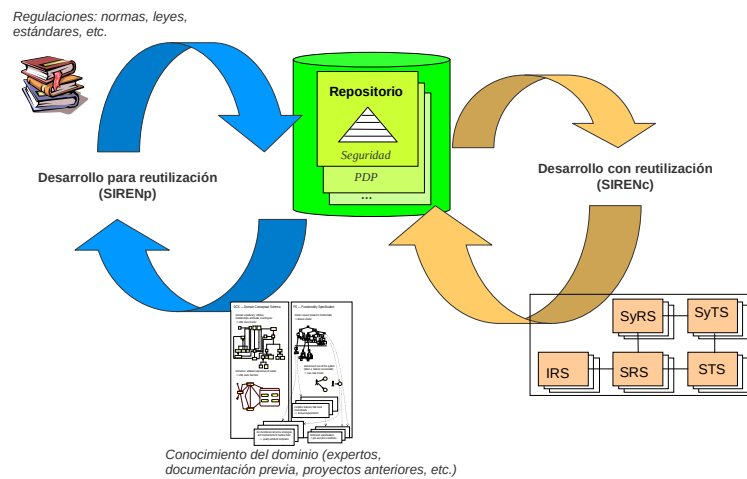


Figura 13 Ciclos con y para reutilización.

En las siguientes secciones se describe en profundidad el método SIREN, siguiendo a Nicolás *et al.* [249], documento en el que se recoge la evolución de SIREN (versión 2.0) a partir de la experiencia en el proyecto GARTIC (descrito en la Sección 1.3.3.1). Los elementos que conforman la descripción de SIREN son los siguientes: (1) en la Sección 4.2 se estudia el modelo de procesos de SIREN; (2) en la Sección 4.3 se estudian las guías que SIREN proporciona para la definición de requisitos, para la reutilización de los mismos, para la organización de los requisitos y para la mejora del catálogo; (3) en la Sección 4.4 se muestra la experiencia acumulada en la aplicación de SIREN; (4) en la Sección 4.5 se explican las cuestiones claves que en base a nuestra experiencia deben tenerse en cuenta en la reutilización de requisitos y (5) en la Sección 4.6 se muestra brevemente la herramienta de soporte. Después, en la Sección 4.7 se plantea SIREN_{MD} (donde *MD* significa *Model Driven*), una extensión a SIREN para adaptarlo al *Desarrollo de Software Dirigido por Modelos* (DSDM). Finalmente, la Sección 4.8 recopila las conclusiones acerca de esta línea de investigación de reutilización de requisitos y la aportación al estado del arte en esta línea.

4.2. Modelo de proceso de SIREN

Un proceso del software (*software process*) es un conjunto de actividades cuya meta es el desarrollo o evolución del software [326]. Un modelo de proceso del software (*software process model*) o simplemente modelo de proceso es una descripción simplificada de un proceso del software, que como cualquier labor de modelado, se realiza desde una perspectiva particular [326]. Lo que se modela en un modelo de proceso, es decir, el proceso del software, es un conjunto de actividades y artefactos asociados que producen un software. En esta sección se comienza revisando un modelo de proceso de IR básico (Sección 4.2.1), para extenderlo después con las actividades de reutilización de requisitos propias de SIREN (Sección 4.2.2).

4.2.1. Un modelo de proceso básico de IR

Como se resumía en la Sección 1.1.1, siguiendo a Kotonya y Sommerville [173] las actividades típicas de un proceso general de IR son las siguientes:

- *Identificación y consenso de requisitos (elicitation)*. En esta actividad los requisitos se “descubren” haciendo uso de técnicas de recogida de información (como entrevistas, reuniones informales, reuniones de “tormentas de ideas”, etc. El lector puede consultar [293, 360] si quiere profundizar en el estudio las técnicas de recogida de información). Otras denominaciones para esta actividad son por ejemplo extracción, adquisición o descubrimiento de requisitos.
- *Análisis y negociación de requisitos*. En esta actividad los requisitos se analizan en detalle. Los interesados negocian para llegar a un acuerdo sobre qué requisitos van a ser incluidos en el sistema. Esta actividad es necesaria, pues es inevitable que aparezcan conflictos entre requisitos que proceden de distintas fuentes y porque en ocasiones la información es incompleta o los requisitos identificados son incompatibles con el presupuesto disponible para el desarrollo.
- *Documentación (especificación) de requisitos*. Los requisitos sobre los que se ha alcanzado un acuerdo en la actividad anterior se documentan, con el nivel de detalle apropiado para que sean comprensibles por parte de todos los interesados, al tiempo que son útiles al equipo de desarrollo. Generalmente los requisitos se documentan usando lenguaje natural y técnicas diagramáticos (como diagramas UML [35]).
- *Validación de requisitos*. Los requisitos documentados se deben comprobar cuidadosamente para garantizar la consistencia de los mismos, su completitud y precisión. El objetivo de esta actividad es detectar problemas en el documento de requisitos antes de que sea utilizado como base para el desarrollo del sistema. Todos los actores que han colaborado en las actividades anteriores han de jugar un papel importante en la validación de los requisitos.

Kotonya y Sommerville [173] afirman que un ciclo de vida en espiral (ver Figura 14) es la manera más adecuada para abordar el proceso de IR, dado que no es posible obtener un conjunto completo de requisitos en una sola iteración. Las actividades de IR se repiten hasta que se adopta la decisión de que el documento de requisitos sea aceptado. Si se encuentra algún problema en el borrador del documento de requisitos se vuelve a comenzar la espiral con las mismas actividades de IR. Este proceso continúa

hasta que se produce un documento de requisitos aceptable o hasta que determinados factores externos tales como presiones en los plazos de entrega o falta de recursos obligan a dar por bueno el documento de requisitos en su estado actual.

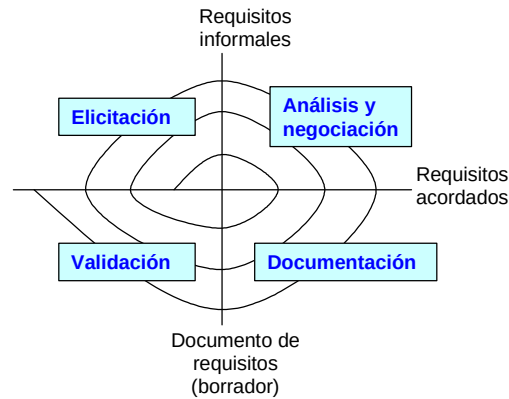


Figura 14 Modelo en espiral de un proceso general de IR.

4.2.2. El modelo de proceso de SIREN

En SIREN se asume el modelo de procesos general en espiral de Kotonya y Sommerville [173] mostrado en la Figura 14, que se enriquece con nuevas actividades para abordar específicamente las cuestiones relacionadas con la reutilización de requisitos. Como se ha señalado en la Sección 4.1, SIREN se concibe como un método de requisitos que puede aportar a cualquier método de IR la consideración explícita de la reutilización. En la Figura 15 se muestra cómo las actividades de SIREN relacionadas con la reutilización (señaladas en rojo) extienden el modelo de procesos de la Figura 14.

La Figura 16 especializa la Figura 15 para mostrar la especificación del modelo de proceso de SIREN utilizando la notación estándar SPEM (*Software Process Engineering Metamodel*) [256]. En dicho modelo aparecen las actividades del modelo de proceso de SIREN (en adelante denominadas tareas siguiendo la terminología de SPEM) junto con los roles (agentes) responsables de llevarlas a cabo (el modelo de roles de SIREN se muestra en la Sección 4.2.3). El modelo de proceso *para* reutilización se representa por la tarea T1, mientras que el modelo de proceso *con* reutilización se representa en la espiral que conforman las restantes tareas.

Para adaptar SIREN al método de IR que se haya implantado en la organización de desarrollo, las tareas específicas de SIREN (que son T1, T2, T3 y T8 en la Figura 16) se deberían ubicar en el modelo de procesos del método de IR en cuestión. Para cada una de las tareas de la Figura 16 se describe en la Sección 4.2.4 un conjunto de buenas prácticas que son de utilidad para su correcta ejecución. El orden de estos listados de buenas prácticas no determina necesariamente el orden en el que deben llevarse a cabo, puesto que en la práctica el trabajo se realiza de forma iterativa, con continuos solapamientos y retroalimentaciones.

En la Sección 4.3 se recogen las pautas de SIREN para abordar las tareas propias de la reutilización. En la descripción de las tareas se detallan las entregas resultado de las mismas. La estructura de los documentos de requisitos que aparecen en la Figura 16 se

Figura 15 Relación de las actividades de SIREN con un modelo de proceso general de IR.

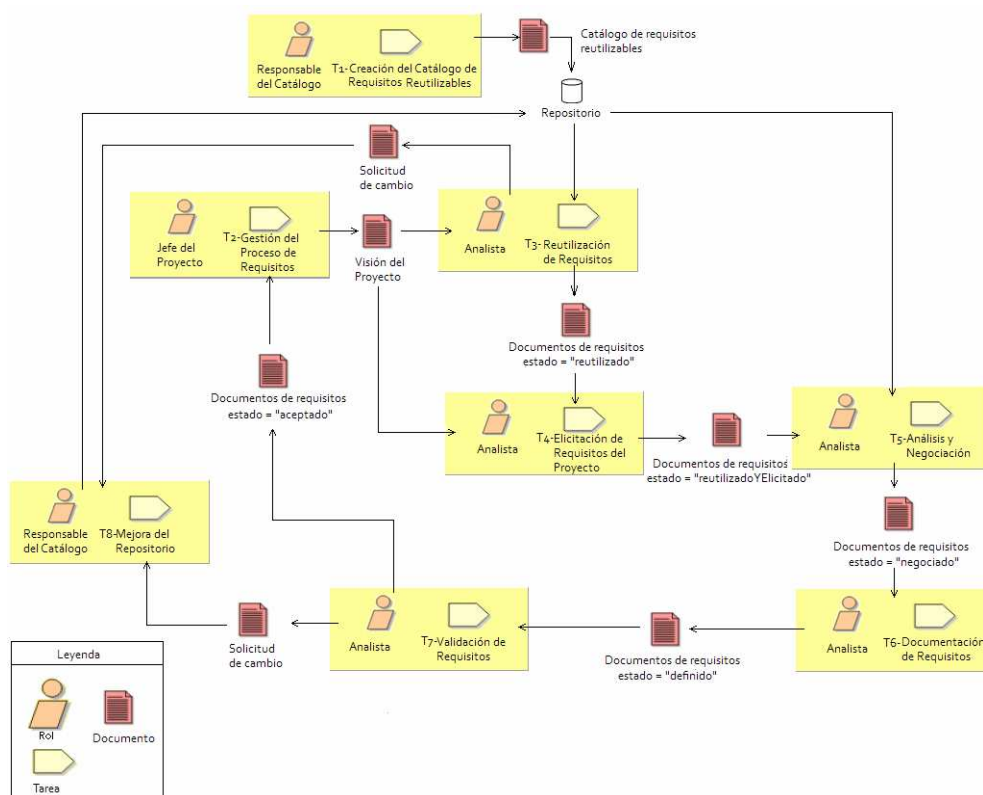


Figura 16 Modelo de proceso de SIREN con la notación SPEM.

4.2.3. Modelo de roles de SIREN

La Tabla 29 muestra los roles que se han definido en SIREN, que representan los actores o agentes que desempeñan un cierto papel en el modelo de proceso del método. Evidentemente una misma persona puede jugar distintos roles y un rol puede ser jugado por distintas personas durante el proceso de requisitos. Dado que SIREN es un método que normalmente extenderá el modelo de procesos de IR propio de la organización de desarrollo, los roles de SIREN serán un subconjunto de los roles implicados en dicho modelo de proceso general.

Rol	Descripción
Responsable del catálogo	Encargado de la coordinación de las actualizaciones de un catálogo. Para no sobrecargar de trabajo a una sola persona, en ocasiones este rol es jugado por distintas personas. El grupo de personas que desempeñan este rol constituye el <i>Equipo responsable del catálogo</i> .
Jefe de proyecto	Encargado de coordinar el desarrollo del proyecto con reutilización.
Analista o Ingeniero de Requisitos	Desempeña labores propias de analista o ingeniero de requisitos en cualquier proyecto de desarrollo de software.
Usuario clave	Conoce muy bien todo el dominio o una parte del mismo y aporta el conocimiento necesario para elaborar los documentos de requisitos. Es seleccionado de una clase de usuarios del cliente <i>equivalentes</i> para ayudar en la extracción y validación de requisitos, debido a su mayor conocimiento del dominio, su mayor jerarquía en la organización cliente o su mayor disponibilidad.
Usuario típico	Conoce el dominio o parte del mismo porque ha sido usuario de versiones anteriores del producto o de productos similares y por tanto puede aportar conocimiento útil para elaborar los nuevos documentos de requisitos.

Tabla 29 Modelo de roles de SIREN.

4.2.4. Tareas de un modelo de proceso general de IR con SIREN

En este apartado se desarrollan las tareas del modelo de procesos de IR definido con SIREN que es reflejado en la Figura 16.

4.2.4.1. T1-Creación del catálogo de requisitos reutilizables

Objetivos: En el marco de un desarrollo *para* reutilización, se trata de desarrollar un catálogo de requisitos reutilizables para un dominio o perfil determinado.

Roles implicados:

- Responsable:
 - Responsable del catálogo
- Participantes:
 - Analista
 - Usuarios clave

Buenas prácticas:

- Definir el ámbito de alcance del catálogo.
- Establecer la jerarquía de documentos de requisitos apropiada para el dominio o perfil (Sección 4.3.5).
- Adoptar una plantilla para cada documento de requisitos seleccionado (Sección 4.3.8).
- Determinar los *niveles de requisitos* (Sección 4.3.1) más adecuados para el dominio o perfil.
- Identificar las características (*features*) de las aplicaciones del *dominio* o *perfil*.
- Crear una partición del dominio o perfil en *módulos de análisis*: escoger una descomposición por funciones, modos, objetos de negocio, estímulos (partición de eventos), etc. (Sección 4.3.8)
- Definir requisitos para cada una de las características.
- Asignar requisitos a módulos de análisis.
- Identificar *clusters* de requisitos (Sección 4.3.4.3).
- Identificación de *puntos de variación* (Sección 4.3.6).
- Especificación de *requisitos parametrizados* (Sección 4.3.3).
- Identificar y establecer las relaciones de traza entre requisitos, y entre requisitos y posibles documentos (ver atributo *fuentes* en la Sección 4.3.2 y las relaciones de traza en la Sección 4.3.4).
- Generalizar requisitos potencialmente reutilizables.

Entregas:

- Catálogo de requisitos reutilizables, que se añade al repositorio.

4.2.4.2. T2-Gestión del proceso de requisitos

Objetivos: En el marco de un desarrollo *con* reutilización, se trata de planificar y monitorizar la ejecución del proceso de IR, controlar las desviaciones que se puedan producir durante dicha ejecución y si procede generar lecciones aprendidas que puedan ser de utilidad en futuros desarrollos.

Roles implicados:

- Responsable:
 - Jefe del proyecto
- Participantes:
 - Analista
 - Usuarios clave

Buenas prácticas:

- Establecer el modelo de procesos de IR (en el marco del cual se ubica SIREN) que debe dirigir el proyecto.
- Definir el ámbito de alcance del proyecto en el que se debe desarrollar el sistema o producto.
- Determinar los *niveles de requisitos* (Sección 4.3.1) más adecuados para el proyecto.

- Determinar los requisitos del negocio, es decir, las necesidades estratégicas del negocio, para determinar una *visión del proyecto* (ver Entregas en esta misma tarea).
- Identificar las características (*features*) del proyecto.
- Crear una partición del proyecto en módulos de análisis, por ejemplo basada en funciones, modos, objetos de negocio, estímulos (partición de eventos), etc.
- Identificar clases de usuarios y sus habilidades o destrezas.
- Seleccionar un *usuario clave* para cada clase de usuario (ver Tabla 29).
- Establecer grupos de *usuarios típicos* (ver Tabla 29) de versiones anteriores del producto o sistema, a partir de los cuales se pueden extraer nuevos requisitos.
- Derivar iterativamente planes y calendarios de trabajo (no sólo de IR sino del resto de desarrollo del software) a partir de los requisitos.
- Renegociar las obligaciones del proyecto (planes y calendarios) cuando los requisitos cambian.
- Documentar y gestionar los riesgos relacionados con los requisitos.
- Registrar el esfuerzo dedicado a la IR.
- Revisar las lecciones aprendidas en otros proyectos relacionadas con los requisitos.
- Generar las nuevas lecciones aprendidas en este proyecto, si las hubiera.

Entregas:

- Modelo de procesos de IR para utilizar en el proyecto.
- Visión del proyecto, que incluye ámbito de alcance del proyecto, requisitos de negocio y/o de sistema y descomposición en módulos de análisis.
- Registro de las clases de usuarios y de los usuarios clave para cada una de ellas.
- Registro de grupos de usuarios típicos.
- Planes, calendarios de trabajo y riesgos identificados.
- Registro de recursos (tiempo, personal, etc.) utilizados en el proceso de IR.
- Lecciones aprendidas (en su caso).

4.2.4.3. T3-Reutilización de requisitos

Objetivos: Se trata de descubrir y seleccionar los requisitos reutilizables del repositorio que son de aplicación en el proyecto actual.

Roles implicados:

- Responsable:
 - Analista
- Participantes:
 - Usuarios clave y típicos

Buenas prácticas:

- Refinar las particiones del proyecto en módulos de análisis.
- Identificar los catálogos reutilizables involucrados en el proyecto.
- Trabajar con los usuarios clave y típicos para la reutilización de características (*features*) y requisitos.
- Definir búsquedas de requisitos.
- Buscar y seleccionar requisitos.

- Determinar la resolución de los *puntos de variación* (Sección 4.3.6).
- Adaptar al proyecto actual los requisitos reutilizados.
- Identificar requisitos reutilizables susceptibles de ser mejorados.

Entregas:

- Documentos de requisitos en estado “reutilizado”.
- Solicitudes de cambios en el catálogo reutilizable utilizado.

4.2.4.4. T4-Extracción de requisitos del proyecto actual

Objetivos: Se trata de descubrir los requisitos que deben formar parte del proyecto actual y que no se encuentran en los catálogos de requisitos reutilizables disponibles en el repositorio.

Roles implicados:

- Responsable:
 - Analista
- Participantes:
 - Usuarios clave y típicos

Buenas prácticas:

- Escribir la visión del producto o sistema a desarrollar en el proyecto, que se debe incluir en el documento de requisitos de sistema (si existe) o en el apartado correspondiente del documento de especificación de requisitos del software. (A partir de la entrega *Visión del proyecto* de la tarea T2.)
- Refinar las particiones del proyecto en *módulos de análisis*.
- Trabajar con los usuarios clave y típicos para la identificación de nuevos requisitos no reutilizados.
- Identificar requisitos susceptibles de ser incluidos en el repositorio.
- Identificar eventos y respuestas del sistema.
- Realizar talleres de extracción de requisitos.
- Observar a los usuarios durante la realización de sus tareas.
- Examinar informes de problemas de sistemas actuales para obtener ideas sobre nuevos requisitos.

Entregas:

- Documentos de requisitos en estado “reutilizado” y “Elicitado”.
- Documento con requisitos susceptibles de ser incluidos en el repositorio.

4.2.4.5. T5-Análisis y negociación de requisitos

Objetivos: Se trata de analizar los conflictos, incompletitudes y ambigüedades que puedan estar presentes en el conjunto de nuevos requisitos reutilizados y extraídos, así como su viabilidad. Este análisis se debe realizar teniendo en cuenta los requisitos que ya han sido validados en los documentos de requisitos en ciclos anteriores del desarrollo iterativo. Por último, se debe proceder a una negociación para resolver los conflictos detectados.

Roles implicados:

- Responsable:
 - Analista
- Participantes:
 - Usuarios clave

Buenas prácticas:

- Dibujar el diagrama de contexto.
- Desarrollar prototipos -funcionales y de interfaz de usuario-.
- Analizar la viabilidad de los requisitos.
- Priorizar los requisitos.
- Modelar los requisitos.
- Crear un diccionario de datos.
- Asignar requisitos a los módulos de análisis.
- Negociar los requisitos en tres etapas: (1) etapa de *negociación*, donde se determina la naturaleza de los problemas asociados a un requisito; (2) etapa de *discusión*, donde analistas y usuarios clave discuten cómo se podrían resolver tales problemas, si es necesario examinando los requisitos presentes en el repositorio en busca de soluciones de compromiso; y (3) etapa de *resolución*, donde se decide qué requisitos pueden ser eliminados, qué requisitos deben ser incluidos en la especificación del sistema, y se sugieren modificaciones específicas para los requisitos, se reasignan prioridades y se cambia el atributo *motivación* (Sección 4.3.2) de los requisitos modificados para reflejar dichos cambios.

Entregas:

- Documentos de requisitos en estado “negociado”.

4.2.4.6. T6-Documentación de requisitos

Objetivos: Se trata de documentar mediante lenguaje natural y/o técnicas diagramáticos los requisitos que han sido acordados durante la negociación anterior.

Roles implicados:

- Responsable:
 - Analista
- Participantes:
 - Jefe de proyecto

Buenas prácticas:

- Establecer la jerarquía de documentos de requisitos apropiada para el proyecto (Sección 4.3.5).
- Adoptar una plantilla para cada documento de requisitos seleccionado (Sección 4.3.8).
- Determinar los tipos de requisitos a utilizar en la especificación (Sección 4.3.1) y revisar los atributos asociados a dichos tipos (Sección 4.3.2).
- Etiquetar de manera única cada requisito.
- Establecer el atributo *fuentes* de los requisitos (Sección 4.3.2).

- Documentar requisitos del negocio y del sistema.
- Documentar las características (*features*) del producto o sistema.
- Documentar los requisitos funcionales conforme a los módulos establecidos.
- Documentar las reglas de negocio.
- Documentar los requisitos de calidad del sistema o producto (facilidad de uso, rendimiento, eficiencia, fiabilidad, etc.).

Entregas:

- Documentos de requisitos en estado “definido”.

4.2.4.7. T7-Validación de requisitos

Objetivos: Se trata de revisar nuevamente los documentos de requisitos, en muchos casos junto con los usuarios clave, con el objetivo de eliminar cualquier error que persista en los documentos de requisitos y darlos por válidos.

Roles implicados:

- Responsable:
 - Analista
- Participantes:
 - Usuarios clave

Buenas prácticas:

- Inspeccionar los documentos de requisitos.
- Derivar casos de prueba a partir de los requisitos.
- Definir criterios de aceptación.
- Validar requisitos susceptibles de ser incluidos en el repositorio y elaborar las solicitudes de cambio en el repositorio.

Entregas:

- Documentos de requisitos en estado “aceptado”.
- Solicitudes de cambios en el repositorio.

4.2.4.8. T8-Mejora del repositorio

Objetivos: Se trata de estudiar posibles mejoras del repositorio de requisitos reutilizables y determinar su inclusión o no en el mismo. Las solicitudes de cambio se generan (1) directamente en la tarea T3, cuando se trata de mejoras de los requisitos ya incluidos en el repositorio, que van directamente al responsable del catálogo; o bien (2) en la tarea T4, debido a que los usuarios clave conocen bien un dominio o perfil del repositorio y descubren nuevos requisitos que no están presentes en el dominio o perfil. Estas últimas solicitudes que se generan en la tarea T4 han de ser validadas en la tarea T7.

Roles implicados:

- Responsable:
 - Equipo responsable del catálogo
- Participantes:
 - Analista

Buenas prácticas:

- Definir un proceso de control de cambios en los requisitos, a través del cual se proponen, analizan y resuelven cambios en los requisitos.
- Realizar el análisis de impacto de los cambios en los requisitos.
- Crear matrices de traza.
- Establecer líneas base y control de versiones de los documentos de requisitos.
- Mantener un histórico de los cambios de los requisitos.
- Medir la volatilidad de los requisitos, es decir, el número de cambios propuestos y aprobados sobre los requisitos por unidad de tiempo.

4.3. Guías de SIREN

En esta sección se incluyen guías que ayudan en la ejecución de las tareas del proceso SIREN. Estas guías son las siguientes: (1) definición de requisitos (Sección 4.3.1); (2) definición de atributos (Sección 4.3.2); (3) definición de requisitos parametrizados (Sección 4.3.3); (4) definición de trazas entre requisitos (Sección 4.3.4); (5) organización del catálogo (Sección 4.3.5); (6) reutilización de requisitos (Sección 4.3.6); (7) mejora del repositorio (Sección 4.3.7); y (8) la organización de la documentación de requisitos (Sección 4.3.8).

4.3.1. Guías para la definición de requisitos

Como se decía en la Sección 1.1.1 de esta memoria, según el IEEE 610 (*Standard Glossary of Software Engineering Terminology*) [127], un requisito se define como: “(1) Una condición o capacidad necesitada por un usuario para resolver un problema o alcanzar un objetivo; (2) una condición o capacidad que debe ser satisfecha o poseída por un sistema o componente de sistema para satisfacer un contrato estándar, una especificación u otro documento impuesto formalmente; y (3) una representación documentada de una condición o capacidad como en (1) o en (2).” Todos los requisitos no se refieren siempre a un mismo nivel de abstracción en la descripción del espacio del problema. Existen distintos niveles de requisitos, entre los cuales destacan los siguientes:

- **Requisitos del negocio** (*business requirements*, necesidades estratégicas o de alto nivel del negocio): representan objetivos de alto nivel del cliente o de la organización que solicita el sistema.
- **Requisitos del sistema** (*system requirements*): describen los requisitos de alto nivel de un sistema. Un sistema puede estar compuesto completamente de software o puede incluir subsistemas de software y hardware. Puesto que las personas también forman parte del sistema, ciertas tareas del sistema podrían asignarse a seres humanos.
- **Requisitos funcionales** (*functional requirements*): especifican los requisitos del software que se debe desarrollar para permitir a los usuarios realizar sus tareas, satisfaciendo así los requisitos del sistema. Tradicionalmente estos requisitos se documentan como: “El sistema deberá...” (en inglés “The system shall...”).

- **Reglas del negocio** (*business rules*, que en SIREN también se denominan *requisitos del dominio*): incluyen cuestiones transversales a las funciones del sistema, tales como políticas corporativas, regulaciones administrativas, estándares industriales, prácticas contables y algoritmos computacionales. Para muchos autores las reglas de negocio no son en sí mismas requisitos del software ya que existen fuera de los límites de cualquier sistema software específico. Sin embargo, las reglas de negocio obligan a que el sistema contenga funcionalidad que fuerce el cumplimiento de las mismas.
- **Requisitos no funcionales** o de calidad (*non-functional requirements*): estos requisitos extienden la descripción de la funcionalidad del sistema o del software proporcionando características emergentes que son importantes para los usuarios y los desarrolladores. Por ejemplo, los requisitos no funcionales incluyen cuestiones como facilidad de uso, integridad, portabilidad, eficiencia y robustez. Otros requisitos no funcionales describen interfaces externas entre el sistema y el mundo exterior y restricciones sobre el diseño y la implementación. Las restricciones limitan las elecciones disponibles para el diseño y construcción del producto.
- **Características** (*features*): Son abstracciones de conjuntos de requisitos funcionales relacionados que proporcionan una capacidad distinguible por parte del usuario y posibilitan la satisfacción de un requisito del negocio. Una característica está relacionada con un grupo de requisitos reconocible por un interesado, que ayuda en la toma de una decisión de compra del producto –en muchas ocasiones una característica se expresa tradicionalmente como un ítem distinto en una lista que describe el producto–.
- **Casos de uso** (*use cases*): describen secuencias de interacciones entre el sistema y un actor externo (escenarios). Un actor es una persona, otro sistema software o un dispositivo hardware que interacciona con el sistema para conseguir un objetivo. Un caso de uso refleja una actividad independiente del negocio o del software que un actor puede iniciar con el objetivo de alcanzar un valor. Un caso de uso engloba generalmente distintos requisitos funcionales que sirven a un objetivo común.

De acuerdo con Wiegers [360] un requisito debería satisfacer las características descritas a continuación:

- **Completa**: cada requisito debería describir completamente la funcionalidad que debe ser entregada. Debería contener toda la información necesaria para que el desarrollador diseñe e implemente dicha porción de funcionalidad. Si conscientemente se está omitiendo cierta información se recomienda usar el acrónimo TBD (*To Be Determined*) como una bandera estándar para resaltar este estado, que deberá ser resuelto.
- **Consistente**: un requisito consistente no debe tener conflictos con otros requisitos del mismo tipo o con requisitos de sistema o del negocio de más alto nivel.
- **Correcto**: cada requisito debería describir de forma precisa la funcionalidad que debe ser construida. La fuente del requisito (atributo *fuentes*, ver Sección 4.3.2), ya sea un usuario o un requisito de alto nivel del negocio o sistema, debe servir como referencia para la corrección.

- *Viable*: debe ser posible implementar cada requisito teniendo en cuenta las capacidades y limitaciones conocidas del proyecto, del sistema y de su entorno operativo.
- *Necesario*: cada requisito debe documentar una capacidad que los clientes necesitan realmente o que es requerida para conformar con un requisito de un sistema externo o de un estándar.
- *Priorizado*: se debe asignar una prioridad de implementación a cada requisito funcional, característica o caso de uso, para indicar cómo de esencial es para una determinada versión del producto.
- *No ambigua*: todos los lectores de un requisito deberían llegar a una interpretación única y consistente del mismo. El lenguaje natural es altamente dado a la ambigüedad, y por tanto, los requisitos se deben escribir con un lenguaje simple, conciso y directo, apropiado al dominio del usuario.
- *Verificable*: se deben poder diseñar casos de prueba u otras aproximaciones de verificación (tales como inspección o demostración) para comprobar si el producto o sistema implementa adecuadamente cada requisito. Si un requisito no es verificable no se puede determinar de manera objetiva si fue correctamente implementado.
- *Modificable*: un requisito debe ser revisado cuando sea necesario y se debe mantener una historia de los cambios realizados sobre él. Esto obliga a que cada requisito sea etiquetado de forma única y especificado de manera separada de otros requisitos, de manera que se pueda referenciar de forma unívoca.
- *Trazable*: un requisito trazable puede ser enlazado hacia atrás, a su origen, y hacia delante a los elementos de diseño o código fuente que lo implementan y a los casos de prueba que verifican que la implementación es correcta.

Independientemente de los niveles de requisitos presentados anteriormente, en SIREN, cada requisito tiene asociado un *tipo de requisito*. Por ejemplo, requisitos de sistema (SYRS), características (FEAT), requisitos del software (SRS), requisitos de seguridad del software (SRSS), etc. El tipo de requisito determina el formato del identificador único del requisito y su conjunto de atributos. Cada catálogo de requisitos tiene un conjunto predefinido de tipos de requisitos, que puede ser ampliado. Cuando se reutiliza un requisito, es preciso tener en cuenta los atributos de los tipos del requisito origen y del requisito destino (Sección 4.3.2).

4.3.2. Guías para la definición de atributos

Cada tipo de requisito tiene predefinido un conjunto de atributos que permiten almacenar información acerca de distintas características de los requisitos del tipo. Para cada tipo de requisitos, el conjunto de atributos puede ser ampliado.

En SIREN, todos los tipos de requisitos tienen asociado un conjunto mínimo de atributos, que se denomina MIN [249, 339] y que se muestra a continuación. Están marcados como forzosos aquellos a los que hay que asignar un valor obligatoriamente en el momento de la creación del requisito:

- *texto* (forzoso): la sentencia en lenguaje natural que especifica el requisito.
- PUID (*Project Unique Identification*) o identificador Único (forzoso) del requisito dentro del proyecto.

- *riesgo*: se debe ponderar cada uno de los requisitos comparándolos con el resto y realizar una estimación del riesgo inherente al requisito (por ejemplo: alto, medio y bajo).
- *criticidad*: cómo de importante es el requisito para el cliente. Por ejemplo, un esquema sencillo es prioridad alta -requisito obligatorio-, media -requisito recomendable-, y baja -requisito opcional-. En el caso de que exista una relación de exclusividad (Sección 4.3.4.4) entre dos o más requisitos obligatorios, se entenderá que sólo uno de ellos debe estar obligatoriamente en la especificación final de requisitos.
- *prioridad*: el valor de este atributo será establecido por el analista y ayudará a establecer un orden en el desarrollo. Los atributos riesgo y criticidad puede servir como guía para establecer el valor de la prioridad.
- *motivación*: indica la razón por la que el requisito está incluido en el proyecto.
- *estado*: se identifican nueve estados en los cuales un requisito se puede encontrar, tal y como se muestra en la Figura 17.
 - *Pendiente de definición*: la redacción del requisito reutilizado o extraído no se considerada definitiva. Por ejemplo, se ha reutilizado un requisito parametrizado que no ha sido instanciado todavía o se ha definido un requisito parametrizado al que todavía no se ha asignado un tipo al parámetro. En el caso de que el requisito esté incompleto, este estado también se puede indicar explícitamente en el texto del requisito utilizando el acrónimo TBD (*To Be Determined*).
 - *Pendiente de revisión*: el texto del requisito ya se considera completo y está en espera de su revisión (análisis y negociación).
 - *Pendiente de documentación*: tras el análisis y la negociación el requisito es aprobado y está pendiente de su documentación, es decir, de su ubicación definitiva en el documento de especificación de requisitos y, opcionalmente, de la revisión del texto y/o atributos del requisito.
 - *Descartado*: como resultado del análisis y negociación o bien de la validación el requisito es descartado y debe ser renegociado o eliminado.
 - *Pendiente de validación*: una vez que el requisito ha sido documentado se encuentra en estado pendiente de validación. Después de la validación, es posible que el requisito pase al estado *Descartado* si precisa su renegociación.
 - *Validado en análisis*: el requisito ha sido aceptado por todos los interesados, usuarios clave incluidos.
 - *Modelado*: el requisito ha sido modelado en análisis y/o diseño.
 - *Implementado*: el requisito ha sido codificado.
 - *Verificado*: una vez implementado, se comprueba por parte del equipo de desarrollo que la implementación se ajusta a la especificación del requisito. En caso contrario el requisito requiere la revisión de su implementación
 - *Validado en implementación*: una vez verificado, la implementación del requisito ha sido aceptada por todos los interesados, incluidos los usuarios clave. Si la implementación del requisito no es aceptada, el

requisito puede requerir su redefinición o la revisión de la implementación.

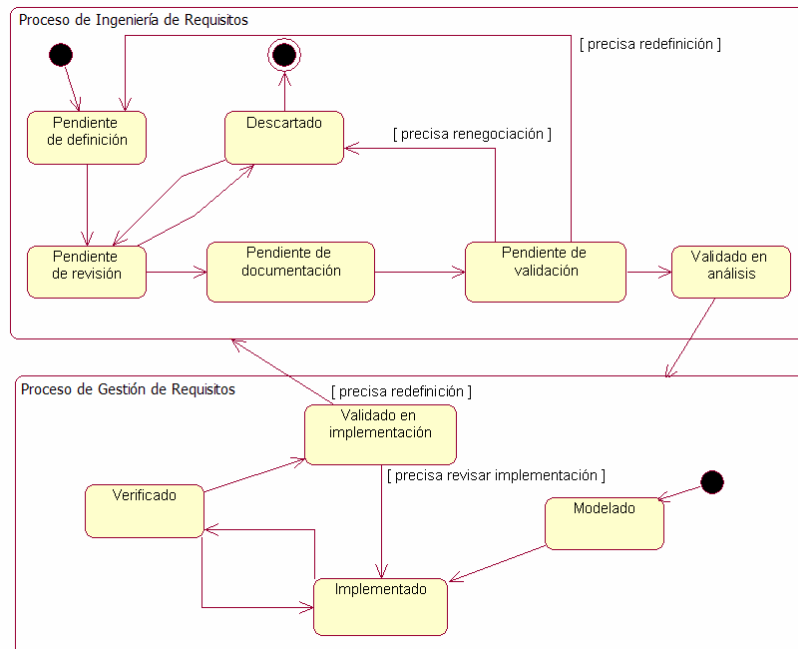


Figura 17 Diagrama de estados de un requisito.

- *fuente*: es un campo de texto en el que se debe poner de dónde procede el requisito. Los requisitos pueden proceder (1) de las necesidades del cliente (y entonces en este atributo se indicará el nombre del usuario clave o típico involucrado); o (2) de restricciones sobre la solución técnica del proyecto, de la legislación vigente o de estándares (y entonces en fuente se deberán plasmar las referencias bibliográficas, direcciones web, estándares, etc. correspondientes). En un requisito que haya sido reutilizado del repositorio, este atributo fuente sirve también para reflejar de qué catálogo proviene y de qué requisito reutilizable.
- *criterios Validación*: indica los criterios de validación que son necesarios para probar el requisito. Normalmente estos criterios se incluyen en el documento STS (*Software Test Specification*), descrito en la Sección 4.3.5.
- *propuestoPor*: indica el usuario que solicita que el requisito sea incluido.
- *responsable*: miembro del equipo de desarrollo responsable de la implementación del requisito.
- *sección*: indica la sección del documento en la que está especificado el requisito. Por ejemplo, “1.2 Alcance”, “3.1.1 Interfaces de usuario”. En este atributo se incluye, además del número de la sección, el nombre de la misma para facilitar búsquedas de requisitos a reutilizar por este atributo.

- *históricoVersiones*: realiza un registro de todas las versiones del requisito: autor de la formulación o reformulación del requisito, fecha, versión y texto de la versión enésima.

Cuando se reutiliza un requisito perteneciente a un tipo A, el nuevo requisito pertenecerá a un tipo B. Este tipo B podrá ser el mismo que A, en cuyo caso todos los valores de los atributos del requisito reutilizado serán los mismos que tenía el requisito original de A, excepto el campo fuente que indicará la procedencia del requisito reutilizable del que es copia. Sin embargo, si B es distinto de A, entonces el requisito reutilizado tendrá los mismos valores que el requisito reutilizable en los atributos pertenecientes al conjunto MIN, puesto que $MIN \subseteq A \wedge B$. El valor de un atributo de B que no esté en A será el valor por defecto del tipo de dicho atributo de B. La herramienta CARE de soporte a SIREN deberá permitir la modificación de los valores de los atributos propios de B (no incluidos en A) del requisito reutilizado.

4.3.3. Guías para la definición de requisitos parametrizados

Un requisito se ha de parametrizar si ciertas opciones se pueden elegir entre un conjunto de posibles valores cuando el cliente adquiere el producto o el equipo de desarrollo especifica los requisitos del sistema a desarrollar. Un requisito parametrizado dispone de uno o varios parámetros que sirven para diferenciar un producto reutilizado de otro posible producto reutilizado dentro del catálogo. El parámetro sirve para especificar un punto de variación en la especificación de requisitos, esto es, un punto en el que se debe seleccionar entre distintas alternativas para la configuración de un producto concreto. Si todos los tipos posibles deben ser soportados por el producto, y se puede elegir entre uno u otro en tiempo de explotación y no en tiempo de desarrollo, tal cuestión no se debería especificar como un parámetro. Por ejemplo:

SRS1. El sistema permitirá la exportación a ficheros externos para la generación de informes.

- De tipo CVS
- De tipo Excel

El anterior sería un requisito parametrizado si cuando el cliente adquiere el producto se puede elegir entre una u otra clases de exportación (o ambas), quedando el requisito siguiente:

SRS2. El sistema permitirá la exportación a ficheros externos para la generación de informes en [unFormato].

donde el valor del parámetro unFormato podría ser: CVS o Excel.

Si no fuera así, es decir, si las dos exportaciones debieran ser soportadas por el producto, no se debería especificar como un requisito parametrizado, sino más bien cómo tres requisitos diferentes con una relación *padre-hijo* (esta relación se describe en la Sección 4.3.4.1):

SRS1. El sistema permitirá la exportación a ficheros externos para la generación de informes en cualquiera de estos formatos:

- SRS1.1 Formato CVS
- SRS1.2 Formato Excel

Todo parámetro viene caracterizado por tres atributos:

- *Nombre*.
- *Tipo*. En todo catálogo existen dos tipos predefinidos: *numérico* y *cadena* de texto. Estos tipos se pueden ampliar con la definición de nuevos tipos específicos para cada catálogo. Los nuevos tipos representan un conjunto finito de valores (enumerados), y será necesario determinar el número mínimo y máximo de valores que el cliente podrá elegir en el momento de la instanciación. Si el mínimo y el máximo se han fijado con valor 1 quiere decir implícitamente que el tipo no es multivaluado.
- *Alcance*. Un parámetro puede ser *local* a un requisito o *global* al catálogo, en cuyo caso al tomar un valor todos los requisitos que incluyan ese parámetro tomarán el mismo valor.

A continuación se presentan ejemplos de requisitos parametrizados extraídos del catálogo *SEGURIDAD-PDP* (Protección de Datos de carácter Personal) (Sección 4.4.1):

SRS2. El responsable del fichero elegirá un [unSoporteFisico] para la realización de copias de seguridad.

Parámetro [nombre="unSoporteFisico", tipo =SOPORTE_FÍSICO {cinta magnética, disco óptico, disco magnético, memoria flash, papel} (1,1), alcance = GLOBAL]

SRS3. El responsable del fichero adquirirá la [unaUnidad], para el [unSoporteFisico] elegido.

Parámetro [nombre = "unaUnidad", tipo= UNIDAD_ALMACENAMIENTO {disco duro, disco duro externo, disquete, CD-ROM, DVD, unidad ZIP, unidad de cinta, pendrive, Blu-ray, HD-DVD, disco USB} (1,1), alcance = GLOBAL]

Vemos que el parámetro unSoporteFisico se ha definido con alcance global, por tanto, ambos requisitos (SR2 y SR3) deben tener el mismo valor para este parámetro cuando se instancien.

SRS4. El periodo máximo de almacenamiento de los pedidos será de [tiempoEnDias] días.

Parámetro [nombre = "tiempoEnDias", tipo = Numérico, alcance = LOCAL]

El requisito anterior también podría incluir un nuevo parámetro para determinar el periodo (días, meses, etc.). Si el conjunto de valores para los posibles periodos de tiempo es finito, se definiría un nuevo tipo de parámetro en el catálogo, por ejemplo, UNIDAD_TIEMPO.

SRS4bis. El periodo máximo de almacenamiento de los pedidos será de [tiempoEnDias] cada [unidadTiempo].

Parámetro [nombre = "unidadTiempo", tipo = UNIDAD_TIEMPO{día, mes, año}(1,1), alcance = LOCAL]

SRS5. El sistema transformará datos de una fuente que estarán en [formato].

Parámetro [nombre = "formato", tipo = FORMATO{xml, jpg, texto}(1,3), alcance = LOCAL]

En este último ejemplo vemos que se trata de un parámetro multivaluado puesto que se puede elegir en el momento de la instanciación del parámetro de uno a tres valores. Un requisito parametrizado inicialmente se encuentra en estado *TBD*, el valor del atributo *estado* es *Pendiente de definición*. Este valor no cambiará a *Definido* hasta que todos los parámetros que contenga en su especificación hayan sido instanciados.

En tiempo de reutilización no se pueden añadir nuevos valores a un parámetro. Si un parámetro *p* tiene como posibles valores *a*, *b* y *c*, el Analista no puede añadir un nuevo valor *d*, sino que debe dejar sin instanciar el parámetro *p* y utilizar después la aplicación de gestión de parametrizados de la herramienta (Sección 4.6) para añadir el nuevo valor.

4.3.4. Guías para la definición de trazas entre requisitos

Las trazas permiten establecer relaciones entre requisitos (ver Sección 1.1.3.3). El modelo de trazas definido en SIREN incluye las trazas *padre-hijo*, *requiere*, *relacionadoCon*, *exclusiva* y *materializaEn*. En las siguientes secciones se describen con detalle estas trazas.

4.3.4.1. Traza padre-hijo

Definición

Relación de descripción de un requisito más general por parte de una secuencia de requisitos específicos. Los identificadores de los hijos extienden el identificador del padre con la notación *idPadre.1*, *idPadre.2*, etc. En el documento y en el repositorio de requisitos, usualmente el texto del padre y el de los hijos está localizado espacialmente.

Excepción a la regla general de que los requisitos sean autocontenidos: se permite que el texto del padre esté escrito parcialmente, siendo completado por los hijos.

En tiempo de reutilización (desarrollo *con* reutilización):

- (a) Los requisitos hijos son opcionales (el punto de variación viene dado implícitamente).
- (b) Los requisitos hijos sólo tienen sentido en el contexto del padre: por tanto, reutilizar un hijo implica la reutilización obligatoria del padre y el mantenimiento de la relación *padre-hijo*.

Uso

Cuándo un requisito se descompone directamente en otros:

- (a) Descripción de los componentes, atributos o características de un objeto de negocio o de un componente de la interfaz. Por ejemplo: REQ1. Un socio es descrito por los siguientes campos: REQ1.1. Nombre. REQ1.2. DNI. REQ1.3. Domicilio. (...)
- (b) Descripción de las distintas alternativas para desarrollar un proceso o un objeto. Por ejemplo: REQ1. La autenticación del usuario se puede realizar: REQ1.1. Por contraseña. REQ1.2. Por la voz. REQ1.3. Por el iris. Este tipo de requisitos no se debería especificar como parametrizado en el caso de que los hijos fueran susceptibles de tener una traza con otros requisitos: en ese caso el punto de variación siempre se tiene que especificar mediante una relación *padre-hijo*.

- (c) Enumeración de ejemplos que sirven para mejorar la comprensión del requisito padre. Tras la reutilización, se debe asegurar que o bien se eliminan los ejemplos, o se amplían para cubrir todas las alternativas exigidas por la aplicación: un conjunto incompleto de ejemplos no se puede considerar propiamente un conjunto de requisitos.
- (d) Especificación de casos de uso. El nombre del caso de uso se pondría en el texto del requisito, y los pasos de los escenarios se pondrían como requisitos hijos: propiamente “requisitos” serían únicamente los pasos realizados por el sistema, que deberían ser escritos como requisitos atómicos. Sin embargo, para que se pueda reutilizar el contexto de los requisitos, las acciones de los actores también se han de poner como requisitos. Los campos de la plantilla de descripción del caso de uso también serían requisitos hijos. Obsérvese que el hecho de especificar los casos de uso mediante relaciones *padre-hijo* y no otro tipo de relaciones implica que el grano de reutilización sea el del caso de uso, no el del paso del caso de uso: un hijo (paso) no se puede reutilizar sin su padre (caso de uso), y por tanto no se pueden reutilizar pasos individuales (si se considera esta limitación demasiado fuerte, habría que usar la traza *relacionadoCon* –ver más adelante).
- (e) Para organizar los requisitos según el criterio usado para estructurar el repositorio (funcional, estímulo, escenario, estado, etc., ver modos de organización según el IEEE 830 en la Sección 4.3.8), no se usa *padre-hijo*: se usa la traza *relacionadoCon* –ver más adelante.

4.3.4.2. Traza *requiere*

Definición

Relación de dependencia direccional entre dos requisitos. *A requiere B* significa que B es una precondition de A. Ello implica que para que A se cumpla en el nuevo sistema B se tiene que cumplir necesariamente. Por tanto, la reutilización de A implica obligatoriamente la reutilización de B. Se podría decir que *requiere* es una relación inclusiva fuerte.

Uso

La existencia de un requisito es condición necesaria para la existencia de otro. Por ejemplo, dados Req.A. El botón de alarma debe tener el icono de una campana y Req.B. El ascensor ha de proporcionar un botón de alarma, tenemos Req. A *requiere* Req.B.

4.3.4.3. Traza *relacionadoCon*

Definición

Relación de dependencia bidireccional entre dos requisitos. Establece que un requisito A está “relacionado con” un requisito B: (1) A está relacionado con B, B refina o complementa A de alguna manera, de forma que cuando se reutilice A “se debe considerar” B para su posible reutilización; o (2) A y B forman parte del mismo *clúster* –ver uso (b) más abajo–. Se podría decir que *relacionadoCon* es una relación inclusiva débil.

Uso

- (a) Enlace entre requisitos de distinto nivel de abstracción. Por ejemplo, un requisito de sistema requiere un requisito de software, un requisito de software requiere unas pruebas de aceptación. No son relaciones inclusivas fuertes (traza *requiere*) porque el diseño e implementación de un requisito se puede realizar de distintas maneras, y no se puede obligar a que sea de una forma determinada.
- (b) Permite aumentar el tamaño del grano de la reutilización a través de la definición de *clusters de requisitos verticales* descripción de una función, estado, etc. (según la organización de IEEE 830 escogida, ver Sección 4.3.8), de forma que los requisitos refinados sean tenidos en cuenta en tiempo de reutilización. La especificación de requisitos queda formada así por un conjunto de árboles de requisitos, cada uno de ellos jerarquizado por función/subfunción, estado/subestado, etc.
- (c) Ayuda a la definición en la especificación de requisitos de *clusters de requisitos horizontales* o *aspectos de requisitos* a partir de cuestiones como la seguridad o la protección de datos personales, las relaciones inclusivas débiles podrían ser “transversales”, horizontales a los árboles citados en el punto (b) anterior, atravesando transversalmente la jerarquía de documentos de requisitos y la organización escogida de tales documentos. Examinando el apartado del documento al que pertenece el requisito raíz de estos árboles horizontales se podría conocer el clúster al que pertenece un requisito (seguridad, por ejemplo), pero parece adecuado introducir un nuevo campo en los requisitos de SIREN, *clúster*, que recoja una colección de las “etiquetas” que se han asignado a las trazas entre requisitos: [seguridad], [protección de datos], etc. Como vemos, los clusters no son disjuntos, pues se trata de una colección de etiquetas para las relaciones entre requisitos. Un requisito puede así estar relacionado con distintos requisitos de acuerdo a distintos criterios (*clusters*). La relación *padre-hijo* no se puede usar para definir estos clusters ya que un requisito no puede tener más de un padre.

4.3.4.4. Traza exclusiva**Definición**

Relación de exclusividad entre dos requisitos: R1 *exclusivo-con* R2 significa que si R1 está presente en la especificación R2 no puede estar, y recíprocamente si R2 está en la especificación R1 no puede estar.

Uso

- (a) Dos requisitos están en conflicto y no se pueden satisfacer simultáneamente.
- (b) Con los requisitos exclusivos el punto de variación queda implícito: se resuelve al escoger un requisito u otro. Si dicho punto de variación se refiriera a más de dos requisitos, la especificación se puede volver engorrosa, al tener que especificar relaciones de traza exclusiva dos a dos entre todos los requisitos implicados. Puede ser preferible usar en ese caso una relación *padre-hijo*, indicando en el texto del padre que los hijos son exclusivos.

- (c) Requisitos parametrizados en los que hay relación de trazas entre las distintas opciones del parámetro: la solución es hacer explícita cada opción como un requisito distinto, manteniendo relaciones de exclusividad entre ellos. Pero como se comenta arriba en (b), si la especificación se hace engorrosa tal vez sea preferible usar relaciones *padre-hijo*.

4.3.4.5. Traza materializaEn

Definición

Relación entre un requisito y un artefacto del desarrollo, como una clase, un módulo, un componente, etc.

Uso

Ligadura entre un requisito y un artefacto del desarrollo, representado por un tipo de requisito (clase, módulo, componente, etc.).

4.3.5. Guías para la organización de los catálogos

En SIREN el repositorio de requisitos reutilizables está formado por un conjunto de catálogos de requisitos reutilizables que pueden ser dominios o perfiles (ver Sección 4.1). A su vez, cada uno de estos catálogos se organiza a partir de la jerarquía de documentos de requisitos que aparece representada en Figura 18. Cada documento se corresponde con un nivel de especificación diferente y por tanto, tiene objetivos y destinatarios distintos.

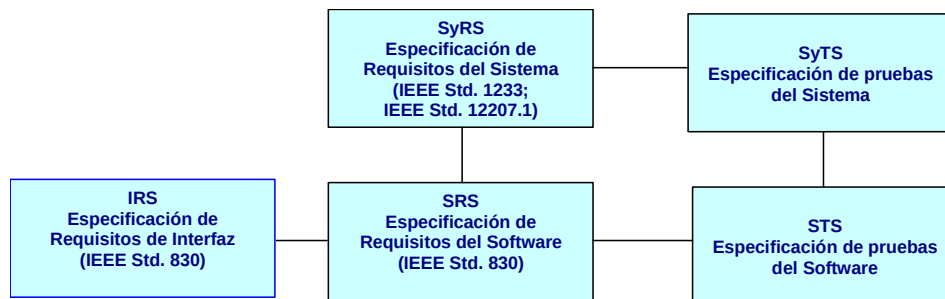


Figura 18 Una posible jerarquía de documentos de requisitos para SIREN.

En la Figura 18 aparecen cinco tipos distintos de documentos. No son todos obligatorios para cualquier catálogo o proyecto. La decisión sobre la jerarquía a utilizar en el proyecto actual corresponde al jefe de proyecto junto a los analistas (ver Tabla 29), y depende del tamaño del proyecto y de la complejidad del sistema que se va a desarrollar. Cuando en esta memoria de tesis doctoral hablamos genéricamente de documento de requisitos se hace referencia bien a un documento concreto de la jerarquía o bien a la jerarquía completa.

Los documentos considerados en la Figura 18 se describen a continuación:

- Los documentos de Especificación de Requisitos del Sistema (que se referenciarán por sus siglas en inglés, *System Requirements Specification, SyRS*) y Especificación de Pruebas del Sistema (*System Test Specification, SyTS*) son únicos y opcionales. No son necesarios cuando el problema es simple y está claro desde el principio que todos los requisitos van a ser soportados por el software -por ejemplo, en el desarrollo de un sistema autónomo simple y bien conocido-.
- El documento de Especificación de Requisitos del Software (*Software Requirements Specification, SRS*) es obligatorio y puede dividirse en subdocumentos (por ejemplo, un SRS por cada módulo o subsistema).
- El documento de Especificación de Requisitos de Interfaz (*Interface Requirements Specification, IRS*) es único y opcional y se utiliza cuando los requisitos de interfaz son muy numerosos, con el objeto de evitar que el SRS sea muy extenso; en otro caso, estos requisitos se incluyen en el SRS.
- Se debe definir un documento de Especificación de Pruebas del Software (*Software Test Specification, STS*) por cada SRS de la jerarquía.

En SIREN las plantillas de documentos de requisitos están basadas en los estándares de IEEE (Figura 18). Sin embargo, en la práctica los documentos de requisitos raramente se derivan directamente del estándar correspondiente, sino que cada organización debe particularizarlos teniendo en cuenta el tipo de proyectos que desarrolla y el tamaño y la cultura del equipo de desarrollo. A continuación se describen estas plantillas:

- *Especificación de Requisitos del Sistema (SyRS)*. De acuerdo con el estándar IEEE 12207.0 [129], el SyRS debe incluir: (1) funciones y capacidades del sistema (requisitos del sistema); (2) requisitos del negocio, organizativos y de usuario; (3) requisitos de seguridad (*security*), seguridad a terceros (*safety*) y privacidad (*privacy*); (4) requisitos de ingeniería de factores humanos; (5) requisitos de operaciones y mantenimiento; y (6) restricciones de diseño. A partir de estas cuestiones, el estándar IEEE 12207.1 [130] detalla el contenido específico de la plantilla del SyRS haciendo referencia al estándar IEEE 1233 [132], *Guide for Developing System Requirements Specifications*, donde se pueden encontrar guías para especificar los requisitos del sistema y la propuesta de un esquema del SyRS. Este esquema es la base de la plantilla del SyRS de SIREN.
- *Especificación de Requisitos del Software (SRS)*. La mayoría de los requisitos del software se obtienen directamente a partir de los requisitos del sistema; de ahí la visión jerárquica presentada en la Figura 18. En SIREN el SRS se basa en el estándar IEEE 830 [131], en el que se recogen requisitos acerca de la funcionalidad del sistema (requisitos funcionales); interfaces externas; restricciones de diseño; requisitos no funcionales o de calidad (rendimiento, portabilidad, mantenimiento, seguridad, disponibilidad y fiabilidad); características (*features*) y reglas de negocio.
- *Especificación de pruebas del sistema y del software (SyTS y STS)*. En los documentos de pruebas (SyTS o STS) se especificarán casos de prueba para

asegurar que el sistema o el software cumple los requisitos especificados en el SyRS y SRS.

- *Especificación de requisitos de interfaz* (IRS). Los requisitos relacionados con los interfaces entre los elementos del software y entre el usuario y el sistema podrían incluirse en el SRS. No obstante, con el objetivo de reducir el tamaño del SRS en el caso de documentos muy extensos, en algunas ocasiones es conveniente hacer uso de un documento separado del SRS denominado IRS. Por tanto, se deben establecer las relaciones de traza adecuadas entre el SyRS, el SRS y las interfaces descritas en el IRS. La plantilla del IRS tiene la misma estructura que el apartado del SRS dedicado a la especificación de requisitos de interfaz [131].

Si bien esta es la organización documental propuesta en SIREN para la especificación de requisitos, usualmente las herramientas CARE almacenan los requisitos en una base de datos, de modo que los analistas pueden trabajar directamente sobre esta base de datos, en lugar de manejar documentos textuales de requisitos. Por tanto, también se debe prestar atención a la organización de los requisitos en la base de datos. Es recomendable organizar los requisitos en la base de datos en carpetas o módulos que mimeticen la estructura de los documentos de requisitos, en lugar de tener una larga lista no estructurada de requisitos.

4.3.6. Guías para la reutilización de requisitos

Una vez establecido el ámbito de alcance del proyecto se debe buscar en el repositorio SIREN si existe algún catálogo de dominio conforme a dicho ámbito. Si es así, posiblemente el proyecto se corresponde con el desarrollo de un producto concreto del dominio especificado en el catálogo. En este caso, las búsquedas en el repositorio podrían comenzar buscando en el catálogo del dominio los requisitos con el valor *alta* en el atributo *críticidad* (Sección 4.3.2). Al ser obligatorios, estos requisitos forman parte de cualquier producto de dicho dominio y por tanto deben reutilizarse todos ellos, analizando sus trazas para determinar otros requisitos no obligatorios que también deban formar parte de la especificación del proyecto actual.

Una vez establecida la especificación común del producto se deben definir nuevas búsquedas guiadas por los requisitos del negocio y las características (*features*) identificadas. Estas guías pueden implicar búsquedas en el mismo catálogo de dominio o bien en distintos catálogos de perfil.

Si como resultado de alguna de las búsquedas se reutiliza un requisito que tiene relaciones de traza *relacionadoCon* otros requisitos, se ha encontrado un *clúster* de requisitos. En este caso se debe considerar la selección de los requisitos englobados en dicho clúster.

La reutilización de los requisitos seleccionados implica la resolución de los puntos de variación encontrados en tales requisitos:

- Instanciación de los parámetros de los requisitos parametrizados con los valores adecuados al proyecto actual.
- Resolución de trazas exclusivas.
- Resolución de trazas *relacionadoCon*, que son opcionales.
- Resolución de trazas *requiere*, que deberían incluirse en condiciones normales.

- Resolución de trazas *padre-hijo*, que deberían incluirse en condiciones normales.

4.3.7. Guías para la mejora del repositorio

Se recomienda crear una línea base (*baseline*) para cada catálogo del repositorio cuando finaliza un proyecto que lo haya utilizado y haya implicado una mejora del mismo.

Un proceso sencillo de control de cambios del repositorio es el siguiente:

1. Los analistas que estén trabajando en un proyecto en el que se reutilicen requisitos de algunos de los catálogos, podrán enviar peticiones de cambio al responsable del catálogo (Tabla 29). Además de las peticiones de cambio de requisitos existentes en el catálogo, también pueden formular peticiones para la inclusión de nuevos elementos en el catálogo (tipos de parámetros, requisitos del proyecto actual, etc.) tras la validación de los mismos.
2. El equipo responsable del catálogo analizará las peticiones de cambio para determinar su viabilidad. En el caso de que la petición implique una mejora en un catálogo (cambios menores en la redacción de un requisito, cambios en los valores de los atributos, etc.), el cambio podría ser automáticamente incorporado y con ello visible a todos los proyectos que estén utilizando el catálogo. En el caso de que la petición de cambio implique una extensión o reducción de los requisitos del catálogo (ampliación o reducción de los valores de un tipo parametrizado, nuevas trazas exclusivas, etc.), es preciso estudiar los proyectos en marcha que hacen uso del catálogo para determinar cómo les afecta el cambio.

4.3.8. Guías para la organización de la documentación de especificación de requisitos

De manera general, la organización del documento de especificación de requisitos de un producto o sistema concreto se puede abordar de acuerdo con las plantillas propuestas por el estándar IEEE 830 [131]. En la Sección 3 de la estructura propuesta por el estándar se plantean distintas organizaciones para los requisitos específicos del sistema: (1) por modos; (2) por clases de usuarios; (3) por objetos; (4) por características (*features*); (5) por estímulos; y (6) por jerarquía funcional. Según el ámbito del proyecto y la cultura del equipo de desarrollo se debe elegir una u otra organización.

De forma más concreta unas guías sencillas para la obtención y refinamiento de los requisitos del proyecto son las siguientes:

- Realizar la declaración de ámbito de alcance.
- Establecer formalmente los límites del sistema (por ejemplo, mediante un diagrama de contexto [364]).
- Determinar los módulos de análisis.
- Determinar (reutilizar) las características (*features*) del sistema o producto a desarrollar.
- Determinar (reutilizar) los requisitos funcionales y no funcionales a partir de las características anteriores.

- Determinar (reutilizar) las reglas de negocio a partir del conocimiento de los dominios y perfiles implicados, incluyéndolas en un apartado diferenciado del documento de especificación de requisitos denominado “Reglas de negocio” o “Requisitos del dominio”.
- Recorrer los *clusters* de requisitos especificados en los catálogos reutilizables considerados en el proyecto para determinar requisitos funcionales, no funcionales y/o reglas de negocio que pueden ser reutilizados.
- Asignar las características, requisitos funcionales, no funcionales y reglas de negocio a los apartados del documento siguiendo el criterio de organización escogido.

Estos pasos se realizan con continuas iteraciones y solapamientos entre ellos.

4.4. Experiencias en la aplicación de SIREN

SIREN se ha utilizado tanto en dominios verticales o perfiles (seguridad [116, 340, 343] y ley de protección de datos de carácter personal [344]) como en dominios horizontales (sistemas teleoperados [248]) y se ha aplicado en proyectos reales, tanto en su vertiente *para* reutilización como en su vertiente *con* reutilización. Además, ha servido como base para la definición de un método de auditoría de datos personales (AUDIT-PDP) que ha sido validado mediante nueve casos de estudio en empresas reales utilizando el método de Investigación en Acción [206]. En los siguientes apartados resumimos las experiencias en los ámbitos donde esta doctoranda ha tenido algún grado de participación.

4.4.1. SIREN y la seguridad

Actualmente hay definidos tres catálogos de requisitos reutilizables, que están a disposición de los usuarios de SirenTool (Sección 4.6), los tres en el ámbito de la seguridad de sistemas de información. Según la terminología de SIREN son *perfiles*, pues tratan con dominios horizontales. Los dos primeros son los siguientes:

- Catálogo *Seguridad-PDP (Protección de Datos Personales)* [344], que incluye los requisitos procedentes de la Ley Orgánica de Protección de Datos Personales (LOPD) [195] (que es una adaptación de la directiva europea 95/46/CE) y del Reglamento de Medidas de Seguridad (RMS) [288].
- Catálogo *Seguridad-MAGERIT* [343], que incorpora requisitos procedentes de MAGERIT [202], el Método de análisis y gestión de riesgos de la administración pública española, basado en ISO/IEC 15408-1999, *Evaluation Criteria for Information Technology Security Standard*, también conocido como *Common Criteria Framework-CCF* [139].

En una primera fase, esta doctoranda ha participado en los trabajos iniciales con estos catálogos, especialmente con el de *Seguridad-MAGERIT* (compatible con MAGERIT 1.0) [340, 343], y ha participado activamente en su revisión en el marco del proyecto fin de carrera de Miguel Ángel Martínez Aguilar [205]. Con posterioridad, en una segunda fase, en el marco de la tesis doctoral de Joaquín Lasheras [178], centrada en esta línea de investigación, se generaron las versiones actuales de los catálogos

(revisándolas de nuevo y adaptándolas a MAGERIT 2.0), y se desarrolló un nuevo catálogo, también disponible en SirenTool:

- Catálogo *Seguridad-ISO 27001*, que integra los requisitos de los catálogos anteriores y los relaciona con los objetivos del estándar de seguridad ISO 27001 [140] para un sistema de gestión de seguridad de la información.

También en el ámbito de la seguridad, pero no disponible en SirenTool, se ha definido un catálogo de requisitos en el marco de los sistemas basados en servicios web (SW) con el objeto de simplificar la tarea de selección de los mecanismos y estándares de seguridad más adecuados para ser aplicados en el diseño sistemas basados en este paradigma:

- Catálogo *SWS (Security for Web Systems)* incluye plantillas de requisitos para el conjunto de factores de seguridad que componen la seguridad para los SW (autenticación, autorización, confidencialidad, integridad, privacidad, disponibilidad, no repudio, auditoria de seguridad, seguridad perimetral, gestión de la confianza, federación y fiabilidad de mensajes) definidas a partir de los mecanismos de seguridad definidos en las especificaciones de seguridad para SW [115] y las líneas de investigación abiertas por la comunidad científica en relación a la seguridad de los SW.

Esta doctoranda no participó en la definición de este último catálogo puesto que formaba parte de la tesis doctoral de Carlos Gutiérrez [114], sino que colaboró en la aplicación del método SIREN para la fase de requisitos de un proceso completo que permite el desarrollo de sistemas software seguros basados en SW.

Siguiendo a Toval *et al.* [340], en este apartado sintetizamos los antecedentes, objetivos y principales resultados de los primeros trabajos con los catálogos *Seguridad-PDP* y *Seguridad-MAGERIT* en los cuales esta doctoranda ha participado.

4.4.1.1 Antecedentes

En el campo de la seguridad se puede distinguir entre el concepto de seguridad propiamente dicho (en inglés, *security*: la capacidad de un sistema para gestionar, proteger y distribuir información sensible) y lo que llamaremos seguridad respecto a terceros (en inglés, *safety*: la ausencia de consecuencias catastróficas para el entorno) [362]. En los últimos años ha habido un interés creciente en la seguridad de los sistemas de información (en cuanto a *safety* y *security*), que están acechados por múltiples amenazas de distinta naturaleza. Por citar sólo algunas, (1) la falta de adecuación a la legislación vigente (como a la LOPD, que en nuestro país puede acarrear importantes sanciones administrativas, o multas de hasta €750.000); (2) la inexistencia de áreas de protección contra incendios; (3) la ausencia de SAIs (Sistemas de Alimentación Ininterrumpida) en los servidores; e incluso (4) se puede considerar como amenaza la huelga o marcha del personal estratégico de la organización. Es evidente que la aceptación generalizada del comercio electrónico y de la firma electrónica para transacciones administrativas y/o comerciales requiere un aumento de la confianza en la seguridad de los sistemas de información.

Esta situación ha promovido la aparición de numerosas normas y métodos relacionados con la seguridad informática, entre las que destacamos el ISO 15408, *Common Criteria Framework* (CCF) [139], y los métodos CRAMM (en el Reino Unido),

MARION (en Francia) y MAGERIT [202], la Metodología de análisis y gestión de riesgos de la administración pública española, que es una adaptación de CCF. Aunque la necesidad de seguridad afecta a todas las formas de información y a sus soportes o bien a cualquier método usado para transmitir conocimiento, datos e ideas, MAGERIT se limita a considerar la seguridad de la información de los sistemas de información que se incluye en bases de datos, cintas o disquetes, dejando al margen flujos de información informales como por ejemplo las conversaciones.

La práctica habitual en las primeras etapas del desarrollo de sistemas de información ha prestado poca atención a la seguridad, que se ha considerado, en el mejor de los casos, en el diseño, implementación o pruebas, e incluso a veces sólo cuando el sistema ha sido totalmente construido. Sin embargo, la seguridad es barata a corto plazo: cuanto más temprano se actúe para dar seguridad a un sistema de información, más sencilla y económica resultará la seguridad en la organización [49].

En este apartado presentamos una experiencia para considerar la seguridad de un sistema de información desde el proceso de IR, presentando un enfoque que se basa en la reutilización de requisitos procedentes de dos catálogos de requisitos de seguridad: (1) *Seguridad-MAGERIT*, procedente de MAGERIT 1.0 (y por tanto compatible con un subconjunto de CCF); y (2) *Seguridad-PDP*, procedente de la LOPD y el RMS. Hablamos de “experiencia” y no de “caso de estudio” pues en el proceso de creación de los catálogos no se realizó la recogida sistemática de unas variables definidas con antelación. Con este enfoque la elección de un requisito de seguridad implica la eliminación de un riesgo en el futuro sistema, garantizándose que un sistema de información que incorpore nuestros requisitos de seguridad superará con éxito un estudio de riesgos realizado con MAGERIT (que incorpora la LOPD).

Este enfoque basado en los catálogos de seguridad surge a partir de un contrato del GIIS con la CARM (*Comunidad Autónoma de la Región de Murcia*), denominado proyecto CARMMA [13], con el objetivo de realizar un análisis de riesgos en la Dirección General de Sistemas de Información y Telecomunicaciones. Dos de las principales conclusiones del estudio fueron las siguientes: (1) el costo de implementar las salvaguardas para reducir el riesgo de la mayor parte de los activos críticos identificados habría sido menor si estos activos se hubieran desarrollado desde el principio teniendo en cuenta cuestiones de seguridad; (2) aunque las salvaguardas en MAGERIT están ligadas a amenazas, la implementación de cada salvaguarda depende del activo específico amenazado, y por tanto las salvaguardas deberían estar ligadas a activos. Estas dos conclusiones nos llevaron a la definición de un catálogo de seguridad, *Seguridad-MAGERIT*, estructurado de acuerdo con la jerarquía de activos de MAGERIT. Este catálogo ayuda a hacer explícitas las cuestiones de seguridad desde los pasos iniciales en el proceso de desarrollo del sistema. Para profundizar en las cuestiones relativas a la privacidad, el desarrollo de este catálogo fue seguido por el de otro catálogo relacionado con la protección de datos personales, *Seguridad-PDP*.

4.4.1.2. La seguridad y la Ingeniería de Requisitos

Como hemos dicho, la motivación esencial de esta propuesta es la incorporación explícita de la seguridad en el desarrollo de sistemas de información, desde el proceso de IR, de forma que se mejore la calidad. En este sentido, Sommerville [327] afirma que un

buen número de funcionamientos anómalos relacionados con la seguridad del sistema se deben a errores de especificación más que a errores de diseño. Por ejemplo, la especificación de requisitos puede ser incompleta y no describir la conducta del sistema en ciertas situaciones críticas. Por otro lado, en un estudio sobre sistemas empotrados, Lutz [198] concluye que "...la causa principal de los errores del software relacionados con la seguridad a terceros que han persistido hasta la integración y prueba del sistema son problemas con los requisitos".

Existe un número incipiente de iniciativas para incluir la seguridad dentro del ciclo de vida. Este número se reduce si consideramos sólo las iniciativas que incluyen la seguridad en el proceso de IR. Notables excepciones son la aproximación de Chung [64] y la integración de SSADM con CRAMM y de Métrica 3 con MAGERIT 1.0. Estas integraciones ligán expresamente el desarrollo del sistema de información (incluyendo la especificación de requisitos) con la seguridad, aunque no proporcionan nada parecido a un repositorio con requisitos reutilizables de seguridad.

Los estándares de seguridad tampoco soportan adecuadamente el proceso de IR. Creemos que parte del estudio de Fenton y Neil [86] sobre las carencias de los estándares de seguridad a terceros se puede aplicar también a los estándares de seguridad (en particular a ISO 15408): (1) los requisitos que imponen los estándares son imprecisos; y (2) los estándares son demasiado complejos y difíciles de utilizar. De hecho, en dicho estudio se propone un proceso de mejora de los estándares de seguridad a través de la mejora en la organización y escritura de los requisitos que contienen. Además, Fenton y Neil afirman que los estándares de seguridad a terceros están concebidos para evaluar sistemas ya creados, y no hacen hincapié en cómo dotar de seguridad a sistemas de nueva creación. En esta línea, pero en el campo de la protección de datos personales, Breaux y Antón [38] afirman que los requisitos de seguridad y privacidad han de estar alineados con las leyes y regulaciones existentes, que sin embargo describen los derechos y obligaciones en un lenguaje legal complejo y algunas veces ambiguo. Antón ha realizado un extenso trabajo en el campo de los requisitos de protección de datos personales [23], incluyendo la definición de un método para extraer requisitos a partir de regulaciones legales [38].

En otra línea de trabajo, Mellado *et al.* [211] incorporan la seguridad en el paradigma de líneas de productos desde el proceso de requisitos, para lo cual proponen un modelo de variabilidad de seguridad y un modelo de decisión dirigido por estándares de seguridad como ISO/IEC 27001 [140] e ISO/IEC 15408 [139].

Creemos que todos estos trabajos respaldan el interés de desarrollar sendos catálogos de requisitos de seguridad reutilizables, compatibles con MAGERIT por un lado y con la LOPD y el RMS por otro, de manera que SIREN se pueda aplicar en el campo de la seguridad de los sistemas de información.

4.4.1.3. MAGERIT, la LOPD y el RMS como fuentes de requisitos de seguridad

MAGERIT [202] es la Metodología de análisis y gestión de riesgos de sistemas de información de la administración pública española, compatible con ISO 15408 [139], *Common Criteria Framework*. Para MAGERIT, el objetivo de seguridad de los sistemas de información consiste principalmente en mantener la continuidad de los procesos de la organización que soportan dichos sistemas de información. Asimismo, se intenta

minimizar tanto el coste global de la ejecución de dichos procesos como las pérdidas de los recursos asignados a su funcionamiento.

A principios de los 70, surgió en EE UU la primera generación de métodos para analizar los riesgos de los sistemas de información, básicamente listas de comprobación y modelos muy rudimentarios causa-efecto. A partir de mediados de los 80 surgió una segunda generación de métodos, más formalizados, encabezados por la británica CRAMM (*CCTA Risk Analysis and Management Method*). Desde finales de los 90 ha comenzado una tercera generación de métodos más relacionados con la legislación y normativa en materia de seguridad y más ligados a los métodos de desarrollo de sistemas de información. Es el caso de la nueva versión de CRAMM con SSADM o de la primera versión de MAGERIT con Métrica 2.1 y 3 [201]. En estos métodos de tercera generación se consideran los problemas organizativos de modo que se reconcilian la seguridad y funcionalidad del sistema, ya que se tienen en cuenta los aspectos de seguridad desde las primeras fases del proceso de desarrollo [27].

MAGERIT se dirige principalmente a sistemas ya existentes, mientras que Métrica es un método de desarrollo de nuevos sistemas de información. Para integrar ambos métodos, Métrica 3 aporta un *interfaz de seguridad* que indica las actividades a realizar para considerar la seguridad en todo el desarrollo.

En realidad, MAGERIT sólo se aplica a una de las fases de la gestión de la seguridad en los sistemas de información: el análisis y gestión de riesgos. A partir de los objetivos, estrategia y política de seguridad de los sistemas de información existentes en la organización, mediante la aplicación de MAGERIT se estudian primero los riesgos que soporta cada sistema de información y su entorno, y después se recomiendan las medidas (*salvaguardas*) que deberían ser adoptadas para registrar, prevenir, impedir, reducir o controlar los riesgos investigados. Estas salvaguardas quedan plasmadas en un *Plan de seguridad de los sistemas de información*, que después es implantado y cuya aplicación se somete a un seguimiento durante el mantenimiento. La aplicación de MAGERIT es cíclica: en la fase de mantenimiento se realizan periódicamente nuevos análisis de riesgos y se proponen nuevas medidas de salvaguarda.

El modelo de análisis y gestión de riesgos de MAGERIT 1.0 ofrece una guía sólida para la obtención de requisitos reutilizables de seguridad para nuevos proyectos. Este modelo incluye: (1) el *submodelo de elementos*, que proporciona las entidades básicas relacionadas con el análisis de riesgos del sistema de información: *activos, amenazas, vulnerabilidades, impactos, riesgos y salvaguardas*, y (2) el *submodelo de procesos*, que describe las etapas del proyecto de seguridad que se va a construir: *Planificación, Análisis de riesgos, Gestión de riesgos y Selección de salvaguardas*.

Los activos son los recursos del sistema de información o relacionados con éste (formando parte de su entorno). Como vemos en la Figura 19, MAGERIT 1.0 organiza los activos en cinco grandes capas o niveles, según sean más o menos tangibles (en MAGERIT 2.0 esta organización de los activos ha cambiado y se corresponde con lo descrito en el “Catálogo de elementos”).

Cada activo se caracteriza por su estado de seguridad, que se concreta estimando los niveles de cuatro estados de autenticación, confidencialidad, integridad y disponibilidad (ACID). Actualmente se tiende a considerar también la auditabilidad y el no repudio (ACIDAN).



Figura 19 Jerarquía de activos de MAGERIT 1.0.

La estructura del catálogo de requisitos reutilizables *Seguridad-MAGERIT* preserva la estructura de los activos de MAGERIT 1.0: (1) en la *Capa de entorno del sistema de información* se incluyen las instalaciones donde descansa el sistema de información, el mobiliario, los suministros, etc.; (2) la *Capa del sistema de información* (que se detalla en la Figura 20) se estructura en tres bloques, que se refinan a su vez: software, hardware y personal; (3) la *Capa de información* incluye la información manejada por el sistema de información: datos de las aplicaciones (incluyendo los informatizados y los resultantes del sistema de información), metadatos (como estructuras de datos o diccionarios de datos), y soportes (tratables informáticamente y no tratables). Es por tanto este nivel el que más interés tiene en relación con la seguridad en bases de datos; (4) la *Capa de funciones de la organización* justifica la existencia de los sistemas de información, dotándoles de finalidad: bienes y servicios producidos, y personal usuario de los servicios; y finalmente (5) la *Capa de otros activos* describe activos que frecuentemente son intangibles y no encajan en las categorías inferiores, como la credibilidad de la organización.

En la Figura 20 cada nivel se refina en *bloques* de activos homogéneos y cada bloque se divide a su vez en bloques más específicos. Las iniciales en la figura (IS, ISHW, ISSW, etc.) se usan para mantener la estructura de la jerarquía de activos de MAGERIT en el catálogo *Seguridad-MAGERIT*.

En la Capa 3 de MAGERIT 1.0 se menciona la protección de datos de carácter personal. Dada la importancia de la protección de datos, los requisitos relacionados con ésta, aun estando muy relacionados con los de seguridad, se consideran dentro del repositorio SIREN como un perfil aparte.

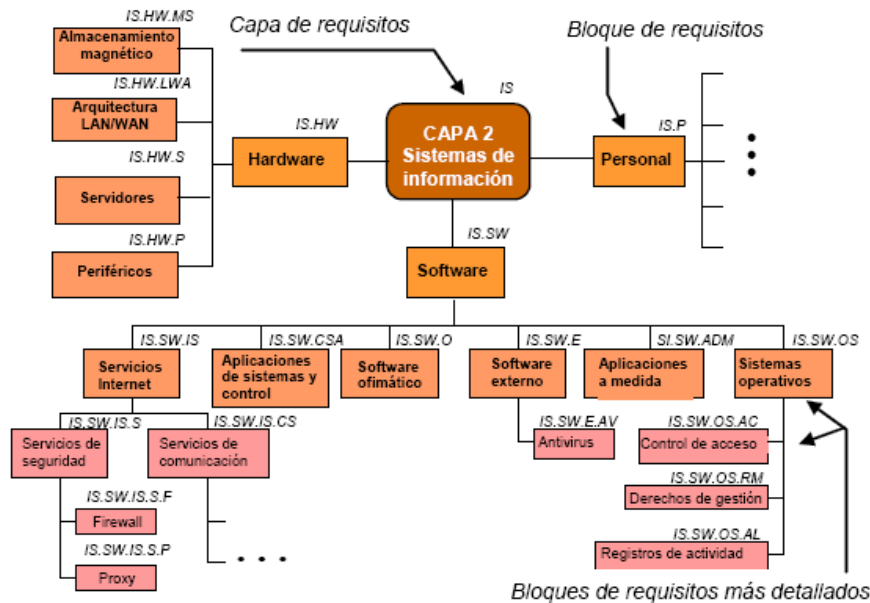


Figura 20 Un extracto de la Capa 2 de Sistemas de información de MAGERIT 1.0.

4.4.1.4. Un ejemplo del catálogo Seguridad-MAGERIT

En la Figura 21 se muestran ejemplos extraídos del catálogo de *Seguridad-MAGERIT*. En dicha figura las etiquetas R1-R4 sólo se utilizan como abreviatura.

Los requisitos en el perfil de seguridad y de la LOPD redefinen el atributo fuente y criticidad de MIN, el conjunto mínimo de atributos de SIREN (ver Sección 4.3.2):

- *fuerza*: en el caso de MAGERIT, el valor de este atributo se corresponde con la posición específica de cada requisito de seguridad en la jerarquía de activos de MAGERIT, esto es, el bloque y la capa (por ejemplo, acrónimos IP, IIGS, IPR, etc. en la Figura 20). De esta manera, el perfil de seguridad dentro del repositorio podría ordenarse de acuerdo a dos criterios: (1) el estándar para la especificación de requisitos; y (2) la estructura de MAGERIT. En el caso de la PDP, este atributo almacena la procedencia del requisito, sea la LOPD o el RMS, y el artículo correspondiente.
- *criticidad*: Proporciona al analista información sobre la necesidad de incluir los requisitos en un proyecto. En el caso de MAGERIT, los posibles valores para este atributo son: *obligatorio*, *recomendable* y *opcional*. La reutilización de los requisitos recomendables y opcionales dependerá del estudio de los riesgos que afectan al sistema de información, más o menos pormenorizado en función del proyecto. En el caso de la PDP, los posibles valores son: *obligatorio* y *recomendable*.

Cabe destacar que aunque los requisitos R4.1 y R4.2 definen condiciones sobre los parámetros estas restricciones no son soportadas por la versión actual de SirenTool (ver Sección 4.6) y serán un aspecto a mejorar en la nueva versión de la herramienta que vamos a desarrollar en el contexto del DSDM.

SyRS (*Especificación de Requisitos del Sistema*)

3. Capacidades del sistema. Condiciones y restricciones.

3.1. Físicas.

3.1.1. Construcción.

R1 (SyRS311S68) Los documentos y disquetes se guardarán bajo llave cuando no se estén utilizando y fuera de la jornada laboral.

3.3 Seguridad del sistema

3.3.1 Requisitos del sistema para alcanzar el nivel básico de seguridad

SyRS331L4. El procedimiento de notificación y respuesta ante incidencias contendrá un registro en el que se haga constar:

SyRS331L4.1 El tipo de incidencia.

SyRS331L4.2 El momento en el que se ha producido la incidencia.

SyRS331L4.3 La persona que realiza la notificación de la incidencia.

SyRS331L4.4 A quién se le comunica la incidencia.

SyRS331L4.5 Los efectos derivados de la incidencia.

3.6. Política y Regulación

R2 (SyRS36S26) Los usuarios autorizados del sistema de información deberán conocer sus obligaciones relacionadas con los controles de acceso y la información que tienen a su cargo. Para ello se establecen tres condiciones:

R2.1 Los usuarios autorizados tendrán que usar su clave de manera adecuada.

R2.2. Los usuarios autorizados no pueden descuidar ni un momento la información que manejan.

R2.3 Los usuarios autorizados tienen que seguir las medidas de seguridad impuestas para evitar accesos no autorizados a la información que manejan.

relacionadoCon: R1, R3 (...)

SRS (*Especificaciones de Requisitos del Software*)

3. Requisitos Específicos.

3.5. Atributos del Software.

3.5.3. Seguridad.

3.5.3.1. Confidencialidad.

R3 (SRS3531S1) El sistema operativo que se utilice proporcionará el mecanismo de claves para controlar y/o limitar el acceso a los usuarios.

relacionadoCon: R4

R4 (SRS3531S14) El sistema operativo que se utilice proporcionará programas para verificar la calidad de las contraseñas en el Sistema de Control de Accesos. Se dice que una clave es de calidad si cumple por lo menos estas tres características:

R4.1 El número mínimo de caracteres es $[n, n \geq 6]$.

R4.2 Tiene al menos $[n, n \geq 1]$ caracteres numéricos y $[m, m \geq 1]$ caracteres alfanuméricos.

R4.3 Se cambia cada $[tiempoEnDias]$ para usuarios generales y cada $[tiempoEnDias]$ para usuarios con privilegios.

Figura 21 Ejemplos de requisitos de seguridad del sistema y del software.

4.4.2. SIREN y los sistemas teleoperados para el mantenimiento de cascos de buques

En esta sección vamos a presentar una línea de investigación en la que esta doctoranda participó al comienzo de su andadura [246] y en la que luego se especializó el Dr. Joaquín Nicolás [245, 247, 248] y culminó con la presentación de su tesis doctoral [245]. Vamos a seguir a Nicolás [245] para la descripción de los resultados alcanzados en esta línea.

Los sistemas teleoperados para mantenimiento de cascos de buques (en adelante, simplemente STO) son sistemas robóticos extremadamente útiles en tareas tales como limpiar o pintar el casco de un barco [89, 262]. Estos sistemas constituyen una línea o familia de productos (ver Sección 1.1.2.1), pues son un conjunto de sistemas intensivos en software que comparten un conjunto de características comunes que satisfacen las necesidades específicas de un segmento del mercado [53]. En los últimos años se han desarrollado arquitecturas software de referencia para los STO (por ejemplo, véase Álvarez *et al.* [22] o Iborra *et al.* [126]).

Estas arquitecturas genéricas están formadas por un conjunto común de artefactos software reutilizables (*assets*) a partir de los cuales se acelera el desarrollo de los STO, que suelen compartir un alto número de comandos de una implementación a otra. Por comandos entendemos básicamente operaciones que cambian el estado del sistema, como por ejemplo operaciones de movimiento de una articulación del robot o de cambio del modo de funcionamiento del sistema. El esfuerzo dedicado a la reutilización de requisitos en los STO ha sido mucho menor que el dedicado a la reutilización de componentes arquitectónicos. En la práctica, en este tipo de sistemas robóticos, las especificaciones de requisitos no han sido desarrolladas rigurosamente con el objetivo de promover la reutilización. En este contexto, nos planteamos la conveniencia de iniciar el desarrollo de nuevos productos dentro de esta línea de productos a partir de un nivel de abstracción más alto –desde los requisitos de la línea de productos en lugar de desde la arquitectura genérica de la misma–, con la obtención de obtener lo que se puede denominar intuitivamente un *modelo del dominio* de los STO. En definitiva, la idea consiste en dotar a los STO de una interfaz que permita un razonamiento más abstracto sobre los productos de la línea.

Los catálogos SIREN desarrollados hasta este momento (seguridad y protección de datos de carácter personal, Sección 4.4.1) pueden considerarse como resultado de un proceso *ad-hoc* de análisis de sus respectivos dominios, ya que ambos perfiles de requisitos se han obtenido sin seguir un método definido. Esto ha sido posible porque los dominios están muy bien estructurados, al considerar como fuente de los requisitos documentos legislativos, como la LOPD o el RMS, o métodos bien definidos como MAGERIT. Por este motivo fue relativamente directo estudiarlos y estructurar su conocimiento en catálogos de requisitos. Pero, ante un dominio vertical complejo como el de los STO nos planteamos si el método SIREN proporciona las técnicas más adecuadas para realizar un análisis de este dominio: (1) por un lado, un riesgo en la elaboración mediante SIREN de un catálogo de requisitos reutilizables para un dominio amplio y complejo como el de los STO es que se puede obtener un catálogo preciso y correcto pero difícil de manejar, al estar formado por largas *listas de la compra* de requisitos textuales, relacionados mediante trazas; y (2) por otro lado, sabemos que existen técnicas

gráficas más potentes para capturar la variabilidad de una línea de productos (básicamente las cuestiones comunes y variables de los STO) que los requisitos textuales en lenguaje natural (por ejemplo [44, 276, 305]). En conclusión, el ingeniero de requisitos necesita asistencia metodológica en el proceso de análisis del dominio, ya que en este contexto, una aproximación *ad-hoc* no es suficiente para obtener un catálogo de requisitos de calidad.

En consecuencia, surge la necesidad de extender el método SIREN con nuevas técnicas que permiten capturar de forma más expresiva la variabilidad de los requisitos de la línea de productos. Así pues, el modelo de procesos, las técnicas y las guías de SIREN se extienden para dar soporte al análisis del dominio de los STO, dando lugar a la propuesta que denominamos *SIRENspl* (donde *spl* procede de *software product line*). Esta propuesta se corresponde con lo que hemos denominado inicialmente como *SIRENp* (Figura 13), el modelo de proceso de SIREN en su faceta *para* reutilización.

Para mejorar la aplicabilidad de la propuesta y la comprensión de sus resultados, en el desarrollo de *SIRENspl* a través del caso de estudio de los STO se eligieron técnicas de análisis del dominio bien conocidas que se ordenaron en un marco conceptual sencillo y se extendieron para dar cuenta de las particularidades del dominio. Estas incluyen:

- *Modelo de clases UML*, que muestra una taxonomía de las entidades del dominio (con sus atributos, relaciones y restricciones de integridad), y un conjunto de eventos del dominio.
- *Máquina de estados UML*, que describe la dinámica de los eventos del dominio.
- *Modelo de características*, que especifica de manera intuitiva la visión de la línea de producto que tienen las personas interesadas en el dominio (*stakeholders*).
- *Modelo de casos de uso genérico*, que describe las interacciones entre los actores y el sistema.
- *Requisitos textuales*, que detalla las características que son especialmente complejas y que implican una cantidad de información tan grande que no es adecuada incluirla en la plantilla de la característica.
- *Especificaciones de comandos*, que se corresponden con las pre y post-condiciones de los comandos utilizados en la implementación de los STO.
- *Plantillas de atributos de calidad*, que modelan los requisitos no funcionales y que sirven para enlazar los requisitos con el diseño de la arquitectura

El lector puede consultar el documento de la tesis doctoral de Dr. Joaquín Nicolás [245] si desea profundizar en el proceso seguido para la elaboración de los modelos y el resultado final de dicho proceso.

El modelo del dominio de los STO obtenido sirve para: (1) acelerar la comprensión de la línea de productos (indicando sus elementos comunes y variables); (2) obtener en contexto los requisitos reutilizables de la línea de productos; (3) tomar decisiones de forma eficiente sobre la especificación de los nuevos sistemas desarrollados dentro de la línea de productos; y (4) reutilizar la arquitectura de referencia y los comandos utilizados en la implementación de la línea de productos.

4.4.3. SIREN en casos de estudios reales: Proyecto GARTIC

En el marco del proyecto GARTIC, descrito en la Sección 1.3.3.1, se han desarrollado cinco casos de estudio, uno por cada una de las pequeñas y medianas empresas de desarrollo de software participantes: *Bahia IT S.A.*, *Foro Digital S.L.*, *Global Software*, *Adalid Management & Outsourcing S.L.*, y *S.Q.A.* Esta doctoranda estuvo involucrada como tutora en dos de ellos correspondientes a las empresas, a las que por motivos de confidencialidad, hacemos corresponder las etiquetas E1 y E2.

El proyecto GARTIC se inició con un curso de formación en IR, SIREN y la herramienta CARE comercial Requisite Pro, y continuó con la implantación en dos fases de SIREN: (1) en la primera, se hizo una identificación de dominios de aplicación candidatos de interés para las empresas participantes (verticales y horizontales). Cada empresa designó un analista responsable del desarrollo de los proyectos. Después de seleccionar un dominio, se implantó independientemente SIREN *para* reutilización en cada empresa, y se obtuvo un catálogo de requisitos para cada dominio seleccionado, en un proceso que como mínimo involucró una iteración entre los analistas de las empresas y los tutores del GIIS; (2) en la segunda fase, se implantó SIREN *con* reutilización, desarrollando un proyecto a partir de los documentos de requisitos especificados en la primera fase, es decir, instanciando el catálogo de requisitos, al menos una vez, haciendo uso de la nueva versión de SirenTool (Sección 4.6). En las siguientes tablas (Tabla 30 y Tabla 31) se muestra un resumen de los proyectos piloto, centradas en la primera fase, para reutilización.

EMPRESA-TIC	E1
Catálogo	<i>Armorknown</i> , requisitos reutilizables de todos los módulos implicados en una aplicación TPV (<i>Terminal de Punto de Venta</i>)
Tipo	Catálogo de Dominio (vertical)
Propósito	Documentar un desarrollo ya elaborado.
Método/técnicas de análisis que proponen usar	Diagramas de contexto.
Documentos desarrollados en el catálogo	SRS
Estado del catálogo	Tres iteraciones sobre el catálogo.
Número de requisitos identificados	277 requisitos.

Tabla 30 Resumen proyecto piloto GARTIC empresa E1 en el desarrollo *para* reutilizar.

En el proyecto GARTIC se han desarrollado verdaderos casos de estudio, pues desde el principio del proyecto se había establecido un conjunto de medidas con el objetivo de cuantificar la experiencia. ECAPRIS (*Evaluación de la Calidad y la Productividad al implantar un método de Ingeniería de Software*), es un método de medición ágil desarrollado por Duque *et al.* [78], basado en GQM (*Goal Question Metric*) y CMMI, que tiene por objetivo evaluar la mejora en calidad y productividad que conlleva la implantación de nuevos métodos de Ingeniería de Software en PYMEs. ECAPRIS se ha aplicado en GARTIC en las dos fases de desarrollo *para* y *con* reutilización, con el

objetivo de evaluar las variaciones en la calidad y la productividad del proceso de gestión de requisitos y del producto software al implantar SIREN para producir un catálogo de requisitos y al menos un documento de requisitos específicos. Los resultados que Duque *et al.* [78] han medido en los proyectos piloto de las empresas E1 y E3 se muestran en la Tabla 32.

EMPRESA-TIC	E2
Catálogo	<i>DataTransformer</i> , requisitos reutilizables de un traductor genérico de cualquier tipo de dato fuente a cualquier tipo de dato destino.
Tipo	Catálogo de Dominio (vertical)
Propósito	Especificar los requisitos de una aplicación en curso.
Método/técnicas de análisis que proponen usarr	Casos de uso.
Documentos desarrollados en el catálogo	SRS
Estado del catálogo	Una iteración sobre el catálogo
Número de requisitos identificados	50

Tabla 31 Resumen proyecto piloto GARTIC empresa E2 en el desarrollo para reutilizar.

El bajo nivel de madurez en cuanto al proceso de IR de las empresas participantes ha determinado que el proyecto se haya centrado no sólo en transferir un método de IR basado en reutilización, sino también un proceso de IR básico. Las empresas no tenían experiencia con métricas del software, de manera que el proyecto también les ha servido para iniciarse en este campo. Todas las empresas entregaron los sucesivos cuestionarios y rellenaron la información de uso de la herramienta fuera de plazo.

Al implantar SIREN con reutilización, existe un consenso en las empresas participantes en mencionar beneficios como los que destacamos a continuación: (1) la utilización del catálogo de requisitos facilita la estimación del esfuerzo de sucesivos proyectos, mejorando de esta manera las estimaciones de tiempo y recursos; (2) el incremento en el uso de estándares de requisitos y el mejor conocimiento del dominio de aplicación redunda en mejoras en la documentación, trazas de requisitos y comunicación con el cliente; (3) la reducción de las desviaciones y la disminución del número de inconsistencias entre requisitos; y (4) la mayor agilidad en la validación, que permite que los requisitos se ajusten más rápidamente a las expectativas de los clientes.

El proceso de implantación de SIREN en las empresas ha permitido refinar las características de SirenTool (Sección 4.6), que ha sido revisada completamente e implantada en las empresas en la fase de SIREN *con* reutilización, obteniéndose así una útil retroalimentación. Para una discusión más detallada de los resultados cuantitativos y cualitativos de este proyecto remitimos a Duque *et al.* [78].

RESULTADOS			EMPRESAS	
			E1	E2
SIREN para reutilización				
M1	# de req. del catálogo		277	50
M2	Esfuerzo de creación del catálogo (H/h)		300	12
M3	Esfuerzo por req. (H/h)	= M2 / M1	1.08	0.24
SIREN con reutilización				
M4	# de req. Reutilizados sin modificación		120	25
M5	Esfuerzo en inserción de requisitos (H/h)		6	15
M6	# de req. reutilizados con modificación		10	12
M7	Esfuerzo en modificación (H/h)		3	5
M8	# de req. parametrizados sin modificar		5	8
M9	Esfuerzo de inserción de requisitos parametrizados (H/h)		2	1
M10	# de req. nuevos		245	30
M11	Esfuerzo en creación (H/h)	= M3 * M10	176	40
M12	% del catálogo reutilizado	= (M4 + M6 + M8) / M1	49%	90%
M13	% de requisitos reutilizados	= (M4 + M6 + M8) / (M4 + M6 + M8 + M10)	36%	60%
M14	% de esfuerzo de reutilización de requisitos	= (M5 + M7 + M9) / (M5 + M7 + M9 + M11)	6%	34%
M15	% de requisitos reutilizados sin modificaciones	= (M4 + M8) / (M4 + M6 + M8 + M10)	33%	44%
M16	Esfuerzo no invertido (H/h)	= (M4 + M6 + M8) * M3	146.0	10.8

Tabla 32 Resultados de las empresas E1 y E2 en GARTIC (H/h: Horas/hombre).

4.5. Ocho cuestiones clave en reutilización de requisitos

La experiencia obtenida en la definición y puesta en práctica del método SIREN (resumida en la Sección 4.4), junto con la investigación realizada a lo largo de los años en el campo de la reutilización de requisitos y especialmente de las herramientas CARE, nos ha llevado a formular en el GIIS ocho cuestiones clave que, en nuestra opinión, deberían ser soportadas de una u otra manera por cualquier método de IR basado en reutilización. Estas cuestiones clave se presentan en Toval *et al.* [339] y son las mostradas en la Tabla 33.

Las ocho cuestiones clave sintetizan una suerte de lecciones aprendidas en las experiencias mostradas en la Sección 4.4, y han servido (1) para reflexionar sobre las principales aportaciones y los aspectos más destacados de la investigación en IR y reutilización presentada en este capítulo; y (2) para identificar las principales cuestiones relacionadas con la reutilización que deberían ser soportadas por un método de IR

basado en reutilización y en particular por una herramienta CARE que soportara un método de IR basado en reutilización. Estas ocho cuestiones clave han sido fundamentales a la hora de establecer un marco para revisar el estado del mercado de herramientas CARE en relación con la reutilización, como se muestra en la Sección 4.6.1. Se trata de cuestiones sencillas dado que uno de los objetivos de la línea de investigación de IR y reutilización, es acercar el estado del arte y el estado de la práctica. Con la propuesta de un método práctico y sencillo, y a través de la satisfacción de estas cuestiones clave, pensamos que una empresa con un nivel bajo de madurez puede dar un primer paso en la mejora de su proceso de IR.

<i>K1. Organización de los requisitos reutilizables</i>
<i>K2. Motor de búsqueda de requisitos reutilizables</i>
<i>K3. Selección y reutilización de requisitos con distintos niveles de granularidad</i>
<i>K4. Reutilización de atributos de los requisitos</i>
<i>K5. Reutilización de relaciones de traza</i>
<i>K6. Gestión de requisitos parametrizados</i>
<i>K7. Mejora del repositorio</i>
<i>K8. Soporte CARE para la reutilización</i>

Tabla 33 Ocho cuestiones clave en reutilización de requisitos.

Se trata de cuestiones clave que en todos los casos están íntimamente ligadas al soporte automatizado del método de IR que contempla reutilización (especialmente K2, K3 y K8). En Toval *et al.* [339] se muestra un estudio de tres de las principales herramientas CARE comerciales, que revela una carencia de soporte automatizado a estas ocho cuestiones, lo cual nos ha llevado a proponer una herramienta propia de soporte de SIREN, SirenTool, y a mostrar cómo esta herramienta soporta cada una de las cuestiones clave. Un resumen de este estudio se presenta en la Sección 4.6.1.

Estas ocho cuestiones clave y la viabilidad de la herramienta desarrollada han sido evaluadas en el contexto de un caso de estudio para el desarrollo de un sistema de información para la gestión de historias clínicas en la unidad de cuidados intensivos del Hospital de Getafe en Madrid, en colaboración con Manuel Campos, miembro del equipo de desarrollo de dicho proyecto de transferencia tecnológica [14].

4.5.1. Organización de los requisitos reutilizables (K1)

Compartimos la visión de Lam *et al.* [174] en el sentido de que la definición de requisitos reutilizables se beneficia de disponer de una guía para estructurar el dominio y para organizar los requisitos reutilizables. Para facilitar la reutilización es fundamental que los requisitos reutilizables estén bien organizados y estructurados, pues si es así el tiempo para extraer requisitos del repositorio se reduce.

En SIREN, por ejemplo, hemos elegido organizar los requisitos reutilizables por medio de catálogos que usan unas plantillas de documentos predefinidas. Un catálogo agrupa un conjunto de requisitos relacionados que se corresponde con un dominio de aplicación o con un perfil. Otras estructuras también sirven para ejemplificar K1: libros

de patrones (*pattern book*) [292], casos de uso [18] o mapas de estructura (*structure maps*) [272].

4.5.2. Motor de búsqueda de requisitos reutilizables (K2)

Un método de IR basado en reutilización debe tener disponible un motor de búsqueda fácil de usar que satisfaga todas las necesidades de consulta de los ingenieros de requisitos en tiempo de reutilización. La capacidad de seleccionar los requisitos más adecuados para un nuevo proyecto es fundamental. El ingeniero de requisitos debe ser capaz de definir un amplio rango de criterios de búsqueda para realizar la selección de requisitos reutilizables. Esta cuestión clave está directamente ligada con la cuestión clave K8, “Soporte CARE para la reutilización”, pues el motor de búsqueda debería estar implementado en el contexto de una herramienta de gestión de requisitos en la cual se debe proporcionar también la capacidad de seleccionar requisitos a partir del conjunto resultado de la búsqueda.

Por ejemplo, en el caso de utilizar plantillas de documentos, como en SIREN, puede ser necesario buscar los requisitos especificados en una sección particular del documento o aquellos requisitos relacionados con una determinada incumbencia de los interesados (como por ejemplo, “cuestiones legales”). También puede ser necesaria la búsqueda de requisitos que contengan una palabra específica, como los requisitos que contengan la palabra “firewall” en el catálogo *Seguridad-MAGERIT*.

4.5.3. Selección y reutilización de requisitos con distintos niveles de granularidad (K3)

Se debe poder definir en tiempo de reutilización el tamaño de grano de lo reutilizado en el nivel de requisitos. El conjunto de requisitos reutilizados debería poder estar formado por un único requisito, por un subconjunto de requisitos procedentes de algunos de los catálogos disponibles, o por un catálogo completo.

Existen aproximaciones a la reutilización de requisitos en las que la granularidad es fija. Por ejemplo, aquellas basadas en un documento base de requisitos. Esta clase de reutilización es menos eficiente, pues es obligatorio reutilizar un conjunto determinado de requisitos, cuando sólo se requiere un subconjunto de los mismos. Si no se permiten distintos niveles de reutilización, la reutilización se hace menos eficiente pues se debe dedicar un tiempo a la eliminación de requisitos que no son necesarios para el nuevo proyecto.

4.5.4. Reutilización de atributos de los requisitos (K4)

Un requisito reutilizable debe estar descrito por medio de un conjunto de atributos. Estos atributos se deben manejar cuidadosamente en tiempo de reutilización, de manera que cuando un requisito es reutilizado sus atributos también lo sean. El ingeniero de requisitos debe tener disponible la flexibilidad de poder modificar cualquier atributo en tiempo de reutilización. Cuando el ingeniero de requisitos decide cambiar el tipo del requisito reutilizado puede aparecer un problema, porque el conjunto de atributos que

pertenece a cada tipo de requisito, origen y destino, puede ser distinto. Por ejemplo, un requisito de protección de datos puede ser reutilizado como un requisito de seguridad en el caso de los catálogos de *SIREN Seguridad-PDP* y *Seguridad-MAGERIT*.

Para ayudar a mantener la consistencia en tiempo de reutilización es necesario definir un conjunto mínimo de atributos que todos los tipos de requisitos deben tener en común. En SIREN se ha definido un conjunto mínimo de atributos, MIN, basado en el estándar IEEE 1233 [128] (Sección 4.3.2). Por otro lado, el valor de los atributos de los requisitos puede variar de un proyecto a otro pero la lista de atributos englobados por un tipo de requisito ha de permanecer en tiempo de reutilización. Posteriormente, a través del desarrollo del proyecto actual, se pueden definir nuevos atributos para ese tipo de requisito.

4.5.5. Reutilización de relaciones de traza (K5)

Normalmente los requisitos reutilizables no están aislados sino que están relacionados unos con otros por medio de trazas. Obviamente la existencia de estas trazas se debe tener en cuenta en tiempo de reutilización. En tiempo de reutilización el ingeniero de requisitos debe conocer las relaciones de traza entre requisitos de manera que pueda decidir qué requisitos deben permanecer en el nuevo proyecto. Es útil que el soporte automatizado permita que estas relaciones de traza se puedan visualizar gráficamente de forma adecuada, de manera que puedan ayudar en el proceso de toma de decisiones en tiempo de reutilización. Además, cuando la especificación de requisitos evoluciona, las relaciones de traza pueden ser utilizadas para evaluar el impacto del cambio de un requisito en los requisitos que dependen de él. La gestión de cambios se puede beneficiar así de una adecuada gestión de las trazas.

Ramesh y Jarke [281] proponen una amplia taxonomía de trazabilidad, pero (1) como afirman von Kethen *et al.* [171], no indican qué tipo de dependencias deben ser trazadas para soportar reutilización; y (2) en nuestra opinión se trata de una taxonomía demasiado compleja para ser utilizada en un proceso de IR práctico como el que se pretende definir con SIREN. Por lo tanto, proponemos un conjunto reducido de relaciones de traza horizontales, que son aquellas trazas que relacionan artefactos en el mismo nivel de abstracción, en este caso, en el nivel de los requisitos. Los requisitos relacionados pueden ubicarse en el mismo documento (por ejemplo, el SRS) o en distintos documentos (por ejemplo, consideremos un requisito del SYRS que se relaciona con un requisito del SRS). El modelo de traza propuesto para SIREN se presenta en la Sección 4.3.4. Este modelo de traza revisa el de Toval *et al.* [339] a través de la experiencia del proyecto GARTIC (Sección 4.4.3).

4.5.6. Gestión de requisitos parametrizados (K6)

La parametrización es un concepto que ha sido extensamente utilizado en programación para favorecer la reutilización. En el nivel de los requisitos, dejando al margen aproximaciones formales como por ejemplo la de OBJ3 [105] (un lenguaje basado en reescritura de términos), la parametrización ha sido trasladada en forma de: (1) características parametrizadas (*parameterized features*) como las de FODA [158] y FORM [159]; y especialmente en (2) requisitos parametrizados (*parameterized*

requirements) (ver por ejemplo [92, 174]). Los requisitos parametrizados contienen parámetros, es decir, partes que deben ser adaptadas a cada aplicación o sistema y a las que se debe dar valores en tiempo de reutilización o en tiempo de especificación.

Los requisitos parametrizados promueven la reutilización factorizando detalles específicos de la definición del sistema en forma de parámetros del requisito, de manera que se permite a los ingenieros de requisitos especificar requisitos con un nivel de flexibilidad mayor. Es interesante además que en tiempo de reutilización el ingeniero de requisitos disponga de un conjunto de valores adecuados. El modelo de requisitos parametrizados propuesto para SIREN se ha presentado en la Sección 4.3.3. De nuevo este modelo revisa el de Toval *et al.* [339] a través de la experiencia del proyecto GARTIC (Sección 4.4.3).

4.5.7. Mejora del repositorio (K7)

El repositorio no se debe considerar un producto estático sino evolutivo, dado que uno de los beneficios de la reutilización es la mejora continua de la calidad de los productos reutilizados. A través de la experiencia que se gana con su uso, el repositorio se mejora (1) para mejorar la calidad de los requisitos reutilizables en él incluidos, mejorando su adecuación, solventando ambigüedades, incompletitudes y errores en su especificación, añadiendo nuevas relaciones de traza, nuevos posibles valores para los parámetros, etc.; y (2) para extender el contenido con nuevos requisitos reutilizables. Para permitir esta evolución se debe definir un proceso de gestión de la configuración del repositorio, que implemente en detalle la actividad T8 de SIREN, “Mejora del repositorio” (Sección 4.2.4.8).

4.5.8. Soporte CARE para la reutilización (K8)

Sin soporte automático, la aplicación de cualquier método de IR que contemple la reutilización de requisitos se transforma en lenta y engorrosa, y por tanto, impráctica. Varias de las cuestiones clave anteriores (a saber, K2, K3, K4, K5 y K6) sólo tienen verdadero sentido en un proceso que está respaldado por un soporte automático. Firesmith [91] considera la reutilización de requisitos una propiedad crítica en las herramientas de requisitos. Sin embargo, como se muestra en la Sección 4.6.1, esta propiedad no se soporta suficientemente en las principales herramientas CARE comerciales. Esta es la razón que nos ha llevado a desarrollar SirenTool.

4.6. Herramienta de soporte: SirenTool

La herramienta de soporte a SIREN se denomina SirenTool. En [177, 180, 339] se encuentran referencias a un prototipo inicial en cuya definición ha participado esta doctoranda y que ha sido desarrollado por el actual Dr. Joaquín Laceras también miembro del GIIS. Posteriormente, a lo largo del proyecto GARTIC Joaquín Lasheras revisó completamente este prototipo inicial y desarrolló una nueva versión de la herramienta, SirenTool 2.0.

4.6.1. La reutilización en las herramientas CARE comerciales

Lasheras *et al.* [180] y Toval *et al.* [339] muestran sendos estudios sobre cómo tres de las principales herramientas CARE comerciales soportan las características básicas de reutilización. En esta sección resumimos estos estudios. Las herramientas escogidas para la comparación han sido DOORS, Requisite Pro y Caliber-RM. Se han escogido estas tres herramientas por (1) ser herramientas comerciales bien conocidas (especialmente las dos primeras); (2) aparecer en todas las comparaciones de herramientas que conocemos (INCOSE [136], Volere [356], Alenxander [17] y Ludwing Consulting Services [10]); (3) ser las únicas referenciadas en un interesante artículo de Kaindl *et al.* [151] sobre el estado de la IR; y (4) proporcionar un interfaz de extensibilidad que permite su adaptación.

En la comparación se han adoptado dos aspectos: (1) las ocho cuestiones clave descritas en la Sección 4.5; y (2) las necesidades generales de una herramienta de gestión de requisitos. La Tabla 34 resume las características consideradas necesarias para nuestra herramienta de gestión de requisitos. La tabla sólo muestra las cuestiones clave para la reutilización pues las tres herramientas satisfacen las características esenciales de una herramienta de gestión de requisitos, como muestra el *survey* de INCOSE sobre herramientas CARE [136]. La cuestión K8, “Soporte CARE para la reutilización”, no se muestra en la tabla pues su valoración se deriva de las siete anteriores.

Cuestión clave	REQUISITE PRO	CALIBER-RM	DOORS
<i>K1. Organización de los requisitos reutilizables</i>	Sí, por tipo de requisito		Sí, por tipo de objeto
<i>K2. Motor de búsqueda de requisitos reutilizable</i>	Los requisitos no pueden ser seleccionados a partir del resultado de las búsquedas		
<i>K3. Selección y reutilización de requisitos con distintos niveles de granularidad</i>	Copiar y pegar Documentos base		Copiar y pegar Módulos de req.
<i>K4. Reutilización de atributos de los requisitos</i>	Cuando el requisito se copia y pega		
<i>K5. Reutilización de relaciones de traza</i>	Sólo trazas padre-hijo	No	
K5.1 Requisitos exclusivos	No		
K5.2 Muestra relaciones de traza	Sí, gráficamente		
K5.3 Gestión de inconsistencias	Sí, matrices de trazas		
<i>K6. Gestión de requisitos parametrizados</i>	No		
<i>K7. Mejora del repositorio</i>	No se trata explícitamente		

Tabla 34 Comparativa de las herramientas CARE comerciales respecto a las cuestiones claves de reutilización.

Este estudio concluye que ninguna de las tres herramientas proporciona adecuadamente reutilización de requisitos pues ninguna de ellas soporta todas las cuestiones clave para la reutilización de requisitos. La única forma de reutilizar es mediante copia y pega de requisitos ya definidos. Ninguna de las herramientas ofrece

gestión de requisitos parametrizados, reutilización de relaciones de traza, o gestión de requisitos exclusivos. Matulevicius [208] también incide en las limitaciones de las herramientas CARE para soportar reutilización de requisitos.

4.6.2. SirenTool

A partir de las ocho cuestiones clave para reutilización de requisitos y de los resultados del estudio anterior sobre el soporte de reutilización por parte de las principales herramientas CARE comerciales se ha desarrollado una especificación de requisitos del software de una herramienta CARE de soporte al método SIREN [177]. A la luz de esta especificación de requisitos ninguna de las herramientas analizadas en el apartado anterior era adecuada para ser adoptada directamente para soportar reutilización de requisitos, de manera que se consideraron dos opciones para proporcionar un soporte automatizado a SIREN: (1) adaptar una de estas herramientas; o (2) desarrollar una nueva herramienta desde el principio. Se ha preferido la primera opción, pensando que es más efectiva en coste al tiempo que permite satisfacer completamente las necesidades de SIREN. Se ha escogido Requisite Pro como herramienta base para resolver los problemas identificados en la Tabla 34, debido fundamentalmente a la experiencia previa del grupo en el uso de esta herramienta [343, 344].

SirenTool 2.0 consta de la funcionalidad completa de la herramienta Requisite Pro más cuatro nuevas aplicaciones que incluyen funcionalidades relacionadas con la reutilización. Estas cuatro nuevas aplicaciones, a las que llamaremos conjuntamente, SIRENA (*SIREN Automated*), son las siguientes: (1) SIREN CON REUTILIZACIÓN; (2) SIREN PARA REUTILIZACIÓN; (3) INSTANCIAR PARAMETRO; y (4) ESTADISTICAS (Figura 22). Una descripción completa de SirenTool 2.0 queda fuera del ámbito de esta tesis doctoral. Con fines meramente ilustrativos, en este apartado nos limitamos a mostrar una de las principales ventanas de la aplicación, la interfaz de reutilización de requisitos, perteneciente a SIREN CON REUTILIZACIÓN (Figura 23). En Nicolás *et al.* [249] se puede encontrar una guía completa sobre todas las capacidades de esta herramienta.

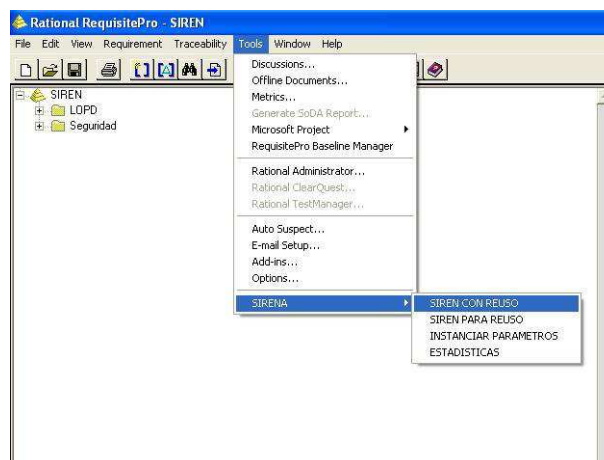


Figura 22 SIRENA como *plug-in* de RequisitePro.

Figura 23 Interfaz para la reutilización de requisitos de SirenTool 2.0.

4.7. SIREN_{MD}: un proceso de Ingeniería de Requisitos dirigido por modelos

La adaptación del método SIREN al DSDM, en lo que vamos a denominar SIREN_{MD} (*SIREN Model-Driven*), no supone cambios metodológicos en cuando al modelo de proceso (Figura 24). Dado que el objetivo final de la fase de IR sigue siendo el mismo, la obtención de la especificación de requisitos, se debe seguir aplicando las tareas y guías propuestas en SIREN (Sección 4.2 y 4.3). El cambio fundamental de SIREN_{MD} respecto a la visión tradicional de SIREN radica en el soporte que se proporciona para la especificación de requisitos. Mientras que SIREN podemos entenderlo como un método centrado en el repositorio y basado en lenguaje textual, la integración de los requisitos en el DSDM supone un giro en esta concepción meramente textual de los requisitos, que pasan ahora a modelarse como el resto de artefactos del desarrollo (ver Sección 1.1.3). Para ello, la propuesta SIREN_{MD} se basa en la definición de un metamodelo de requisitos, que hemos denominado REMM (*Requirements Engineering MetaModel*) que, como veremos en el capítulo siguiente (Capítulo 5), soporta las facetas *para* y *con* reutilización propuestas en SIREN, lo que nos proporciona un enfoque para la reutilización sistemática de requisitos dirigida por modelos. La definición de modelos de requisitos, conformes a un metamodelo como REMM, reporta ventajas sustanciales al proceso de IR como se detalló en la Sección 1.1.3.5. En particular, algunas de las tareas originales de

SIREN (Sección 4.2.2 y 4.2.4) se ven beneficiadas por la integración de las técnicas del DSDM de la siguiente forma:

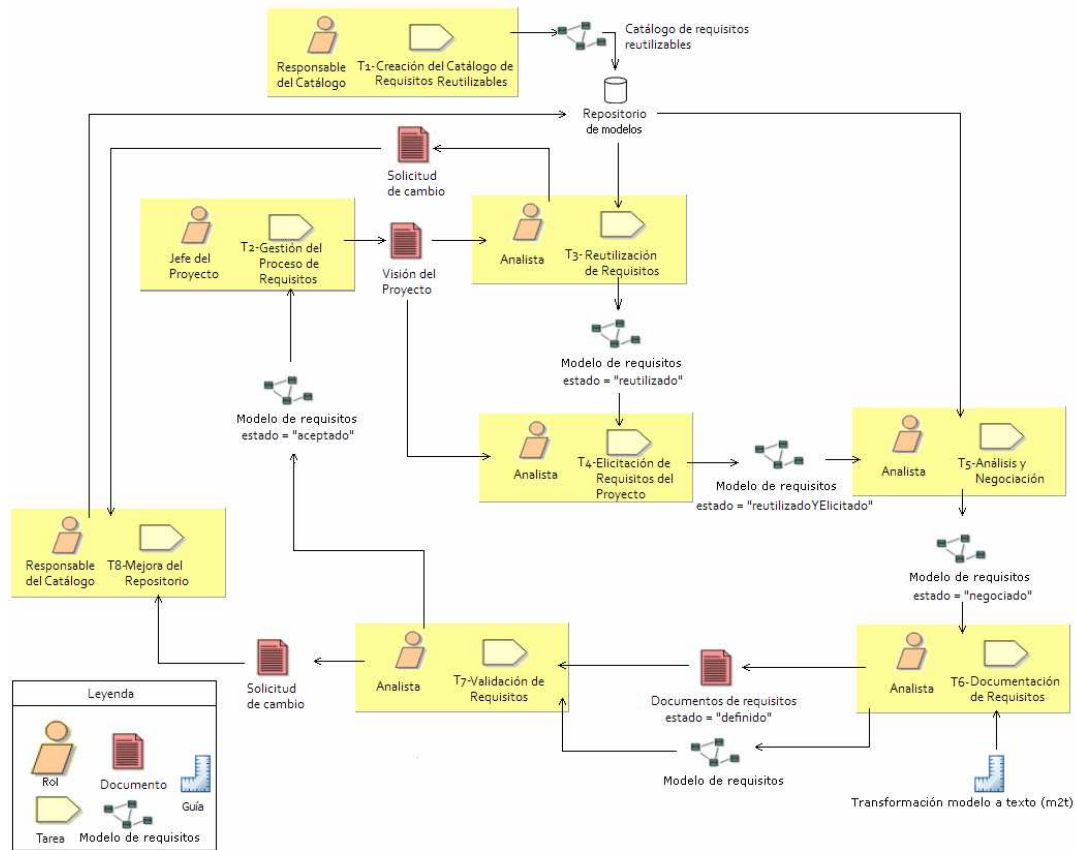


Figura 24 Modelo de proceso para SIREN_{MD}.

- **T3-Reutilización de requisitos.** la definición explícita del modelo de traza tanto entre requisitos como entre los requisitos y el resto de artefactos implicados en el proceso de IR, favorece la reutilización sistemática de requisitos. De manera que, cuando un requisito sea seleccionado para su reutilización en un nuevo proyecto, automáticamente, todos los requisitos y elementos implicados en la definición de este requisito, serán también reutilizados. Estas relaciones de traza pueden definirse y representarse gráficamente mediante un editor de modelos de manera sencilla (la herramienta de soporte será presentada en el Capítulo 7).
- **T6- Documentación de requisitos.** Como apuntábamos en la Sección 1.1.1.1, los modelos no deben reemplazar una especificación de requisitos textuales, dado que el documento de requisitos en este formato es el formato preferido en la industria. Para esta tarea de documentación de los requisitos contenidos en los modelos podemos aprovecharnos de las transformaciones modelo a texto (m2t) que nos permiten generar la especificación de requisitos textual a partir de los modelos de acuerdo a la plantilla más adecuada, por ejemplo IEEE-830 [131]. Además se puede obtener dichos documentos en diferentes formatos, por ejemplo HTML. La generación de la especificación de requisitos en un formato

navegable (HTML) resulta de especial interés para la validación de requisitos puesto que permite navegar por la documentación de requisitos de forma sencilla siguiendo los enlaces. Por ejemplo, se podría seguir todos los requisitos enlazados con una misma palabra del glosario de términos para comprobar si ha habido un entendimiento común en el uso de dicho término.

- *T7- Validación de requisitos.* En SIREN_{MD} debemos diferenciar entre: 1) la *validación de los modelos* de requisitos respecto al metamodelo de requisitos, esto es, la comprobación de que el modelo es conforme a REMM, y 2) la *validación de la especificación de requisitos* [186], esto es, el proceso por el cual se asegura que la especificación de requisitos describe de forma adecuada las capacidades del sistema y que los requisitos son completos, no ambiguos, consistentes, trazables, etc. La ventaja del enfoque SIREN_{MD} es que parte de la validación de la especificación de requisitos se incluye en la validación de los modelos de requisitos, y esta validación de modelos se ejecuta automáticamente por el editor de modelos (ver Capítulo 7). En consecuencia, parte de la validación de la especificación de requisitos se puede soportar de forma automática. Por ejemplo, el editor puede generar de manera única un identificador para cada requisito y además se puede garantizar que cada requisito esté ligado con un caso de prueba. Que cada requisito esté asociado con un caso de prueba puede ser un indicio de validez de la especificación, puesto que si un requisito no está asociado con ningún caso de prueba este requisito puede ser considerado como ambiguo o incompleto. No obstante, es cierto que esta condición de que siempre exista una relación de traza entre un requisito y un caso de prueba es necesaria pero no suficiente puesto que la validación del modelo asegura que la traza existe, pero no comprueba ni la especificación del requisito ni la especificación del caso de prueba que tiene que hacerse mediante sesiones de revisión o inspecciones en las que deben intervenir tanto los analistas como los usuarios clave, tal como se ha definido para el método SIREN tradicional (Sección 4.2.4.7). Como apuntábamos en la tarea anterior (T6), la generación de los documentos de especificación de requisitos en formato navegable puede favorecer el proceso de revisión o inspección manual.

La salida del proceso SIREN_{MD} es un modelo de requisitos que, siguiendo el enfoque DSDM, puede trazarse explícitamente hacia otros artefactos del desarrollo y/o transformarse en otros modelos. La definición de un proceso de IR dirigido por modelos es el primer paso para la integración de los requisitos en un enfoque completo de DSDM utilizando las técnicas propias de este enfoque: metamodelado y transformaciones de modelos. Como veremos en el Capítulo 6, a partir de los modelos de requisitos se pueden definir transformaciones, no sólo m2t (como acabamos de ver para la documentación de requisitos) sino modelo a modelo (m2m) para generar otros modelos del desarrollo y trazar los requisitos con los artefactos de los modelos generados automáticamente.

Como veremos en el Capítulo 7, la nueva propuesta SIREN_{MD} cuenta con una herramienta de soporte que hemos denominado REMM-Studio⁺, que podríamos considerar como una evolución de SirenTool (sección 4.6) hacia el DSDM. De forma resumida, REMM-Studio⁺ da soporte a: 1) la definición de modelos de requisitos tanto en la faceta *para* reutilización como en la faceta *con* reutilización, 2) validación de modelos de requisitos, 3) reutilización sistemática de requisitos, 4) transformaciones m2t para la

generación de documentos de especificación de requisitos, y 5) transformaciones m2m para la generación de modelos de análisis conformes a un DSL.

4.8. Conclusiones y aportaciones al estado del arte

En este capítulo se han recogido los trabajos relacionados con la línea inicial de trabajo en esta tesis doctoral: IR y reutilización, que constituye la base para la línea de investigación propia de esta tesis doctoral, la integración de los requisitos textuales en el DSDM en el contexto de un proceso de IR basado en reutilización. El resultado de la línea de trabajo de IR y reutilización fue la definición del método SIREN, que se ha definido en detalle, incluyendo sus experiencias de aplicación y su herramienta de soporte. Además, hemos visto de manera general como SIREN puede adaptarse para el DSDM, SIREN_{MD}, dejando para capítulos posteriores la explicación detallada de cada una de las técnicas implicadas en el proceso y la herramienta de soporte.

En el ámbito de la reutilización de requisitos, con SIREN se proporciona un método práctico, sencillo y flexible para mejorar el estado de la práctica de IR, que ha sido probado en entornos industriales y que pensamos puede servir como un primer paso para acercar dicho estado de la práctica al estado de la investigación en IR. Buscando flexibilidad, SIREN se concibe como una especie de “*plugin* metodológico” que señala un conjunto reducido de tareas, técnicas y guías que pensamos se pueden extrapolar al marco dado por cualquier método de IR. A partir de la experiencia ganada con SIREN se han identificado ocho cuestiones clave sencillas que en nuestra opinión deberían ser soportadas por cualquier método de IR basado en reutilización y que son de utilidad para evaluar un método o una herramienta de IR basadas en reutilización.

Las aportaciones de SIREN al estado del arte las podemos resumir como sigue:

- Se trata del primer enfoque que conocemos [343] que abordó el problema de la reutilización de requisitos de seguridad y de protección de datos personales utilizando catálogos de requisitos en lenguaje natural especificados a partir de estándares (como la metodología MAGERIT, estándar de la Administración española compatible con el *Common Criteria Framework*) y de textos legales (como la LOPD y el RMS).
- Se trata del primer enfoque que conocemos [180, 339] que proporcionó soporte automatizado a la reutilización a través de una herramienta CARE comercial (en particular, a través de IBM Rational Requisite Pro, aunque el enfoque es trasladable a otras herramientas comerciales). El soporte automatizado a la reutilización de requisitos sigue siendo escaso en la actualidad.
- Se trata del único enfoque que conocemos [339] que proporciona guías explícitas para la reutilización de metainformación en forma de atributos y trazas. Estas cuestiones forman parte de una lista de ocho cuestiones clave para la reutilización, que se han determinado como resultado de la experiencia acumulada con este método.

Pensamos que los resultados planteados en este capítulo son de interés para profesionales del desarrollo de software, desarrolladores de herramientas CARE e investigadores: (1) los profesionales del desarrollo de software disponen de un método suficientemente sencillo como para ser adoptado en un corto espacio de tiempo por una

organización inmadura en cuanto a su proceso de requisitos. En caso de que la organización sea madura en cuanto a la gestión de los requisitos y por tanto haya definido explícitamente su método de IR, con SIREN se dispone de un conjunto de tareas, técnicas y guías que se pueden incorporar a dicho método de IR para que soporte la reutilización de requisitos textuales en lenguaje natural, que siguen ocupando un lugar importante en la industria y que facilitan la participación de todos los interesados en el análisis, negociación y validación de los requisitos. SIREN se ha adoptado con éxito por parte de cinco pequeñas y medianas empresas de desarrollo en el proyecto GARTIC, y los analistas involucrados no lo han considerado como una sobrecarga de trabajo importante, al tiempo que consideran muy significativo el valor añadido de reunir en un catálogo de requisitos el conocimiento de la empresa sobre un dominio de aplicación en el que ha realizado desarrollos o tiene previsto iniciarlos; (2) los desarrolladores de herramientas CARE disponen de una serie de ocho cuestiones clave que sus herramientas deben soportar de una forma u otra. Se trata de cuestiones sencillas, que plantean cuestiones que creemos son fácilmente extrapolables al metamodelo de las técnicas soportadas por cualquier herramienta de requisitos, sin crear problemas operativos reseñables; y (3) los investigadores disponen de un método sencillo y probado, basado en reutilización de requisitos, que pueden extender con trabajos en otras dimensiones: DSDM (como se va a desarrollar a lo largo de esta tesis doctoral), desarrollo global de software (como la propuesta realizada por Joaquín Nicolás en su tesis, el lector puede leer la Sección 2.10 para más información [245]), IR basada en metas, trazabilidad y componentes, etc.

Capítulo 5

REMM: un metamodelo para Ingeniería de Requisitos

5.1. Introducción

El principio básico “*todo es un objeto*”, que en los 80 era importante para reforzar la tecnología orientada a objetos, ha evolucionado de forma natural hacia el principio “*todo es un modelo*” que sustenta la tendencia actual del desarrollo de software dirigido por modelos (DSDM, en inglés *MDD, Model Driven Development*), también denominada Ingeniería Dirigida por Modelos (*MDE, Model Driven Engineering*) [33]. El cambio fundamental es que los programadores ya no utilizan los modelos como mera documentación de sus desarrollos, que a menudo contenían información inconsistente, sino que ahora se van a utilizar directamente para dirigir las herramientas de producción de software. Este cambio de paradigma en el desarrollo del software tiene importantes consecuencias en la forma en la que los sistemas de información se construyen y se mantienen [33].

De acuerdo con Hailpern y Tarr [118] el DSDM es un enfoque de desarrollo de software que consiste en la aplicación de modelos y tecnologías de modelos para elevar el nivel de abstracción en el que los desarrolladores crean y hacen evolucionar el software, con el objetivo tanto de simplificar (reduce tanto el esfuerzo del desarrollador como la complejidad de los artefactos software que utiliza) como de formalizar (estandarizando, de manera que es posible la automatización) las distintas actividades y tareas que componen el ciclo de vida del software. Permitiendo a los desarrolladores trabajar a un nivel de abstracción más alto, el DSDM pretende reducir la complejidad y, por tanto, mejorar la calidad del software.

Bran Selic [312] afirma que la principal ventaja de centrarse en modelos en lugar de en los programas es que los modelos se expresan utilizando conceptos que están mucho menos ligados a la tecnología de implementación subyacente y están mucho más cercanos al dominio del problema, en comparación con los lenguajes de programación más populares. Esto hace que los modelos sean más fáciles de especificar, entender y mantener; en algunos casos incluso podría ser posible que el experto del dominio sea el

que produzca el sistema, en lugar del especialista en tecnología de la computación. También hace que los modelos sean menos sensibles a los cambios en la tecnología escogida (el concepto de modelos *independientes de la plataforma* está muy ligado al DSDM).

Las características que a juicio de Selic [312] debe poseer un modelo de ingeniería son:

- *Abstracción*: el modelo es siempre una interpretación reducida del sistema que representa. Eliminando u ocultando los detalles que son irrelevantes desde un punto de vista, nos permite entender los detalles esenciales con mayor facilidad.
- *Entendible*: los conceptos representados se tiene que presentar de forma intuitiva. Que un modelo sea entendible está directamente relacionado con la expresividad de la forma de modelado utilizada.
- *Precisión*: el modelo debe proporcionar una representación realista del sistema que se está modelando.
- *Predicción*: se debe poder utilizar el modelo para predecir correctamente propiedades interesantes (pero no obvias) del sistema que se está modelando mediante experimentación o mediante algún tipo de análisis formal. Claramente esto depende enormemente de la precisión y de la forma de modelado.
- *Barato*: el modelo tiene que ser significativamente más barato de construir y analizar que el sistema modelado.

Selic [312] considera que, probablemente, el motivo por el que las técnicas de modelado del software no han tenido mucho éxito en el pasado se deba a que no cumplían algunos de los criterios que acabamos de listar.

Por su parte, Atkinson y Kühne [25] afirman que la motivación que subyace al DSDM es la mejora de la productividad, esto es, incrementar el rendimiento derivado del esfuerzo dedicado al desarrollo del software de dos formas: 1) mejora de la productividad a corto plazo, incrementando la funcionalidad que se deriva de los artefactos del software, y 2) mejora de la productividad a largo plazo, reduciendo el ratio por el que un artefacto del software se vuelve obsoleto.

Las herramientas conceptuales a las que estaba enfocado el desarrollo de software de los 80, se han renovado de manera que lo que ahora parece ser importante es que una vista particular (o aspecto) de un sistema se puede capturar por un *modelo* y que cada modelo se escribe en el lenguaje de su *metamodelo* (Figura 25) [33]. De acuerdo con Kleppe *et al.* [170] un *modelo* es una descripción de (parte de) un sistema escrito en un lenguaje bien definido. A su vez, un *lenguaje bien definido* es un lenguaje con forma (sintaxis) y significado (semántica) bien definido, que es adecuado para ser interpretado automáticamente por un ordenador. El mecanismo para la definición de estos lenguajes se conoce como *metamodelado* (Figura 26).

Del mismo modo que un modelo define qué elementos pueden existir en un sistema, un lenguaje define los elementos que pueden existir o ser usados en los modelos. Siguiendo esta semejanza, el lenguaje se puede describir mediante un modelo denominado *metamodelo*. Esto es, el modelo del lenguaje que describe los elementos que se pueden utilizar en este lenguaje. Cada tipo de elemento que utiliza un modelador en sus modelos está definido en el metamodelo del lenguaje que utiliza. Dado que un metamodelo también es un modelo, el propio metamodelo debe escribirse en un lenguaje

bien definido. Este lenguaje se denomina *meta-lenguaje* y está definido por un *meta-metamodelo* [170]. El lector puede encontrar una amplia discusión sobre el significado de los conceptos de modelo y metamodelo en el artículo publicado por Seidewitz titulado *What models mean* [310].

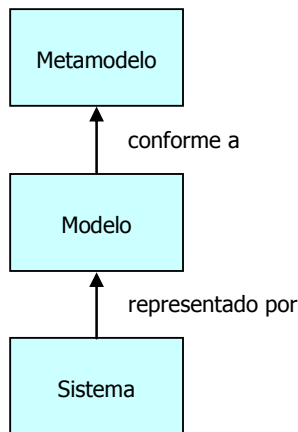


Figura 25 Conceptos básicos en la Ingeniería de Modelos [33].

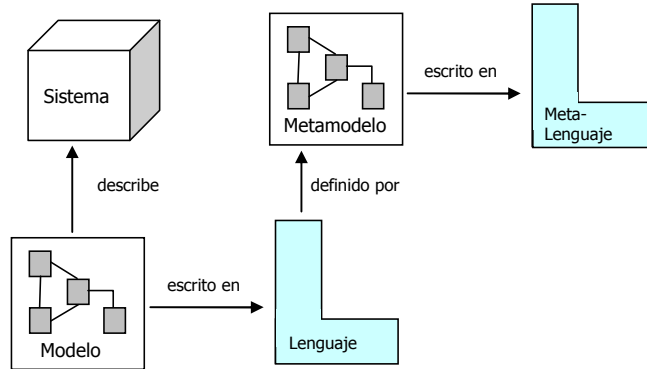


Figura 26 Modelo, Metamodelo y Meta-lenguaje [170].

Este capítulo está dedicado a la definición del lenguaje para el modelado de requisitos definido por el metamodelo REMM (*Requirements Engineering Metamodel*). REMM surge con el propósito de integrar los artefactos implicados en la fase de Ingeniería de Requisitos (IR) en un proceso general de DSDM. En la Sección 1.1.3.2 se presenta brevemente la propuesta MDA y cómo se sitúa REMM en este contexto, pero no vamos a particularizar nuestra propuesta a este enfoque. Parafraseando a Jean-Marie Favre [84] *MDE no es MDA. El estándar MDA del OMG es simplemente una encarnación específica del enfoque MDE*.

En primer lugar presentamos una visión general de los elementos básicos que conforman el lenguaje de modelado de requisitos (Sección 5.2). A continuación, pasamos a detallar los conceptos y relaciones: 1) comunes a las fases *con* y *para* reutilización (Sección 5.3) y 2) los propios de la facetas *para* reutilización y *con* reutilización (Sección 5.4). Después, listamos las restricciones de diseño impuestas para la validación de los modelos de requisitos (Sección 5.6). Por último, presentamos las conclusiones que incluyen nuestras aportaciones al estado del arte en este campo (Sección 5.6).

5.2. Un lenguaje para el modelado de requisitos

Para la integración de los requisitos en un enfoque de DSDM, hemos justificado en la Sección 1.1.3.1, la necesidad de que los requisitos sean modelados de manera que se alineen con el resto de artefactos del desarrollo. De acuerdo con el esquema de la Figura 25, el sistema que queremos modelar en nuestra propuesta es la especificación de requisitos, y el modelo de requisitos que lo represente se definirán conforme a un metamodelo que hemos denominado REMM (*Requirements Engineering MetaModel*) - Figura 27-. Siguiendo el esquema de la Figura 26, es necesario definir un lenguaje a

partir del cual se puedan escribir los modelos de requisitos que incluya tanto la sintaxis abstracta como la sintaxis concreta, junto con las reglas que restringen el uso de los conceptos del lenguaje y su semántica [25]. Estos elementos para el lenguaje de requisitos se resumen en la Tabla 35.

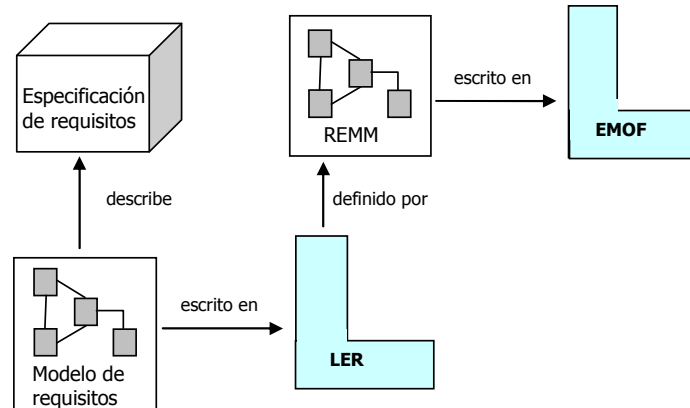


Figura 27 Propuesta de modelado de requisitos.

Concepto	Propósito	Solución propuesta
Sintaxis abstracta	Conceptos a partir de los cuales se crean los modelos	REMM (<i>Requirements Engineering MetaModel</i>)
Sintaxis concreta	Notación utilizada para representar los modelos	Notación REMM
Cualidad de estar bien formado	Reglas que se aplican a los conceptos	Restricciones OCL o Java definidas en REMM
Semántica	Descripción del significado de los modelos	Especificación en lenguaje natural

Tabla 35 Definición de un lenguaje para modelos de requisitos.

A lo largo de este capítulo (Secciones 5.3, 5.4 y 5.5) vamos a presentar REMM, el metamodelo que define el lenguaje (LER, Lenguaje para la Especificación de Requisitos en Figura 27) que va a permitir escribir un modelo de requisitos que represente o describa una especificación de requisitos (Figura 27). Como apuntábamos en la sección 1.1.3.3, este metamodelo especifica formalmente los conceptos y relaciones de traza que a nuestro juicio, basado tanto en el estudio de la bibliografía relacionada (Capítulo 3) como en la experiencia de aplicación del método SIREN (Sección 4.4), deben formar parte del proceso de IR. Para la definición del metamodelo hemos optado por la definición de un DSL (*Domain Specific Language*) utilizando el meta-lenguaje propuesto por el OMG denominado MOF [254] (en realidad un subconjunto llamado EMOF, *Essential MOF*), en lugar de definir un perfil UML [94] (en la Sección 1.1.3.3 discutimos los motivos que nos han llevado a tomar esta decisión). Actualmente, la implementación de un DSL es considerada por algunos autores [163, 354] como la opción más adecuada pues de esta forma el diseño y la implementación no está restringida por las limitaciones del mecanismo de creación de perfiles. La descripción de la sintaxis

concreta del lenguaje, esto es, la notación utilizada para la representación de los modelos, se pospone hasta el Capítulo 7, donde se presentan los editores de modelos en el marco de la herramienta de soporte REMM-Studio[†]. No obstante, en la Figura 35 se muestra un ejemplo de un modelo de requisitos donde se puede apreciar la notación utilizada.

5.3. Conceptos básicos y relaciones de traza en REMM

En esta sección vamos a presentar los conceptos y relaciones de trazas que conforman los elementos (o artefactos) básicos de una especificación de requisitos, tanto en la faceta *con* reutilización como en la faceta *para* reutilización, definidas en SIREN (Sección 4.1). De acuerdo con la taxonomía propuesta por Pinheiro [269], las relaciones de traza se pueden clasificar según los objetos implicados en la traza en: relaciones *extra-requisitos* o *entre-requisitos*. Como explicábamos en la Sección 1.1.3.3, en el primer caso los requisitos se trazan hacia otros artefactos del desarrollo (como casos de prueba, stakeholders, etc.) y en el segundo caso los requisitos se relacionan con otros requisitos. En primer lugar nos centraremos en la descripción de los conceptos básicos que configuran el metamodelo de requisitos (Sección 5.3.1) y, a continuación detallaremos las relaciones de traza tanto extra-requisitos (Sección 5.3.2) como entre-requisitos (Sección 5.3.2). Merece la pena destacar que REMM podría considerarse una formalización de los elementos que constituyen el método SIREN, en el sentido de que se formaliza, mediante la definición de un metamodelo, los elementos y relaciones previamente especificados a partir de nuestra experiencia (ver Sección 4.4). En ocasiones la definición de SIREN ha sido revisada. Estas revisiones serán debidamente comentadas y justificadas en cada caso.

5.3.1. Conceptos básicos en REMM

En esta sección se presentan los conceptos básicos (metaclases) que conforman el metamodelo de requisitos (*Catálogo*, *Requisito*, *Stakeholder*, *Objeto Externo*, *Caso de Prueba* y *Glosario de Términos*). Todos estos conceptos se muestran en la Figura 32, donde también se detallan las relaciones existentes entre ellos.

5.3.1.1. Catálogo

En SIREN, el repositorio de requisitos reutilizables se encuentra organizado por catálogos, que se corresponden con un conjunto de requisitos reutilizables (Sección 4.1). Si bien en SIREN se hace referencia a los catálogos de requisitos como contenedores de requisitos reutilizables, y por tanto, estos catálogos son el resultado de la faceta *para* reutilización, en REMM hemos generalizado esta idea de catálogo para cubrir tanto la faceta *para* como la faceta *con* reutilización, en el sentido de que el producto resultado de ambas facetas serán sendos catálogos de requisitos. En el caso de la faceta *para* reutilización se especificará un catálogo reutilizable, mientras que como resultado de la faceta *con* reutilización se especificará un catálogo de requisitos de producto (la diferencia entre ambos tipos de catálogos se detalla en la Sección 5.4. Así, de manera

general, en REMM los requisitos se almacenan en *catálogos* (*Catalog* en Figura 32) que se caracterizan por un *nombre* y un *propósito*. En REMM se han definido dos tipos de catálogos, correspondientes a cada una de las facetas de la reutilización, cuya implementación detallaremos en la Sección 5.4.

5.3.1.2. Requisito

De acuerdo con la definición de SIREN, los *requisitos* (*Requirement* en Figura 32) se caracterizan por una serie de atributos (Sección 4.3.2) que conforman el conjunto mínimo (MIN). Este conjunto mínimo ha sido especificado en REMM: 1) mediante atributos que caracterizan a la metaclase que representa a los requisitos, y 2) mediante relaciones de traza con otros conceptos (relaciones de traza extra-requisitos, Sección 5.3.2), para poder añadir más información. Los atributos procedentes de MIN que caracterizan a un requisito en REMM son: el *identificador* y el *texto*, que son obligatorios, la justificación para su inclusión (*rationale*), *prioridad*, *estado* y *riesgo*. Los valores posibles para estos tres últimos atributos vienen definidos por distintos tipos de datos enumerados (*PriorityType*, *StatusType* y *RiskType*, respectivamente) que incluye los valores establecidos en las guías de SIREN (Sección 4.3.2).

En REMM aparecen nuevos atributos para los requisitos respecto a los definidos en SIREN: *nombre*, *tipo* y *nivel de abstracción*. Por un lado, el *nombre*, que representa una descripción simplificada del requisito y será útil para favorecer la legibilidad de la representación gráfica de los modelos de requisitos. Por otro lado, los atributos *tipo* y *nivelAbstracción* nos van a servir para caracterizar los tipos de requisitos propuestos en SIREN (Sección 4.3.1). En REMM consideramos que los requisitos se pueden definir a distintos niveles de abstracción (*AbstractionLevelType*), que son: requisitos del negocio (*BUSINESS_REQUIREMENTS*), requisitos del sistema (*SYSTEM*), características (*FEATURE*), requisitos del software (*SOFTWARE*) y reglas del negocio (*BUSINESS_RULES*). Los requisitos del negocio representan los objetivos que el negocio quiere conseguir o las necesidades de los stakeholders, los requisitos del sistema describen requisitos de alto nivel de un sistema, mientras que los requisitos del software están relacionados con las funcionalidades o capacidades que tiene que tener el producto. Por su parte, las reglas de negocio guían el comportamiento del sistema en función de la legislación o normas de la organización. Las reglas de negocio establecen si algo puede o no hacerse, o bien las condiciones o criterios que tienen que cumplirse. Por tanto, los requisitos se tienen que implementar para hacer cumplir las reglas de negocio. Por ejemplo, un requisito software podría ser “SR1: El sistema permitirá la retirada de dinero de la cuenta corriente”, este requisito debe cumplir con las reglas del negocio (BR):

BR1: El límite de dinero a retirar de la cuenta es de 1.000 euros al día.

BR2: No se puede retirar efectivo más de tres veces al día.

BR3: La cuenta no puede quedar con saldo negativo.

Generalmente los requisitos del negocio se centran en el “qué” se quiere alcanzar y, por tanto, se mueven en el espacio del problema, por ejemplo, “Ser la compañía nº 1 de consultas de ofertas de empleo”. En cambio, los requisitos del sistema representan objetivos más técnicos, y tiene que ver con la solución al problema que se plantea en los

requisitos del negocio, especifican el “cómo”, por ejemplo, “Implementar un portal web de ofertas de empleo”.

Las características (*FEATURE*), de acuerdo con Leffingwell [186] (Figura 28), se derivan de las necesidades de los stakeholders (requisitos del negocio), y representan los servicios que el sistema debe proporcionar para cumplir una o más de estas necesidades. Las características son expresiones de alto nivel que se concretarán mediante la especificación de requisitos software. Por ejemplo, una necesidad podría ser “mantener al cliente informado del estado de su solicitud” y de esta necesidad se deriva la características “capacidad de notificación por correo electrónico”.

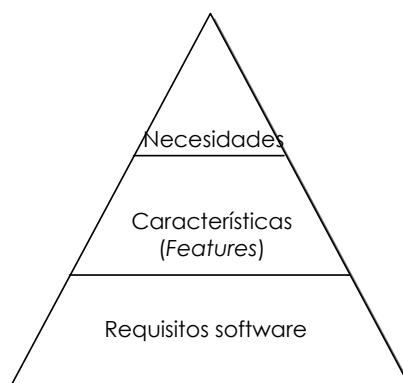


Figura 28 Relación entre necesidades, características y requisitos software según Leffingwell [186].

De acuerdo con el SWEBOK (*Software Engineering Body of Knowledge*) [133], los requisitos del sistema (*SYSTEM*) especifican los requisitos del sistema como un todo. La palabra “sistema” se refiere a una combinación de elementos que interaccionan para conseguir un objetivo. Estos elementos pueden ser hardware, software, *firmware*, personas, información, técnicas, facilidades, servicios y otros elementos de soporte, según se recoge en la guía del INCOSE [135]. En un sistema que contiene componentes software, los requisitos del software se derivan a partir de los requisitos del sistema. Para productos software sencillos, sólo es necesaria la especificación de los requisitos software y por tanto éstos se derivarán directamente de las necesidades del negocio y características del producto.

Gráficamente, la relación entre los distintos niveles de abstracción definidos en REMM se muestra en la Figura 29. Los requisitos de más bajo nivel se derivan de necesidades u objetivos propuestos a un nivel más alto, o dicho de otro modo, los requisitos software se definen para satisfacer o cumplir los requisitos del sistema, o del negocio, propuestos. Hemos omitido en REMM el tipo de requisito *caso de uso* definido en SIREN. Dado que, de acuerdo a la definición que hemos dado en la Sección 4.3.1, un caso de uso engloba generalmente distintos requisitos funcionales, la especificación de los casos de uso la hemos considerado como requisitos funcionales en REMM de acuerdo al nivel de abstracción que corresponda (negocio, sistema o software).

En cuanto a los tipos de requisitos (*RequirementType*), en la literatura es común la clasificación de los requisitos en: *funcionales*, *no funcionales* (o *requisitos de calidad*) y *restricciones* [275]. Los dos primeros tipos están contemplados en SIREN (Sección 4.3.1)

mientras que la inclusión de las restricciones (*CONSTRAINT*) en REMM es fruto de una revisión posterior. De acuerdo con Robertson y Robertson [293], una restricción es un requisito global que restringe el diseño final del producto. Se considera un tipo más de requisito dado que cualquier solución que no cumpla con estas restricciones no es aceptable.

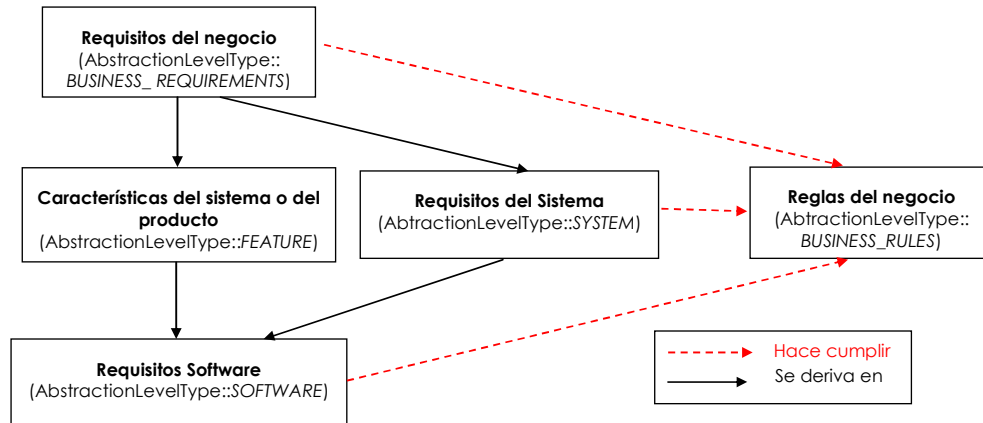


Figura 29 Relación entre los niveles de abstracción de los requisitos en REMM.

Existen muchos tipos diferentes de requisitos de calidad, y se han propuestos distintas taxonomías [173, 293, 360]. Para definir los tipos de requisitos no funcionales en REMM (descritos en el enumerado *RequirementType*) hemos tomado como referencia el modelo de calidad propuesto por el estándar ISO/IEC 9126 [138] pues hemos considerado que la ingeniería de requisitos le resultaría útil la especificación de requisitos que redunden en una mejora de la calidad del sistema software final. El estándar define seis categorías para clasificar la calidad interna y externa del software (Figura 30) y cada una de ellas se divide en diferentes subcategorías. Algunas de las subcategorías no han sido incluidas en REMM (*Suitability*, *Analyzability*, *Stability* and *Testability*) porque hemos considerado que no se corresponden con tipos de requisitos, estrictamente hablando, sino más bien, principios generales que deben guiar el proceso de desarrollo y especificación de los requisitos. Por ejemplo, es frecuente la especificación de un requisito en relación con la disponibilidad (*availability*) del sistema, pero parece que no tiene mucho sentido especificar cuándo un requisito debe ser adecuado (*suitable*) o estable (*stable*). Todos los sistemas se deben diseñar de manera que cumplan con estas propiedades.

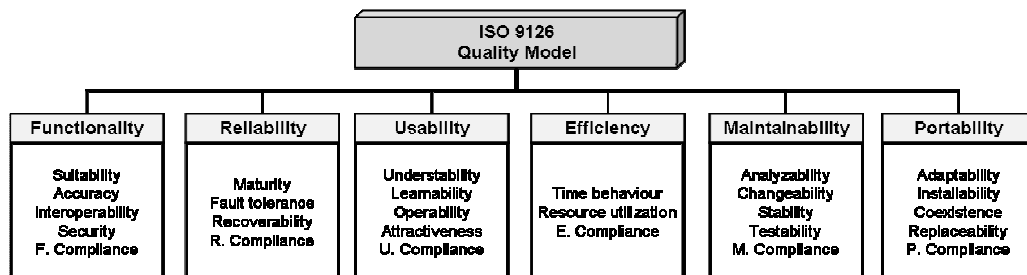


Figura 30 Modelo de calidad del estándar ISO 9126.

El estándar ISO 9126 ha sido revisado y reemplazado por el modelo SQuaRE (*Software Product Quality Requirements and Evaluation*) [141]. El modelo de calidad ha ido evolucionando hasta la versión aprobada recientemente. En todas las versiones la estructura del modelo de calidad en categorías y subcategorías es la misma y únicamente han cambiado ligeramente la taxonomía. Estos cambios implicarían únicamente añadir al enumerado *RequirementType* en REMM las categorías y subcategorías nuevas.

Cabe destacar que, como se puede ver en la Figura 32, en la versión actual de REMM el número de atributos que caracteriza un requisito es fijo. No obstante, podrían surgir nuevas necesidades de recogida de meta-información para un tipo de requisito, que impliquen la definición de nuevos atributos. Las herramientas comerciales como Requisite Pro, soportan la definición de nuevos atributos asociados a tipos de requisitos. Aunque la versión actual de REMM no tiene esta capacidad, su inclusión supondría pocos cambios. En la Figura 31 se muestra las modificaciones que habría que hacer en REMM para que permitiera la definición, por parte del usuario de la herramienta que lo soportase, de nuevos atributos para un requisito. En la Figura 31 se puede ver como el catálogo de requisitos contendría, además del conjunto de requisitos (*Requirement*), las definiciones de los requisitos (*RequirementDefinition*) que conlleva la especificación de nuevos atributos (*Attribute*). Al crear un nuevo requisito se podría establecer que este requisito debe ser conforme a una definición de requisito, en cuyo caso tendría que instanciar los atributos (*AttributeInstance*) asociados a dicha definición. Cada atributo se caracteriza por un *nombre*, un *tipo* y la obligatoriedad o no de que lo especifiquen los requisitos que sean conformes a esta definición (*required*). Por ejemplo, en la Sección 4.4.1.4, se presentaban los atributos específicos para recoger información propia de los requisitos de seguridad: *fuerce* (posición en la jerarquía de activos de MAGERIT [202]) y *criticidad* (grado de obligación de incluir el requisito en el proyecto). Para no confundir este atributo *fuerce* con la traza del requisito a la documentación de la que procede, vamos a renombrarlo como *fuerce_MAGERIT*. Para que todos los requisitos de seguridad incluyan estos dos nuevos atributos tendríamos que especificar una definición de requisito de nombre *Seguridad* que contuviera dos nuevos atributos: *fuerce_MAGERIT* (de tipo cadena de texto) y *criticidad* (de tipo enumerado con tres posibles valores *obligatorio*, *recomendable* y *opcional*). De esta forma, si un requisito establece que su definición debe ser conforme a la de un requisito de seguridad, tendrá que instanciar los atributos *fuerce_MAGERIT* y *criticidad*, con un valor consistente con el tipo con el que se han declarado.

5.3.13. Stakeholder

Los stakeholders son las personas interesadas o implicadas en la especificación de requisitos. En REMM se recoge la información del *Stakeholder* que propone o que participa en la definición de un requisito. Un stakeholder se caracteriza por un *nombre* (que lo identifica de manera única) y el *puesto* que desempeña en la empresa. De acuerdo con las buenas prácticas propuestas por Sommerville y Sawyer [329] (*Guía 4.4 Registrar fuentes de los requisitos*) es importante registrar el puesto junto con el nombre de la persona porque, de esta forma, si es necesario revisar un requisito y la persona que propuso el requisito no se encuentra en la empresa, se puede consultar a otra persona

que ocupe el mismo cargo, dado que el conocimiento sobre el dominio será similar. De acuerdo con Robertson y Robertson [293], la persona que propone un requisito debe tener el conocimiento y la autoridad adecuada para el tipo del requisito que propone.

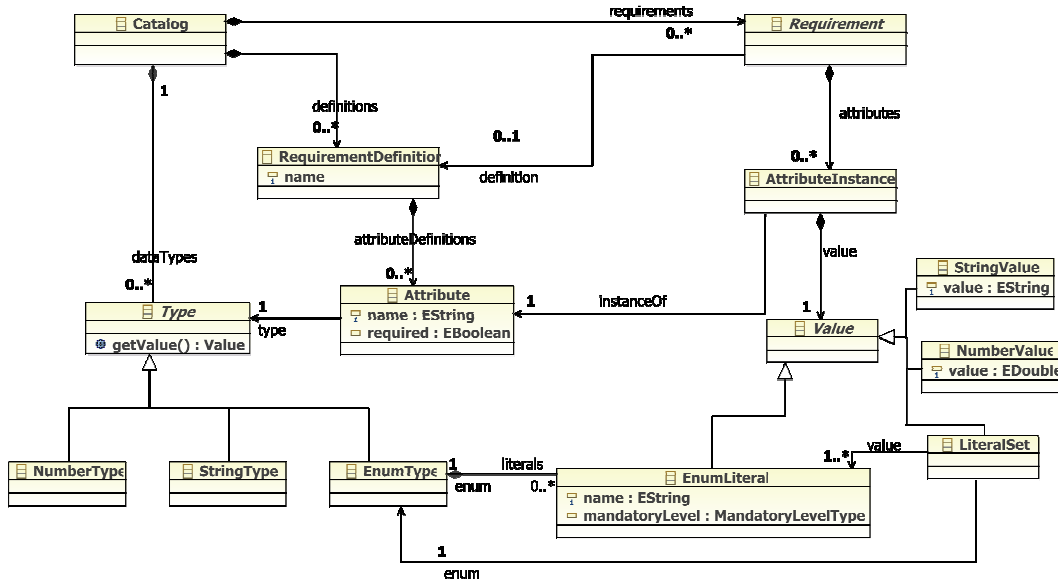


Figura 31 Definición de nuevos atributos en REMM.

5.3.1.4. Objeto Externo

Un objeto externo (*ExternalObject*) en REMM representa cualquier documentación externa a la especificación de requisitos (legislación, normativa de la organización, etc.) a partir de la cual se haya obtenido el requisito especificado o que complemente la especificación de requisitos. Estos objetos externos se caracterizan por el *nombre*, por ejemplo, Reglamento de Medidas de Seguridad, y la *localización*, esto es, la ubicación de esta documentación en Internet (URL para la descarga) o en los archivos físicos o electrónicos de la organización. Por tanto, la doble función de los objetos externos en REMM es:

1. El objeto externo como fuente de un requisito. De acuerdo con las buenas prácticas propuestas por Sommerville y Sawyer [329] (*Guía 4.4 Registrar fuentes de los requisitos*) la fuente de un requisito puede ser cualquier stakeholder, o grupo de ellos, que han propuesto el requisito, o también, estándares de calidad de la organización, documentación técnica o informes de incidencias. Cuando se documenta un requisito se deben registrar la fuente o fuentes en las que se basa la definición de un requisito, de esta forma, 1) si un requisito cambia se sabe inmediatamente qué persona o documentación hay que consultar, y 2) se sabe por qué existe un requisito.
2. El objeto externo como información que complementa la especificación de requisitos. Siguiendo las recomendaciones de Robertson y Robertson [293], no debemos pretender escribir todos los detalles en la especificación de requisitos sino que se debe referenciar cualquier material que sea importante para el requisito. De hecho, el IEEE-830 [131] propone destinar una sección del

documento de especificación de requisitos (*1.4 References*) al listado completo de todos los documentos referenciados desde los requisitos. Esta recomendación está en la línea de las buenas prácticas propuestas por Sommerville y Sawyer [329], quienes aconsejan complementar la descripción del requisito en lenguaje natural con descripciones más precisas utilizando una notación adecuada. De esta forma, si la notación utilizada es conocida por los expertos en el dominio de la aplicación, es menos probable que se cometan errores en la especificación de requisitos y es más probable que se detecten errores que se hayan podido cometer.

5.3.15. Casos de prueba

REMM incluye la definición de *casos de prueba* (*TestCase*) como un concepto propio en el metamodelo. Se ha reconocido que la definición de posibles casos de prueba, para comprobar si el sistema satisface el requisito, es una forma efectiva de detectar problemas en cuanto a la completitud o ambigüedad de la definición de los requisitos. Si nos resulta difícil derivar casos de prueba a partir de un requisito, posiblemente se deba a que falta información (el requisito es incompleto) o que la descripción del requisito no establece claramente lo que se requiere (el requisito es ambiguo) (*Guía 8.7 Proponer casos de prueba para los requisitos* [329]). Robertson y Robertson [293] también proponen la definición de pruebas de aceptación (*fit criterion*) escritos de forma precisa y cuantificable de manera que las soluciones pueden validarse contra los requisitos para determinar si la implementación cumple con los requisitos. En REMM un caso de prueba se caracteriza por: un identificador (*id*), que lo identifica de manera única, la descripción del objetivo de la prueba, las posibles entradas y los resultados esperados.

5.3.16. Glosario de términos

Los términos especializados que se utilizan en la especificación de requisitos deben definirse en un glosario [329] (*Guía 3.5 Definir Términos Especializados*) dado que los problemas con la terminología es quizás la fuente de confusión más común. Diferentes lectores con diferente formación es probable que entiendan el mismo término de forma diferente y por tanto, que lo utilicen de diferente forma a la hora de especificar los requisitos. El uso de un glosario puede evitar estos malentendidos entre lectores y escritores del documento de requisitos. La necesidad de proporcionar un glosario también se pone de manifiesto en la mayoría de guías y estándares de especificación de requisitos, en particular, las recomendaciones del estándar IEEE-830 [131] propone definir todos los términos, acrónimos y abreviaturas necesarias para interpretar el documento de especificación de requisitos (*Sección 1.3 Definitions, acronyms and abbreviations*). Robertson y Robertson [293] también destacan que siempre se sorprenden de cómo ocurren muchos malentendidos simplemente porque no está disponible un glosario centralizado. Según estos mismos autores, merece la pena dedicar esfuerzos a este aspecto para asegurar una buena comunicación posteriormente en el proyecto. En REMM se incluye la metaclass que representa un glosario de términos (*Glossary*). Los términos que se registran en el glosario (*Term*) se caracterizan por un

nombre, la descripción o definición de dicho término y un conjunto de sinónimos (si los tuviera).

5.3.2. Trazabilidad extra-requisitos en REMM

Las relaciones de traza extra-requisitos (Figura 32) son aquellas relaciones de traza que ligán los requisitos con otros artefactos del desarrollo, bien relacionados con la fase de IR (conceptos presentados en la sección 5.3.2) o con otras etapas del desarrollo (como análisis o diseño), que de acuerdo a la taxonomía propuesta por Pinheiro [269] (y presentada en la Sección 1.1.3.3) se denominan *trazas hacia delante*.

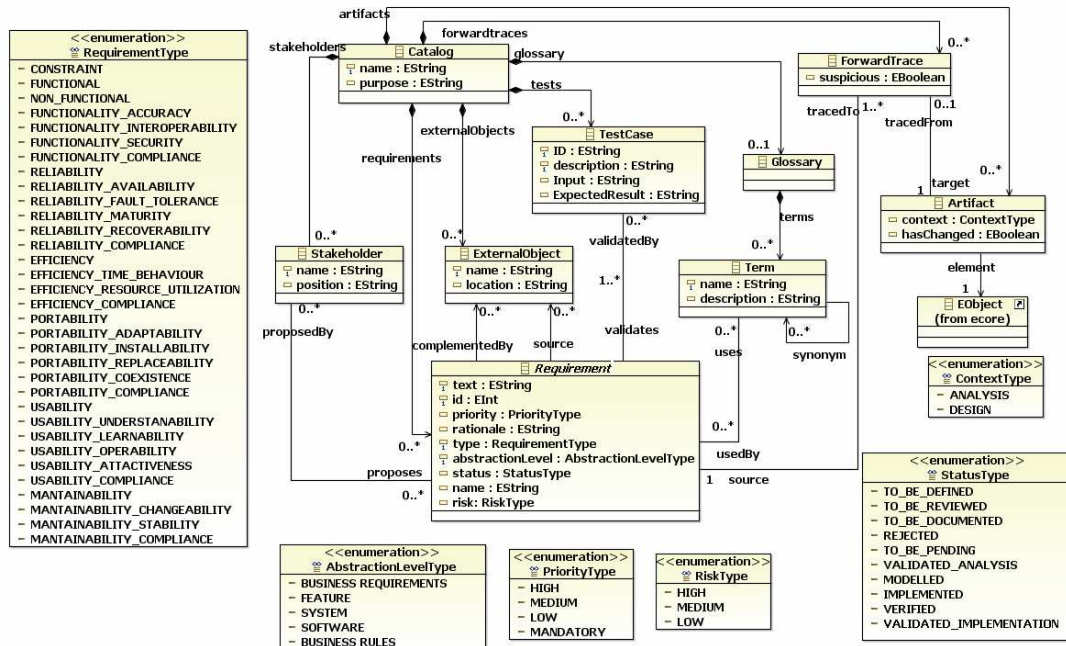


Figura 32 Trazabilidad extra-requisitos en REMM.

Por un lado, las relaciones de traza definidas en REMM que relacionan los requisitos con otros artefactos de la etapa de IR son:

- *proposedBy* relaciona el *requisito* con el/los *stakeholders* que lo propone. Esta relación de traza se corresponde con el atributo *propuestoPor* del conjunto MIN de SIREN (Sección 4.3.2). Esta relación de traza resulta de utilidad en caso de conflicto entre requisitos o si hay que volver sobre la especificación de requisitos para introducir algún cambio, para poder localizar fácilmente a la fuente del requisito.
- *source* relaciona el *requisito* con el/los *objetos externos* (*ExternalObject*) donde se registra la información sobre la documentación de la que se ha extraído el requisito (legislación, normativas de la empresa, etc.). Esta relación de traza se corresponde con el atributo *fuentes* del conjunto MIN de SIREN (Sección 4.3.2). Al igual que en el caso anterior, si la especificación de un requisito tiene que revisarse, resulta de utilidad acudir a su fuente.

- *validatedBy*: relaciona el *requisito* con el/los *casos de prueba* (*TestCase*) que describen cómo comprobar si ese requisito se cumple en el sistema. Esta relación de traza se corresponde con el atributo **criterios Validación** del conjunto MIN de SIREN (Sección 4.3.2).
- *complementedBy*: relaciona el *requisito* con el/los *objetos externos* (*ExternalObject*) que especifican el nombre y la localización del material que complementa la especificación del requisito, evitando de este modo incluir todos los detalles en la redacción de los requisitos y simplificando la especificación de los mismos.
- *uses*: relaciona el *requisito* con el/los *términos* especializados (*Terms*) definidos en el *glosario*. De esta forma, se puede comprobar si todos los requisitos han utilizado el mismo término de forma consistente.

Por otro lado, los requisitos se pueden trazar hacia otros artefactos del desarrollo creados en fases posteriores (análisis o diseño, por ejemplo) mediante las relaciones de traza que hemos denominado *ForwardTrace* en REMM. Estas trazas se han definido en REMM de forma genérica, relacionando un requisito con un elemento genérico (*EObject*). De esta forma, es posible enlazar un requisito con cualquier otro elemento de un modelo predefinido, como por ejemplo un concepto de un DSL o cualquier artefacto UML (elemento de un caso de uso, de un diagrama de clases, etc.). Si el objeto enlazado con el requisito cambia, la traza hacia delante se marcará como *sospechosa* (*suspicious*) para que se verifique si el requisito sigue siendo consistente con el modelo, en función del cambio realizado. Dado que cada requisito mantiene una traza hacia sus fuentes (el stakeholder que lo propuso o la documentación de donde se extrajo) se pueden consultar para validar el cambio. Esta relación de traza hacia delante en REMM (*ForwardTrace*) se corresponde con la relación *materializaEn* propuesta en SIREN (Sección 4.3.4.5).

5.3.3. Trazabilidad inter-requisitos en REMM

La mayoría de los requisitos que desarrollan durante el proceso de IR no se pueden tratar de forma aislada durante el proceso de desarrollo, sino que están relacionados y se ven afectados unos con otros. Un estudio ha demostrado que únicamente alrededor de la quinta parte de los requisitos especificados en un conjunto de requisitos son realmente independientes [48]. Las dependencias entre los requisitos pueden afectar a diferentes decisiones o actividades durante el desarrollo [67], por ejemplo, gestión de cambios de los requisitos, planificación de versiones, reutilización de requisitos o implementación de los requisitos. En REMM incluimos un modelo de trazabilidad (Figura 33) que pretendemos que sirva de soporte a las etapas de IR y soporte la reutilización sistemática de requisitos. Para ello, partimos de las relaciones de traza propuestas en SIREN (Sección 4.3.4), que han sido revisadas teniendo en cuenta el trabajo de Dahlstedt y Persson [67], que han desarrollado un modelo de las relaciones de interdependencias entre requisitos fundamentales, y las recomendaciones de Alan Davis [71], para no descartar aquellas relaciones importantes en la etapa de negociación de requisitos.

En REMM, las relaciones entre requisitos (*InterReqTrace*) enlazan un requisito fuente (*source*) con un requisito destino (*target*) pudiéndose establecer el motivo por el que esta traza existe (*rationale*) y las condiciones que se tienen que cumplir para que la

traza exista (*condition*). Por un lado, el registro de la justificación de la existencia de la traza (*rationale*) es importante para la toma de decisiones, para apoyar la inclusión o no de los requisitos relacionados con uno dado tanto en la planificación de versiones, negociación de requisitos o en el momento de la reutilización. Por otro lado, la *condición* es realmente interesante en el contexto de los requisitos parametrizados, puesto que se pueden establecer las condiciones que se tienen que cumplir, entre los valores de los parámetros de los requisitos implicados en la relación de traza, para que la relación de traza exista. La explicación de los requisitos parametrizados se presenta en la Sección 5.4. La gramática del lenguaje definido para las condiciones se puede encontrar en la Sección 5.5.1.

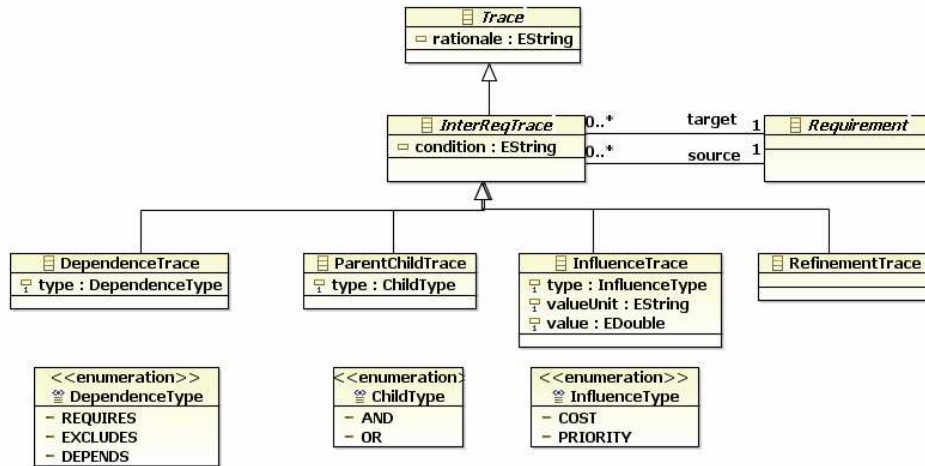


Figura 33 Relaciones de traza inter-requisitos en REMM.

Todas las relaciones de traza, salvo la traza de refinamiento (*RefinementTrace*), se establecen entre requisitos del mismo nivel de abstracción con los siguientes significados:

- Dados dos requisitos R1 y R2, R1 *DependenceTrace::REQUIRES* R2 significa que R2 se necesita para cumplir el requisito R1, o lo que es lo mismo, R2 es una precondition para el cumplimiento de R1 (Sección 4.3.4.2).
- Dados dos requisitos R1 y R2, R1 *DependenceTrace::EXCLUDES* R2 significa que R1 y R2 son requisitos alternativos y no pueden estar los dos al mismo tiempo en la especificación de requisitos (Sección 4.3.4.4).
- Dados dos requisitos R1 y R2, R1 *DependenceTrace::DEPENDS* R2 significa que existe algún tipo de relación entre R1 y R2 o que R1 influye en R2 de algún modo diferente al resto de relaciones (Sección 4.3.4.3).
- Dado un requisito R1 y dos requisitos R1.1 y R1.2, donde R1.1 son los requisitos hijos y R1 es el requisito padre, la relación padre/hijo (*ParentChildTrace*) significa que R1.1 y R1.2 refinan o extienden el significado de R1. Además, en REMM hemos ampliado la definición de este tipo de traza respecto a la definición original de SIREN (Sección 4.3.4.1) permitiendo la creación de dos tipos de trazas padre/hijo diferentes, que serán útiles para modelar puntos de variación en la especificación de requisitos:

- o R1 *ParentChildTrace::AND* R1.1, significa que para cumplir R1, R1.1 tiene también que cumplirse. R1.1 refina la especificación de R1. En tiempo de reutilización, por ejemplo, esta relación implica que si R1 se selecciona será obligatorio seleccionar todos los hijos relaciones mediante una traza AND.
 - o R1 *ParentChildTrace::OR* R1.2, significa que para cumplir R1, R1.2 podría cumplirse pero no es obligatorio, por tanto, R1.2 proporciona un modo alternativo (no exclusivo) para cumplir R1 además de R1.1.
- La relación de traza que representa la influencia de un requisito [48, 66, 67, 71] en otro (*InfluenceTrace*) constituye una novedad respecto al modelo de traza propuesto en SIREN (Sección 4.3.4). De acuerdo con el estudio realizado por Carlshamre *et al.* [48], este tipo de traza era la más común (del 59% al 90% de todas las relaciones identificadas) en tres de los cinco casos de estudios abordados, en concreto, en los casos de estudio etiquetados como de desarrollo de producto. Esta etiqueta se refiere a que existe un producto maduro en el mercado y las situaciones de desarrollo se pueden describir como incrementales y dirigidas por el mercado. Dados dos requisitos R1 y R2:
 - o R1 *InfluenceTrace::COST* R2, significa que la inclusión de R1 en la especificación de requisitos influye en el coste de R2. Esto es, que el cumplimiento de R1 será más fácil (o más difícil) si se cumple el requisito R2. Por ejemplo, la inclusión de R1 puede suponer que el coste de R2 se reduzca en 2 (*value*) persona-mes (*valueUnit*).
 - o R1 *InfluenceTrace::PRIORITY* R2, significa que la inclusión de R1 en la especificación de requisitos influye en el valor que el cliente le da a R2. Bajo nuestro punto de vista, el valor o la importancia que tiene un requisito queda reflejado en la prioridad que se le da a este requisito.

La traza de refinamiento (*RefinementTrace*) se utiliza para enlazar requisitos de distinto nivel de abstracción (Figura 29) dado que los requisitos de más bajo nivel (*requisitos software*) se derivarán de requisitos del sistema o de necesidades u objetivos del negocio (*business requirements*). Por otro lado, la descripción de una característica (*feature*) implicará la especificación de un conjunto de requisitos. Por último, los requisitos (negocio, sistema o software) deben referenciar a las reglas de negocio (*business rules*) que restringen el modo en el que dicho requisito debe abordarse. Los requisitos deben implementarse conforme a las reglas de negocio. Quizá para este caso el nombre de la traza debería haber sido *conforms* pero la hemos incluido bajo el paraguas de la traza de refinamiento para evitar sobrecargar el modelo de trazas de REMM.

Existen pocos trabajos que hayan intentado formalizar las relaciones de traza entre requisitos. Una excepción interesante es la propuesta de Goknil *et al.* [106], donde se formalizan las relaciones con el objeto de definir reglas sobre el impacto de los cambios en los requisitos. El modelo de trazabilidad formalizado incluye las relaciones: *requiere*, *refina*, *contiene*, *estar en conflicto* (*conflicts*). La especificación formal de la semántica de las relaciones de traza queda fuera de los objetivos de esta tesis doctoral.

5.4. Metamodelado de la variabilidad para dar soporte a la reutilización de requisitos

Es reconocida la importancia del modelado de la variabilidad en cualquier estrategia de reutilización [190]. Por tanto, para que REMM soporte la reutilización de requisitos, debe incluir la gestión de la variabilidad de algún modo. Tal y como apuntábamos en la Sección 1.1.2.1, entre las diferentes alternativas para abordar el modelado de la variabilidad, hemos optado por integrar la variabilidad de requisitos directamente en el metamodelo. De manera que REMM soporta ambas facetas de la reutilización: 1) modelado de catálogos de requisitos reutilizables (REMM *para* reutilización), correspondiente a la etapa de ingeniería del dominio en el desarrollo de líneas de productos software (SPL, *Software Product Line*), y 2) modelado de los requisitos de un producto concreto a partir de los catálogos de requisitos reutilizables (REMM *con* reutilización), correspondiente a la etapa de ingeniería del producto en el desarrollo de una SPL.

El concepto central de REMM para dar soporte al modelado *para* reutilización, es el repositorio (*Repository* en Figura 34) que contiene un conjunto de catálogos de requisitos reutilizables (*ReusableCatalog*). Cada catálogo, a su vez, almacena requisitos reutilizables (*ReusableRequirement*) relacionados, que pertenecen al mismo tipo (*CatalogType*). De acuerdo con la definición de SIREN (Sección 4.1), se pueden definir dos tipos de catálogos que se corresponden con la especificación de requisitos de *dominios verticales* (*CatalogType::DOMAIN*), como contabilidad o finanzas, o a la especificación de requisitos de dominios horizontales (*CatalogType::PROFILE*), como seguridad o protección de datos.

Por otro lado, el elemento clave para dar soporte al modelado *con* reutilización, es lo que hemos denominado catálogo de producto (*ProductCatalog* en Figura 34) que contiene los requisitos relacionados con la especificación de un nuevo producto (*ProductRequirement*). Un requisito de producto puede ser: 1) específico del producto actual que se está especificando o, 2) puede haber sido reutilizado a partir de alguno de los catálogos de requisitos reutilizables disponibles en el repositorio. En este último caso, el requisito de producto mantiene una traza al requisito reutilizable del que procede (traza *reusedFrom*) de manera que se pueda consultar el requisito original en cualquier momento.

Los puntos de variación en REMM, necesarios en el modelado *para* reutilización, vienen dados por:

- 1) *requisitos parametrizados*, como vamos a ver a continuación, contiene partes con diferentes alternativas entre las que se tienen que elegir para la especificación de un producto concreto.
- 2) *trazas exclusivas* (*DependenceTrace::EXCLUDES*), sólo uno de los dos requisitos relacionados mediante la traza puede existir en la especificación de requisitos de un producto.
- 3) *trazas padre-hijo OR* (*ParentChildTrace::OR*), especifica alternativas no obligatorias para la implementación del requisito padre en un producto.

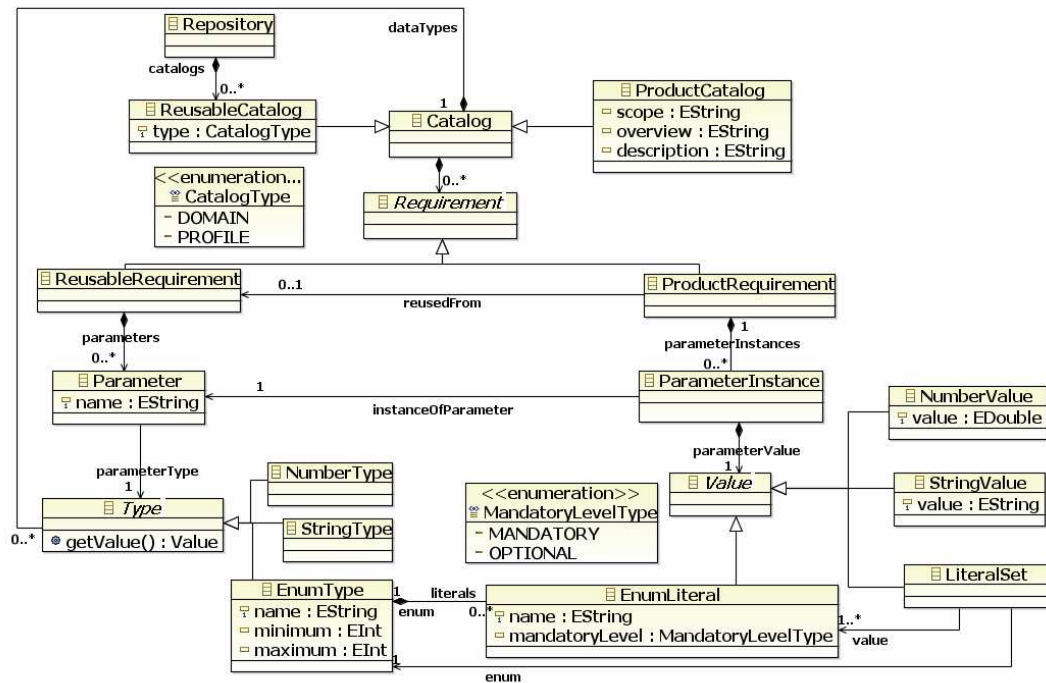


Figura 34 Requisitos reutilizables y requisitos de producto en REMM.

Un *requisito parametrizado* es un requisito reutilizable que contiene uno o más parámetros. Un parámetro (*Parameter*) se caracteriza por un *nombre* y un *tipo*. Los tipos posibles de los parámetros forman parte de la definición del catálogo de requisitos, de esta manera, todos los requisitos definidos en el mismo catálogo pueden utilizar el mismo conjunto de tipos de datos. Por defecto, en el catálogo están disponibles los tipos: numérico (*NumberType*) y las cadenas de texto (*StringType*). El ingeniero de requisitos puede definir en el catálogo tantos tipos enumerados (*EnumType*) como sea necesario. Un tipo enumerado representa un conjunto finito de valores.

Los requisitos parametrizados permiten especificar puntos de variación que pueden variar desde un conjunto infinito de valores (números y cadenas de texto) hasta un conjunto finito de valores (enumerado). En el caso de que el tipo del parámetro sea un enumerado, se tiene que establecer, además del *nombre* del tipo, el *mínimo* y el *máximo* de variantes (*EnumLiteral*) que se pueden seleccionar. Por defecto se inicializa tanto el mínimo como el máximo a uno, lo que significa que se tiene que seleccionar sólo uno de los posibles valores del enumerado en el momento de la instanciación (se explica a continuación). Además, cada una de las variantes (valores del enumerado, *EnumLiteral*) se pueden definir como opcionales (*MandatoryLevelType::OPTIONAL*) o bien como obligatorias (*MandatoryLevelType::MANDATORY*) para indicar si esa variante debe ser incluida o no como parte de cualquier producto que se defina a partir del catálogo de requisitos reutilizables. En la Figura 35 se muestra un ejemplo de requisitos parametrizados que están relacionados mediante trazas padre-hijo, extraídos del caso de estudio que se presentará en Capítulo 8. El modelo representa la especificación de requisitos de un sistema de alarma que puede ser de intrusión, humo, fuego o agua. Para

ello se define el requisito parametrizado Req_5 que incluye el parámetro de nombre *TYPE* y tipo *KindOfAlarm*, que es un enumerado. Este enumerado se ha definido con valores mínimo y máximo igual a uno (*/1,1/*), lo que significa que sólo se puede seleccionar uno de los tipos de alarmas. En el caso de que se haya especificado la alarma contra inundaciones (condición IF (*TYPE = WATER*)) conforme a la gramática que se presentan en la sección 5.5.1) es opcional (*CHILD::OR*) la instalación de un válvula para cortar el suministro de agua. Para cualquier tipo de alarma que se instale es opcional (*CHILD::OR*) el envío de un mensaje al móvil del cliente para avisar de que la alarma ha saltado. Pero si se decide implementar esta opción, es obligatorio (*CHILD::AND*) que se proporcione el número del móvil al que se va a mandar el mensaje (Req_21) y el mensaje que se va a enviar (Req_22). Tanto el Req_21 como el Req_22 se han definido como requisitos parametrizados con un parámetro de tipo cadena de texto.

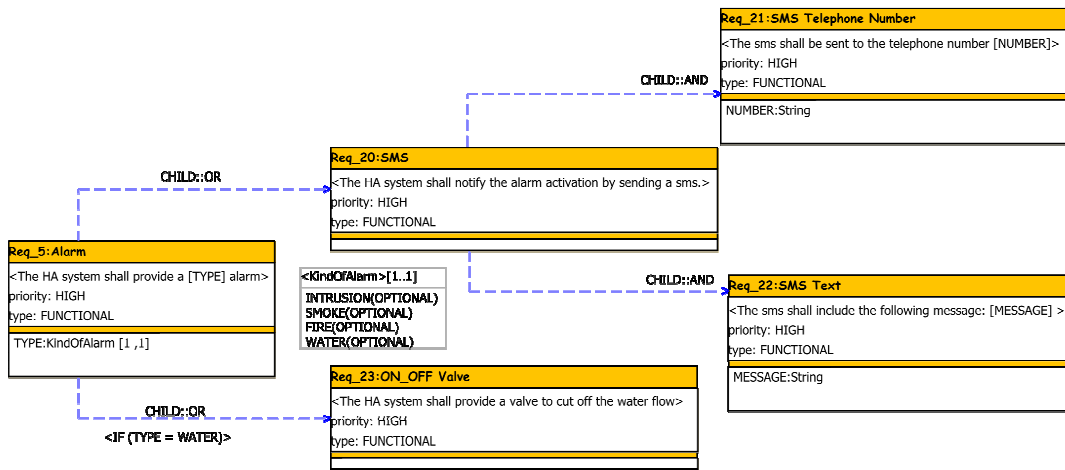


Figura 35 Requisitos parametrizados.

Cuando se selecciona un requisito parametrizado en el momento de la reutilización, el requisito se tiene que instanciar, esto es, se debe dar un valor a cada uno de los parámetros definidos en el requisito reutilizable. En este caso, el requisito del producto que se define a partir del requisito parametrizado contendrá tantas instancias de parámetro (*ParameterInstance*) como parámetros tuviera el requisito parametrizado. La instancia de parámetro mantiene una referencia al parámetro que está instanciando (*instanceOfParameter*) para tener acceso a la definición del parámetro (nombre y tipo) ya que el valor asignado a dicha instancia debe ser consistente con el tipo dado al parámetro. Los valores posibles para instanciar un parámetro son: valores numéricos (*NumberValue*), cadenas de texto (*StringValue*), uno de los valores del enumerado, si la cardinalidad máxima es uno (*EnumLiteral*) o bien un conjunto de valores del enumerado (*LiteralSet*), si la cardinalidad máxima definida en el tipo enumerado es mayor que uno.

Cualquier requisito reutilizable almacenado en el repositorio puede ser seleccionado en el momento de la reutilización. Sin embargo, debemos prestar especial atención a los requisitos que se hayan definido con prioridad obligatoria (*PriorityType::MANDATORY*) pues en el caso de los catálogos de dominio se pueden considerar como los requisitos comunes de la SPL que tienen que incluirse en la especificación de todos los productos que pertenezcan a este dominio de aplicación.

Por último, hay que destacar que, si bien los requisitos contenidos tanto en un catálogo reutilizable como en un catálogo de producto, están relacionados mediante las relaciones de traza descritas en la Sección 5.3, en el catálogo de requisitos de producto no tiene sentido la existencia de trazas exclusivas. Este tipo de traza sólo tiene sentido en el modelado *para* reutilización, para indicar requisitos alternativos pero, una vez que se configuran los requisitos de un nuevo producto, sólo uno de ellos podrá formar parte de la especificación de requisitos.

5.5. Modelos bien formados. Restricciones.

La validación de los modelos, esto es, la comprobación de que los modelos están bien formados, tiene una doble vertiente. Por un lado, se comprueba que los modelos son conformes al metamodelo que se ha definido. La especificación de REMM incluye qué relaciones son posibles entre los conceptos, sus cardinalidades (por ejemplo, todo parámetro tiene que tener asociado un tipo), los atributos que son obligatorios (como el *id* y el *texto* del requisito) y aquellos que deben tener un valor único (como el *nombre* del catálogo, el *id* del requisito, el *nombre* del Stakeholder o el *identificador* del caso de prueba). Además, se han añadido restricciones adicionales implementadas formalmente en el lenguaje de restricciones OCL (*Object Constraint Language*) o bien como funciones en el lenguaje de programación Java. Las restricciones implementadas son:

- En los modelos de requisitos *para* reuso (todas las restricciones implementadas en OCL se encuentran especificadas en la Tabla 36, el resto se han implementado como funciones Java):
 - R1. Los parámetros definidos deben estar contenidos en el texto del requisito.
 - R2. Los parámetros especificados en el texto del requisito deben estar entre corchetes y tienen que estar definidos.
 - R3. Los parámetros de un mismo requisito tienen que tener nombres diferentes.
 - R4. El mínimo y el máximo de un tipo enumerado (*EnumType*) tienen que estar definidos de forma consistente, el mínimo tiene que ser menor que el máximo y valores mayores que cero.
 - R5. En la definición de un tipo enumerado, el número de variantes (*EnumLiteral*) obligatorias no puede ser mayor que el número máximo de establecido en el enumerado.
 - R6. Las condiciones tienen que estar bien formadas, esto es, tienen que ser conformes a la gramática definida con este fin (Sección 5.5.1).
 - R7. Los identificadores utilizados en las condiciones se corresponden con los parámetros especificados en los requisitos origen y destino de la traza.

Modelos de requisitos para reutilización	
Descripción	R3. Los parámetros de un requisito tienen que tener nombres diferentes
Metaclase objetivo	ReusableRequirement -> Requirement
Restricción OCL	self.parameters->forall(p1,p2 p1<>p2 implies p1.name<>p2.name)
Descripción	R4. El mínimo y el máximo de un tipo enumerado tienen que estar definidos de forma consistente
Metaclase objetivo	EnumType -> Type
Restricción OCL	self.minimum<=self.maximum and self.minimum>0
Descripción	R5. El número de variantes obligatorias de un tipo enumerado no puede ser mayor que el número máximo de establecido en el enumerado.
Metaclase objetivo	EnumType -> Type
Restricción OCL	(self.literals->select (literal literal.mandatoryLevel=MandatoryLevelType::MANDATORY)) ->size() <= self.maximum

Tabla 36 Restricciones OCL en el modelado para reutilización.

- En los modelos de requisitos *con* reuso (todas las restricciones implementadas en OCL se encuentran especificadas en la Tabla 37, el resto se han implementado como funciones Java):
 - R8. En un modelo de requisitos en la faceta con reuso no puede existir dos requisitos procedentes de un catálogo de requisitos reutilizable, que estén relacionados mediante una traza exclusiva, se tiene que elegir uno de los dos (*DependenceTrace::EXCLUDES*).
 - R9. No pueden existir requisitos excluyentes, esto es, relacionados con una relación de traza *excludes*, en un modelo que represente la especificación de requisitos de producto.
 - R10. Todos los requisitos reutilizados a partir de un requisito parametrizado tiene que instanciar los parámetros.
 - R11. Los valores asignados a los parámetros de dos requisitos implicados en una relación de traza restringida por una condición, tienen que ser tales que se cumpla la condición establecida (la evaluación de la condición es *true*).
- Respecto a las relaciones de traza, común a las dos facetas (Tabla 38):
 - R12. Salvo en las trazas de refinamiento, el nivel de abstracción del requisito origen y destino de una traza inter-requisitos tiene que ser el mismo. En el caso de las trazas de refinamiento, las relaciones establecidas son las que se muestran en la Figura 29.
 - R13. No se permiten trazas recursivas, el origen y destino de una traza tiene que ser un requisito diferente.

Como veremos en el Capítulo 7, cuando presentemos la herramienta de soporte construida a partir de REMM y que hemos denominado REMM-Studio⁺, los modelos de

requisitos serán validados siempre automáticamente en el momento en el que son salvados en memoria.

Modelos de requisitos con reutilización	
Descripción	R8. No puede haber en un catálogo de requisitos de producto, requisitos reutilizados que se relacionen mediante trazas exclusivas.
Metaclase objetivo	ProductCatalog -> Catalog
Restricción OCL	self.productReq->forall(r1, r2 r1.reusedFrom<>null and r2.reusedFrom<>null and r1.reusedFrom.ownerRC=r2.reusedFrom.ownerRC implies not r1.reusedFrom.ownerRC.intertraces->exists(t (t.source=r1.reusedFrom and t.target=r2.reusedFrom) or (t.source=r2.reusedFrom and t.target=r1.reusedFrom) and (t.oclAsType(DependenceTrace)).type = DependenceType::EXCLUDES))
Descripción	R9. No puede haber requisitos excluyentes en un catálogo de producto
Metaclase objetivo	DependenceTrace -> InterReqTrace
Restricción OCL	(self.source.oclIsTypeOf(ProductRequirement) and self.target.oclIsTypeOf(ProductRequirement)) implies (self.type <> DependenceType::EXCLUDES)

Tabla 37 Restricciones OCL en el modelado con reutilización.

Restricciones comunes a las facetas para y con reutilización	
Descripción	R12. Salvo las trazas de refinamiento el resto de trazas relacionan requisitos del mismo nivel de abstracción.
Metaclase objetivo	DependenceTrace -> InterReqTrace
Restricción OCL	if not self.oclIsKindOf(RefinementTrace) then self.source.abstractionLevel = self.target.abstractionLevel else (self.source.abstractionLevel= AbstractionLevelType::BUSINESS_REQUIREMENTS and self.target.abstractionLevel= AbstractionLevelType::SYSTEM) or (self.source.abstractionLevel= AbstractionLevelType::BUSINESS_REQUIREMENTS and self.target.abstractionLevel= AbstractionLevelType::FEATURE) or (self.source.abstractionLevel=AbstractionLevelType::FEATURE and self.target.abstractionLevel= AbstractionLevelType::SOFTWARE) or (self.source.abstractionLevel=AbstractionLevelType::SYSTEM and self.target.abstractionLevel= AbstractionLevelType::SOFTWARE) or (self.source.abstractionLevel= AbstractionLevelType::BUSINESS_REQUIREMENTS and self.target.abstractionLevel=AbstractionLevelType::BUSINESS_RULES) or (self.source.abstractionLevel=AbstractionLevelType::SYSTEM and self.target.abstractionLevel=AbstractionLevelType::BUSINESS_RULES) or (self.source.abstractionLevel=AbstractionLevelType::SOFTWARE and self.target.abstractionLevel=AbstractionLevelType::BUSINESS_RULES) endif
Descripción	R13. No se permiten trazas recursivas
Metaclase objetivo	Catalog
Restricción OCL	self.intertraces->forall (t t.target <> t.source)

Tabla 38 Restricciones OCL en REMM.

5.5.1. Gramática del lenguaje para la definición de condiciones

La condición de una relación de traza entre requisitos (*InterReqTrace*) es una expresión booleana, expresada en una cadena de texto, que establece las condiciones que se tienen que cumplir para que esa relación de traza exista. Como hemos visto en el ejemplo de la Figura 35, la definición de esta condición tiene sentido fundamentalmente en las relaciones entre requisitos parametrizados, para especificar restricciones en los valores de los parámetros. La gramática definida para la definición de las condiciones es la siguiente:

```

<sentencia> ::= <sentenciaSimple> { ; <sentenciaSimple> }
<sentenciaSimple> ::= IF (<expresion>) |
    IF (<expresion>) THEN (<expresion>) |
    IF (<expresion>) THEN (<expresion>) ELSE (<expresion>)
<expresion> ::= <expresionLogica> | <expresionRelacional>
<expresionLogica> ::= <expresion> <operadorLogico> <expresion>
<operadorLogico> ::= && | ||
<expresionRelacional> ::= <identificador> <operadorRelacional> <valor>
<operadorRelacional> ::= = | < | > | <= | >= | !=
<identificador> ::= <cadenaTexto>
<valor> ::= <cadenaTexto> | <numero>
<cadenaTexto> ::= <letra> { _ | <letra> | <digito> }
<letra> ::= A..Z | a..z
<numero> ::= <digito> {<digito>}
<digito> ::= 0..9

```

El significado del operador > está sobrecargado. Esto es, en el contexto de valores numéricos o cadenas de texto representa la relación de orden *mayor*, mientras que en el caso de valores multivaluados (un enumerado del que se pueden seleccionar más de un valor) el operador significa que el valor de la derecha está incluido en el conjunto de valores de la izquierda. Por ejemplo, la expresión MECANISMO > PULSADOR significa que el valor PULSADOR está definido en el conjunto de mecanismos establecido.

De acuerdo a esta gramática serían correctas las siguientes expresiones:

- (1) IF (SO = Windows_Vista) THEN (MB = 500)
- (2) IF (ACTION = REGULATE && MECHANISM > PUSH_BUTTON)

En (1) se establece que si el sistema operativo elegido es Windows Vista entonces la cantidad de memoria mínima que requiere el sistema tiene que ser de 500 MB y en (2) se especifica que la acción a realizar es la de regular la intensidad de las luces y entre los mecanismos seleccionados debe estar el botón de pulsación.

Para la validación de los modelos de requisitos no es suficiente con que la condición esté bien formada, esto es, que sea conforme a la gramática que hemos definido, sino que también habrá que comprobar que los identificadores y los valores que se han utilizado en la expresión sean correctos. Así, un identificador de la parte del IF de la

expresión será correcto si se corresponde con alguno de los parámetros definido en el requisito fuente (*source*) de la traza. Por su parte, un identificador de la parte THEN...ELSE de la expresión será correcto si se corresponde con alguno de los parámetros del requisitos destino (*target*) de la traza. Además, si el tipo del parámetro es un valor enumerado, la cadena de texto con la que se compara el identificador tiene que ser uno de los posibles valores del enumerado. Por ejemplo, en las expresiones mostradas arriba como ejemplo, el tipo del parámetro ACTION podría ser ActionType, un enumerado con los valores {ON_OFF, REGULATE}. En la Figura 35 se muestra otro ejemplo de condición en la que se tiene que comprobar que en el requisito fuente (Req_5) exista un parámetro de nombre TYPE y que WATER sea uno de los valores del enumerado.

5.6. Formalización de SIREN en REMM

A lo largo del capítulo se ha ido desgranando cómo se han implementado los conceptos de SIREN (Capítulo 4) en REMM. En esta sección recopilamos toda esta información a modo de resumen. La correspondencia entre los conceptos definidos en SIREN (Capítulo 4) y su implementación en REMM se resume en la Tabla 39. En la tabla podemos ver como algunos de los atributos propuestos en el conjunto MIN de SIREN han sido omitidos en la revisión que se ha hecho en REMM. Indicamos a continuación qué atributos han sido omitidos y su justificación:

- 1) *criticidad*: consideramos que viene reflejada en el valor que el ingeniero le otorgue a la *prioridad*, de hecho, en las guías de SIREN se establece que la prioridad se asigne en función del riesgo que entrañe el requisito y cómo sea considerado de crítico para el sistema.
- 2) *sección*: puesto que está ligado al formato del documento que se haya escogido. Nuestra idea es que el modelo de requisitos sea independiente del documento de especificación de requisitos, de manera que, un mismo modelo de requisitos puede ser descrito utilizando distintos formatos (IEEE-830, VOLERE, etc.). En el marco del DSDM la generación del documento de requisitos se puede generar de manera automática mediante una transformación modelo a texto como veremos en el Capítulo 6.
- 3) *versiones*: dado que el control de versiones será soportado por la herramienta encargada de la gestión de modelos, hemos decidido que no se maneje de forma explícita en el metamodelo.

5.7. Conclusiones y aportaciones al estado del arte

En este capítulo hemos presentado REMM, un metamodelo para dar soporte a la fase de Ingeniería de Requisitos y que soporta la reutilización sistemática de requisitos en sus dos vertientes: modelado de requisitos *para* reutilización y modelado de requisitos *con* reutilización. Para ello, REMM incluye el modelado explícito de la variabilidad (en la faceta *para* reutilización) mediante la definición de requisitos parametrizados, trazas exclusivas y requisitos hijos opcionales.

SIREN		REMM
Catálogos reutilizables de dominio o perfil		Modelo de requisitos <i>para</i> reutilización ReusableCatalog>> atributo type: CatalogType CatalogType::DOMAIN CatalogType::PROFILE
Catálogo de requisitos para un nuevo producto		Modelo de requisitos <i>con</i> reutilización ProductCatalog
Niveles de requisitos (Sección 4.3.1):		Requirement >> atributos: type : RequirementType abstractionLevel : AbstractionLevelType
Requisitos del negocio Requisitos del sistema Requisitos funcionales Reglas del negocio Requisitos no funcionales Características (<i>features</i>) Casos de uso		AbstractionLevelType::BUSINESS_REQUIREMENT AbstractionLevelType::SYSTEM RequirementType::FUNCTIONAL AbstractionLevelType::BUSINESS_RULE RequirementType::NON_FUNCTIONAL Se ha definido un conjunto extenso que caracteriza a los distintos tipos de requisitos no funcionales. AbstractionLevelType::FEATURE RequirementType::FUNCTIONAL
Atributos de requisitos (Sección 4.3.2):		
MIN	texto PUID riesgo criticidad prioridad motivación estado fuente criteriosValidación propuestoPor sección históricoVersiones	Requirement>> atributo <i>text</i> Requirement>> atributo <i>id</i> Requirement>> atributo <i>risk</i> No implementado Requirement>> atributo <i>priority</i> Requirement>> atributo <i>rationale</i> Requirement>> atributo <i>status</i> Relación de traza extra-requisito <i>source</i> Relación de traza extra-requisito <i>validatedBy</i> Relación de traza extra-requisito <i>proposedBy</i> No implementado No implementado
Requisitos Parametrizados (Sección 4.3.3)		Requisito reutilizable con parámetros
Relaciones de Traza (Sección 4.3.4)		
Padre-hijo requiere relacionadoCon exclusiva materializaEn		ParentChildTrace DependenceTraceType::REQUIRES DependenceTraceType::DEPENDS DependenceTraceType::EXCLUDES ForwardTrace

Tabla 39 Implementación de SIREN en REMM.

En cuanto a las aportaciones al estado del arte, en el Capítulo 3 se presenta una revisión sistemática de la literatura (RSL) en relación a los metamodelos de requisitos propuestos hasta el momento. El objetivo de esta RSL era estudiar las cualidades de los metamodelos para dar soporte a las relaciones de traza *Pre-RS* y *Post-RS*. Para ello establecimos un marco de comparación que, en el caso de las relaciones de traza *Pre-RS* incluye las buenas prácticas de IR propuestas por Sommerville y Sawyer [329], y en el caso de las relaciones de traza *Post-RS* incluye las relaciones de traza que a juicio de

Dahlstedt y Persson [67] son consideradas como fundamentales. Aplicando este marco de comparación a REMM, obtenemos el resultado que se muestra en la Tabla 40.

En la tabla podemos ver cómo todos los tipos de trazas definidas en el marco de comparación están definidas en REMM. Esto quiere decir que, hasta donde nosotros sabemos, REMM es el único metamodelo propuesto hasta el momento que se preocupa, no sólo de las relaciones de traza Post-RS, sino que tiene en cuenta las relaciones de traza Pre-RS, consideradas fundamentales por Gotel y Finkelstein [111]. Esto quiere decir, que la definición de modelos conformes a REMM va a servir, no sólo para integrar los requisitos en el DSDM sino para dar soporte a las buenas prácticas de IR. Además, también hasta donde nosotros sabemos, REMM es el único modelo propuesto que integra las dos facetas de la reutilización (*para* y *con*) en el mismo entorno.

MM	Trazabilidad Pre-RS				Trazabilidad Post-RS						
	Requisito-fuente		Requisito-razón	Requisito-suplemento	Requisito-Requisito					Requisito-Caso de prueba	Req-artefact
	Req-stakeholder	Req-doc			<i>requiere</i>	<i>refina</i>	<i>conflicto-con</i>	<i>influye</i>	<i>coste/valor</i>		
REMM	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓	✓

Tabla 40 Trazabilidad Pre- y Post-RS en REMM.

La definición de REMM es el primer paso necesario para el desarrollo de una herramienta que dé soporte a la edición y transformación de modelos de requisitos, y también para la integración de los modelos de requisitos en un proceso completo de DSDM, que vamos a presentar en los capítulos siguientes.

Capítulo 6

ReToCo: un proceso completo de DSDM, desde los requisitos hasta el código

6.1. Introducción

Como vimos en la introducción del Capítulo 5, la *Ingeniería Dirigida por Modelos* (*Model Driven Engineering*, MDE) o el *Desarrollo de Software Dirigido por Modelos* (DSDM) es una disciplina de la Ingeniería del Software que confía en los *modelos* como entidades de primer nivel y tiene como objetivo desarrollar, mantener y evolucionar el software ejecutando *transformaciones de modelos* [213]. Dado que contar con un metamodelo preciso es un pre-requisito para poder definir transformaciones de modelos automáticas [213], en el Capítulo 5 se detallan los conceptos y relaciones que conforman REMM, el metamodelo que da soporte a la especificación de requisitos textuales. Ahora, vamos a presentar las propuestas de transformación de modelos a partir de los modelos de requisitos conformes a REMM. Estos dos elementos (metamodelo y transformaciones de modelos) son las piezas claves para la definición de un proceso de Ingeniería de Requisitos (IR) dirigido por modelos que se integre en un proceso de DSDM completo, que abarque desde los requisitos hasta el código. El objetivo final que se persigue es la integración de las especificaciones de requisitos textuales en el enfoque DSDM, potenciando la trazabilidad y reutilización de requisitos.

Una transformación de modelos se refiere al proceso por el cual se toma como entradas uno o más modelos fuente (*source models*) y se generan como salida uno o más modelos destino (*target models*), siguiendo un conjunto de *reglas de transformación* que conforman la *definición de la transformación* [315]. Una herramienta de transformación ejecuta una transformación sobre un modelo fuente específico de acuerdo a las reglas de transformación definidas (Figura 36).

De acuerdo con Sendall y Kozaczynski [315], para que la visión del DSDM sea una realidad, las herramientas tienen que ser capaces de dar soporte a las transformaciones de modelos de forma automática. Las herramientas de desarrollo deberían no sólo ofrecer la posibilidad de aplicar transformaciones de modelos predefinidas bajo demanda, sino también ofrecer un lenguaje que permita a los usuarios (avanzados) definir sus propias transformaciones de modelos y entonces ejecutarlas bajo demanda.

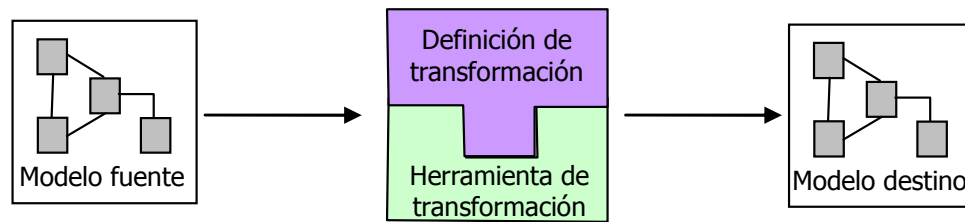


Figura 36 Proceso de transformación de modelos.

De acuerdo con el proceso definido en la Figura 36, la herramienta de transformación típicamente utiliza las mismas reglas para cada transformación sea cual sea el modelo de entrada. Para conseguir que las transformaciones se ejecuten una y otra vez, con independencia del modelo de entrada, las reglas de transformación deben relacionar constructores del lenguaje fuente con los constructores del lenguaje destino. De manera general podemos decir que [170] (Figura 37):

Una *transformación* es la generación automática de un modelo destino a partir de un modelo fuente, de acuerdo a una *definición de transformación*.

Una *definición de transformación* es un *conjunto de reglas de transformación* que juntas describen cómo un modelo en el lenguaje fuente puede ser transformado en un modelo en el lenguaje destino.

Una *regla de transformación* es una descripción de cómo uno o más constructores en el lenguaje fuente se pueden transformar en uno o más constructores en el lenguaje destino.

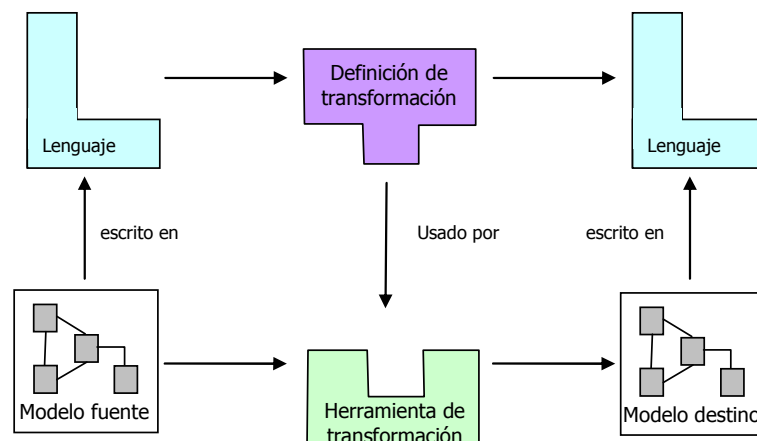


Figura 37 Marco de transformación de modelos. Esta figura es una generalización al DSDM de la propuesta del marco de MDA [170].

Czarnecki y Helsén [57] clasifican las transformaciones en dos grandes categorías: 1) transformaciones *modelo a código* (*m2c*), también denominadas *modelo a texto* (*m2t*) dado que se pueden generar artefactos que no sean código (como XML), y 2) transformaciones *modelo a modelo* (*m2m*). Según estos mismos autores, las transformaciones modelo a código se pueden ver como un caso especial de

transformaciones *m2m*, puesto que sólo es necesario proporcionar el metamodelo del lenguaje de programación destino. Sin embargo, el código a menudo se genera simplemente como texto que constituye la entrada del compilador. En esta tesis doctoral, como veremos en este capítulo, se definen transformaciones tanto *m2t* como *m2m* para generar, entre otros, los documentos de especificación de requisitos y el modelo de la aplicación conforme a un DSL (*Domain Specification Language*), respectivamente.

Por su parte, Mens y Van Gorp [213] proponen una taxonomía de transformaciones de modelos que permita agrupar herramientas, técnicas o formalismos para las transformaciones de modelos en base a sus cualidades comunes. En dicha clasificación, los autores distinguen entre dos dimensiones ortogonales:

- Transformaciones *endógenas* y *exógenas*, atendiendo al lenguaje en el que se expresan los modelos fuente y destino. Las transformaciones endógenas son transformaciones entre modelos expresados en el mismo lenguaje. Las transformaciones exógenas son transformaciones entre modelos expresados utilizando lenguajes diferentes.
- Transformaciones *horizontales* y *verticales*, atendiendo al nivel de abstracción de los modelos fuente y destino. Una transformación horizontal es una transformación en la que los modelos fuente y destino residen al mismo nivel de abstracción. Mientras que una transformación vertical es aquella en la que los modelos fuente y destino se encuentran a distinto nivel de abstracción.

En la propuesta que vamos a presentar en esta tesis doctoral, las principales transformaciones modelo a modelo que se han definido son *transformaciones exógenas y verticales*, puesto que el proceso de transformación toma como entrada los modelos de requisitos y los transforma en modelos de la aplicación conformes a un DSL. Estos modelos (el de requisitos y el de la aplicación) se encuentran a distintos niveles de abstracción, y se expresan utilizando distintos lenguajes.

Por último, dado que el *Modelado Específico de Dominio* (DSM, *Domain Specific Modelling*) promulga la generación completa del código a partir de los modelos [163], nuestra propuesta se basa en este enfoque para generar el código de la aplicación a partir de los modelos de aplicación generados automáticamente a partir de los requisitos. De esta forma, añadiendo la capa de requisitos a partir de la cual se genera el modelo de la aplicación, es posible obtener automáticamente parte del código de la aplicación automáticamente a partir de las especificaciones de requisitos. Como veremos en el Capítulo 8, nuestra propuesta se ha particularizado para los sistemas domóticos, para los que se ha propuesto un DSL denominado Habitation [148] y cuenta con un editor de modelos y un generador de código de la aplicación a partir de los modelos domóticos. La propuesta DSM para los sistemas domóticos forma parte de otra línea de investigación que queda fuera del ámbito de esta tesis doctoral. Aunque los detalles del caso de estudio se verán posteriormente, a lo largo de este capítulo se utilizará este caso de estudio para proporcionar ejemplos que favorezcan la legibilidad y la comprensión del proceso general.

En este capítulo presentaremos en primer lugar el proceso general de DSDM que parte de los modelos de requisitos (Sección 6.2). Las piezas claves de este proceso son el enfoque de IR (*Ingeniería de Requisitos*) dirigido por modelos (SIREN_{MD}, Sección 4.7) que gira en torno al metamodelo de requisitos REMM (Capítulo 5) y la transformaciones *m2m* y *m2t* definidas a partir de los modelos de requisitos. El detalle sobre las

transformaciones m2m y m2t se pospone hasta las Secciones 6.3 para hacer más ligera la presentación del proceso completo.

6.2. ReToCo: un proceso DSDM desde los requisitos hasta el código

Uno de los aspectos centrales que han motivado el trabajo de esta tesis doctoral es la reutilización de requisitos (Capítulo 4). En este contexto, es inevitable tener en cuenta el enfoque de Líneas de Producto Software (*Software Product Line, SPL*) [53] dado que ha sido reconocido como una de las técnicas líderes para conseguir los objetivos de reutilización. Los procesos implicados en la especificación de requisitos para una SPL son *ingeniería del dominio* e *ingeniería de la aplicación*, que claramente se corresponden con las facetas *para* y *con* reutilización propuestas en SIREN (Sección 4.1). Por un lado, los objetivos de la ingeniería del dominio (modelado de requisitos *para* reutilización) son la identificación y establecimiento de los requisitos comunes y los puntos de variación de una SPL. Por otro lado, los objetivos del proceso de ingeniería de la aplicación (modelado de requisitos *con* reutilización) es la especificación de requisitos de un producto específico dentro de la SPL, configurando de forma adecuada los puntos de variación especificados previamente.

Por ejemplo, en el caso de los modelos de requisitos conformes a REMM, durante el proceso de ingeniería del dominio se definen los modelos de requisitos reutilizables, donde se especifican los puntos de variación mediante requisitos parametrizados, trazas exclusivas y trazas padre-hijo (Sección 5.4). Durante el proceso de ingeniería de la aplicación, se seleccionan los requisitos que deben reutilizarse y se instancian los requisitos parametrizados escogidos. La instanciación de los requisitos parametrizados supone establecer los valores adecuados de los parámetros en función del producto concreto de la familia que se esté especificando. El enfoque general de DSDM que proponemos en esta tesis doctoral, cubre los dos procesos asociados al desarrollo de una SPL y el proceso de IR a seguir es el proceso de IR dirigido por modelos (SIREN_{MD}) presentado en la Sección 4.7. De esta forma, el proceso SIREN_{MD} se integra en un proceso completo de DSDM denominado ReToCo (*From Requirements To Code*) como vamos a desarrollar a continuación.

El proceso completo de DSDM se describe globalmente en la Figura 38, detallando cada uno de los subprocesos, ingeniería del dominio e ingeniería de la aplicación, en las Secciones 6.2.1 y 6.2.2, respectivamente. En la Sección 6.2.3 se presentan los criterios de calidad tanto del proceso de transformación m2m como de los modelos generados. Por último, en la Sección 6.2.4 se discute la conveniencia de definir en un mismo proceso los modelos de requisitos y los modelos de dominio, dado que hay quien considera que son modelos que se encuentran al mismo nivel de abstracción.

De forma general, en la Figura 38 vemos como durante la fase ingeniería del dominio se define: 1) el DSL, que incluye los conceptos del dominio; 2) la especificación de requisitos del dominio, que se representa mediante un modelo de requisitos reutilizables conforme a REMM y constituye un lenguaje de especificación de requisitos (RSL, *Requirements Specification Language*); y 3) las reglas de transformación m2m, que van a permitir transformar los modelos de requisitos en modelos de la aplicación conformes al DSL.

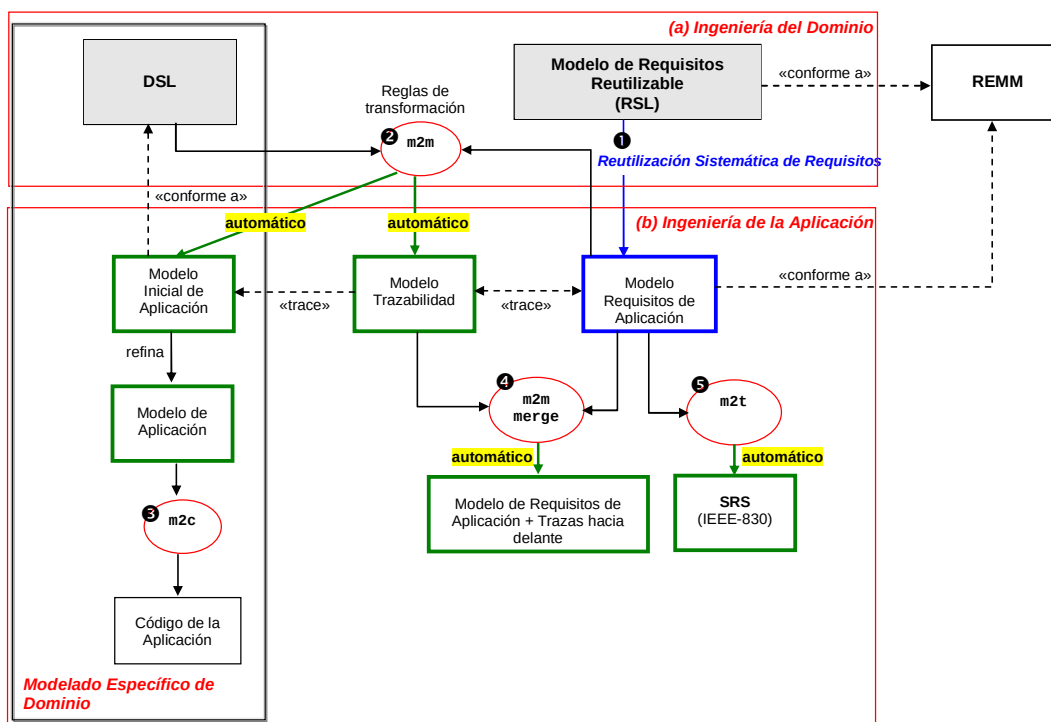


Figura 38 Visión general del enfoque completo de DSDM: ReToCo.

Una vez analizado el dominio, durante la fase de ingeniería de la aplicación, se pueden especificar los requisitos de una nueva aplicación dentro del dominio, reutilizando los requisitos definidos en el catálogo de requisitos reutilizables (❶ en Figura 38). El resultado del proceso de reutilización sistemática de requisitos será un modelo de requisitos, conforme a REMM, que representa la especificación de requisitos de la aplicación. Este modelo de requisitos de la aplicación se puede transformar automáticamente, ejecutando las reglas de transformación m2m definidas en la fase de ingeniería del dominio (❷ en Figura 38), dando lugar, por un lado, al modelo inicial de la aplicación conforme al DSL. Este modelo inicial puede ser refinado, manualmente, utilizando el editor disponible para el DSL, hasta llegar al modelo final de la aplicación, que será entonces transformado en el código de la aplicación, ejecutando las transformaciones m2c predefinidas (❸ en Figura 38). Por otro lado, se generará el modelo de traza que conecta los conceptos del dominio con los requisitos especificados. Este modelo de traza se puede añadir al modelo de requisitos inicial mediante transformaciones m2m (❹ en Figura 38), lo que permite tener todas las relaciones de los requisitos en un mismo modelo. Por último, a partir de los modelos de requisitos se pueden generar los documentos con la especificación de dichos requisitos de acuerdo a las plantillas establecidos en la organización (por ejemplo, IEEE 830 [131]) mediante transformaciones m2t (❺ en Figura 38).

A continuación se detalla cada uno de los procesos implicados en la propuesta, cada una de las sub-secciones de los apartados 6.2.1 (proceso de ingeniería del dominio) y 6.2.2 (proceso de ingeniería de la aplicación) se corresponde con una de las tareas definidas como parte del proceso.

6.2.1. Proceso de Ingeniería del Dominio

El punto de partida de ReToCo es el proceso de ingeniería del dominio. El análisis de un dominio concreto nos va permitir obtener las herramientas necesarias para las transformaciones de las especificaciones de requisitos en modelos del dominio. Las tareas implicadas en el proceso son las que se muestran en la Figura 39. En las secciones siguientes se detallan los objetivos, roles implicados, entradas y entregables de cada una de ellas.

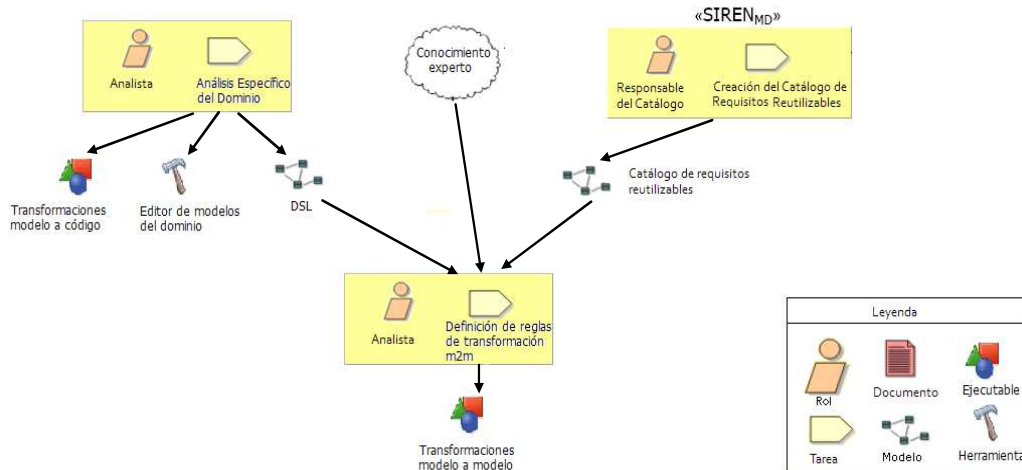


Figura 39 Proceso de Ingeniería del Dominio de ReToCo.

6.2.1.1. Tarea - Creación del catálogo de requisitos reutilizables

Objetivos: esta tarea se corresponde con la tarea homónima en el modelo de proceso SIREN_{MD} (Sección 4.7), que tiene que ver con la faceta de modelado de requisitos *para* reutilización. Se trata de definir un modelo de requisitos conforme a REMM que describa la especificación de requisitos reutilizables para un dominio concreto, esto es, un catálogo de requisitos reutilizables de dominio.

El modelo de requisitos reutilizable que se defina para un dominio específico, como podría ser el de los sistemas domóticos, se puede considerar como un lenguaje de especificación de requisitos (RSL, *Requirements Specification Language*) basado en requisitos parametrizados (*boilerplate requirements* [122]). El uso de requisitos parametrizados permite estandarizar la forma en la que se escriben los requisitos. Como vimos en las Secciones 4.3.3 y 5.4, los requisitos parametrizados son requisitos semi-completos que determinan y restringen la forma en la que se tienen que especificar los requisitos de una aplicación concreta. De esta forma, durante el proceso de ingeniería de la aplicación, a partir del modelo de requisitos reutilizables se puede derivar la especificación de requisitos de una nueva aplicación (*Modelo de Requisitos de la Aplicación* en Figura 38), reutilizando los requisitos parametrizados del dominio más adecuados y estableciendo los valores de los parámetros que corresponda a la aplicación que se está especificando. Por tanto, se restringe la forma en la que se escriben los requisitos de la aplicación, y de esta forma es posible aplicar las reglas de transformación m2m que trazan los requisitos con los conceptos del dominio. En la

Figura 40 se muestra un ejemplo de requisito parametrizado definido en el catálogo de requisitos reutilizables para los sistemas domóticos (el contenido completo del catálogo se presenta en el Capítulo 8). El requisito tiene que ver con la especificación del sistema de iluminación en el que hay que determinar: el número de puntos de luz que se necesitan, el tipo (fluorescente o no), el mecanismo que se va a utilizar para controlar estos puntos de luz (botón, detector de presencia o sensor de iluminación) y la acción que se va a realizar (encender y apagar o regular la intensidad de los puntos de luz).

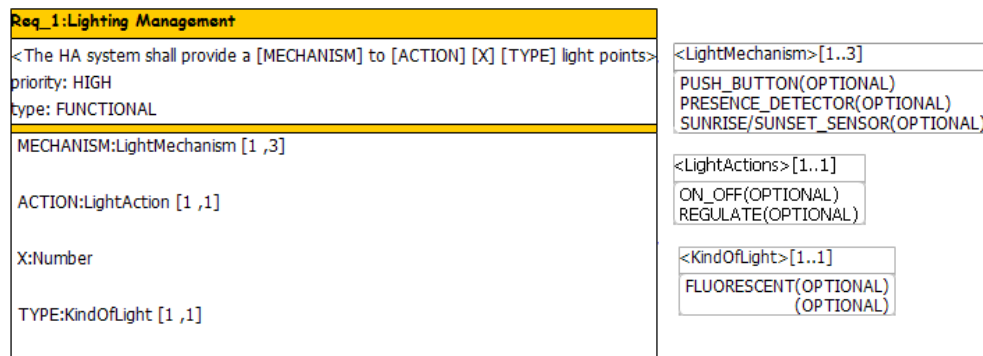


Figura 40 Requisito reutilizable parametrizado para los sistemas domóticos.

Es necesario incidir en la necesidad de utilizar un RSL restringido. Dado que la especificación de requisitos descrita por los modelos de requisitos se basa en el lenguaje natural, la ambigüedad inherente al lenguaje y la amplitud de sentencias que se pueden redactar en la especificación de requisitos, harían imposible la transformación de los modelos de requisitos a cualquier otro modelo de análisis o diseño. Yue *et al.* [368] señalan que los lenguajes que restringen el lenguaje natural son un punto de partida viable para la definición de transformaciones m2m a partir de requisitos textuales.

Roles implicados:

- Responsable:
 - Responsable del catálogo
- Participantes:
 - Analista
 - Usuario clave

Todos estos roles pertenecen al modelo de roles de SIREN que se presenta en la Sección 4.2.3.

Entregables:

- Modelo de requisitos *para* reutilización conforme a REMM que representa un catálogo de requisitos reutilizables para un dominio.

6.2.1.2. Tarea - Análisis Específico del Dominio

Objetivos: se trata de analizar el dominio para definir un metamodelo que caracterice los conceptos y relaciones relevantes en el dominio. Este metamodelo permitirá describir los modelos correspondientes a las aplicaciones del dominio. Siguiendo el enfoque DSM, junto con el DSL, se debe proporcionar un editor que permita

representar modelos del dominio, esto es, modelos conformes al DSL, y un generador de código, que permita obtener el código de la aplicación a partir de los modelos del dominio (transformaciones m2c).

Roles implicados:

- Responsable:
 - Analista
- Participantes:
 - Usuario clave

Entregables:

- El DSL, esto es, un metamodelo que especifique los conceptos relevantes del dominio y sus relaciones.
- Editor de modelos del dominio.
- Generador de código.

6.2.1.3. Tarea - Definición de las reglas de transformación m2m

Objetivos: se trata de definir las reglas de transformación m2m que establezcan las relaciones entre los requisitos especificados en el catálogo (representados en un modelo de requisitos conforme a REMM) y los conceptos definidos en el DSL. La ejecución de estas reglas de transformación permitirá generar tanto los modelos de la aplicación conformes al DSL, como los modelos de traza que enlazan los requisitos con los conceptos del dominio, creados durante el proceso de transformación, tomando como entrada un modelo de requisitos (Figura 41, ❷ en Figura 38). Realmente, estas reglas de transformación implementan las decisiones de diseño que los expertos del dominio toman cuando modelan (con el editor del DSL) la aplicación descrita en una especificación de requisitos, esto es, materializan el conocimiento de los expertos. La definición de estas reglas de transformación se explica en la Sección 6.3.1.

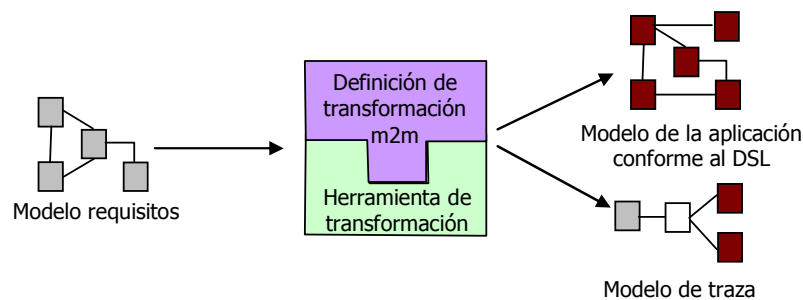


Figura 41 Modelos de entrada y salida en la transformación m2m.

Roles implicados:

- Responsable:
 - Analista
- Participantes:
 - Usuario clave

Entradas:

- DSL para el dominio que se está analizando.
- Modelo de requisitos *para* reutilización (conforme a REMM) que representa un catálogo de requisitos reutilizables.

Entregables:

- Reglas de transformación m2m de los modelos de requisitos a los modelos de dominio y de traza.

6.2.14. Instanciación del proceso para el caso de estudio de lo sistemas domóticos

Particularizando este proceso general al caso de estudio concreto de los sistemas domóticos (que desarrollaremos en el Capítulo 8), el resultado del proceso de ingeniería del dominio (Figura 42) sería: 1) la definición del DSL para los sistemas domóticos, denominado Habitation [148] que, siguiendo el enfoque DSM, dispone de un editor de modelos y generador de código de la aplicación; 2) el modelo de requisitos del dominio reutilizables (catálogo de requisitos reutilizables en términos de SIREN) para los sistemas domóticos conforme a REMM, que hemos denominado HAREL (*Home Automation Requirements Specification Language*); y 3) las reglas de transformación m2m que permiten la transformación de los modelos de requisitos de la aplicación en modelos conformes al DSL Habitation (*REMM2Habitation* en Figura 42) y el modelo de traza que liga los requisitos con los conceptos del DSL generados.

En la Sección 6.3.1.1 se explica el proceso que se ha seguido para la definición de las reglas de transformación para este caso de estudio. En concreto, en la Figura 46 se puede ver la correspondencia entre el requisito de iluminación especificado en el catálogo HAREL (Figura 40) y las unidades funcionales definidas en el DSL Habitation (botón de pulsación, sensor de presencia, punto de luz regulable, entre otros). En el Capítulo 8 se detalla el contenido del catálogo HAREL y todos los modelos de la aplicación domótica que se pueden generar a partir de los requisitos instanciados de este catálogo, que incluye los sistemas de: iluminación, motorización (persianas y toldos), alarmas (incendio, humo, intrusión e inundación) y climatización.

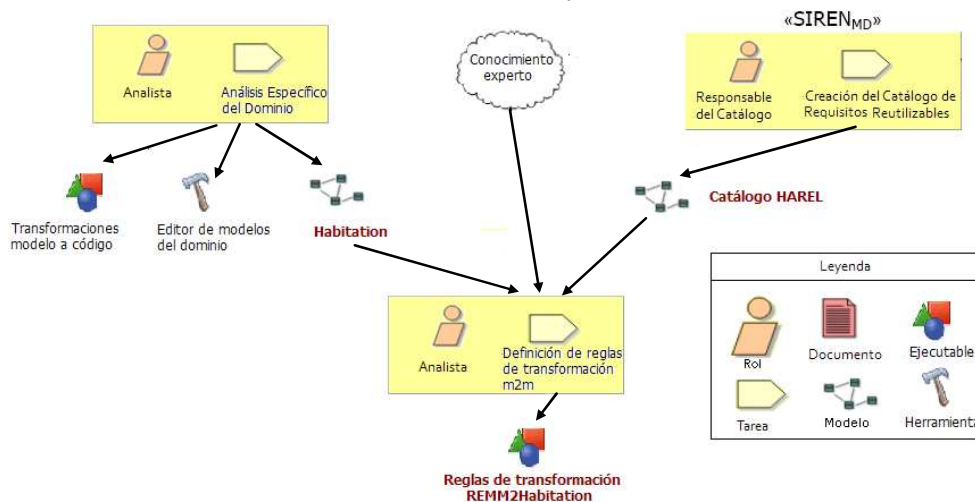


Figura 42 Proceso de Ingeniería del Dominio para los sistemas domóticos.

El enfoque DSM para el desarrollo de aplicaciones domóticas forma parte de la línea de investigación del grupo DSIE de la Universidad Politécnica de Cartagena (UPCT), que se ha materializado en la defensa de dos tesis doctorales [147, 296]. Para el desarrollo del caso de estudio hemos colaborado activamente con este grupo de investigación, como expertos del dominio, para la definición del catálogo de requisitos HAREL y las reglas de transformación. La definición del DSL, el desarrollo de los editores para el modelado de los sistemas domóticos y la generación del código de la aplicación, queda fuera del ámbito de esta tesis doctoral y se puede consultar en las tesis antes mencionadas [147, 296].

Cabe destacar que la propuesta de esta tesis doctoral es un proceso general, que se ha instanciado al dominio particular de los sistemas domóticos (Capítulo 8). Sin embargo, el proceso es perfectamente instanciable a otros dominios como las redes de sensores inalámbricos, en el que se han obtenido buenos resultados en la adopción del enfoque de DSDM [347]. Para aplicar el proceso a otros dominios, se debería abordar un proceso de ingeniería de dominio para elicitar los requisitos de dominio genéricos que serían especificados mediante un modelo de requisitos reutilizable. El conocimiento del dominio también se utilizaría para definir el DSL (editor y generador de código) y para implementar las reglas de transformación que encapsulan el modo en el que los expertos en el dominio modelan los requisitos especificados utilizando el DSL.

6.2.2. Proceso de Ingeniería de la Aplicación

Una vez que el dominio se ha analizado y definido, se puede derivar la especificación de requisitos de un producto concreto del cliente, durante el proceso de ingeniería de la aplicación (Figura 43). El objetivo final es generar el código de la aplicación, a partir de los requisitos, utilizando los productos obtenidos como resultado de las tareas del proceso de ingeniería del dominio: las reglas de transformación modelo a modelo, el editor de modelos del dominio y el generador de código del dominio. A continuación se describen los objetivos, roles implicados, entradas y entregables de las tareas que constituyen el proceso de ingeniería de la aplicación.

6.2.2.1. Sub-proceso $SIREN_{MD}$ en la faceta *con* reutilización

Objetivos: Se trata de obtener la especificación de requisitos de un nuevo producto, o aplicación, del dominio que se ha modelado. El formato de la nueva especificación de requisitos será tanto visual, mediante un modelo de requisitos conforme a REMM, como tradicional (documento textual conforme al estándar IEEE-830). La especificación de requisitos de un producto o aplicación específica de la SPL se puede obtener reutilizando sistemáticamente los requisitos del catálogo de requisitos reutilizables, descritos mediante un modelo conforme a REMM (❶ en Figura 38). La reutilización sistemática de requisitos significa que no sólo se reutilizan los requisitos seleccionados, sino también aquellos con los que están relacionados y las relaciones que existen entre ellos.

VOLERE [291]). Como veremos en la Sección 6.3.2.1, en cada caso sólo es necesario establecer la correspondencia entre los conceptos del metamodelo REMM y las secciones de la plantilla del SRS que se utilice, por ejemplo, la Tabla 41 (al final de la Sección 6.3.2.1), muestra la correspondencia entre REMM y la plantilla IEEE-830. Además, los documentos se pueden generar en el formato que se considere más adecuado. En nuestra propuesta de transformación m2t, que se desarrolla en la Sección 6.3.2.1, los documentos de requisitos se generan en formato HTML, lo que permite navegar fácilmente por la especificación de requisitos, lo que favorece la validación de la especificación de requisitos (tarea T7 en SIREN_{MD}).

Req_1: The HA system shall provide a PUSH_BUTTON to
REGULATE 3 FLUORESCENT light points.

Siendo los parámetros:

MECHANISM = PUSH_BUTTON

ACTION = REGULATE

X = 3

TYPE = FLUORESCENT

Figura 44 Instanciación del requisito de iluminación de la Figura 40.

Sólo existe una salvedad en la aplicación de SIREN_{MD} en la propuesta global de DSDM, y es que no se pueden incluir nuevos requisitos del producto que no estén contenidos en el catálogo de requisitos reutilizables (T4 *Elicitación de Requisitos del Proyecto*). La razón es sencilla, estos nuevos requisitos no estarían contemplados en la definición de las reglas de transformación implementadas durante el proceso de ingeniería del dominio, y por tanto, no serían tenidos en cuenta durante el proceso de transformación de los modelos conformes al DSL. La gestión de nuevos requisitos del producto es una cuestión importante y constituye una línea de trabajo futuro (Capítulo 9).

Roles implicados:

- Responsable:
 - Analista
- Participantes:
 - Jefe del proyecto
 - Usuario clave y típicos
 - Equipo responsable del catálogo

Entradas:

- Modelo de requisitos *para* reutilización, conforme a REMM, que representa un catálogo de requisitos reutilizables.

Entregables:

- Modelo de requisitos *con* reutilización, conforme a REMM, que representa la especificación de requisitos de una nueva aplicación, o producto, del dominio.
- Especificación de requisitos en formato textual conforme al estándar IEEE-830.

6.2.2.2. Tarea - Transformación de modelos

Objetivos: Se trata de generar automáticamente el modelo inicial de la aplicación a partir del modelo de requisitos construido aplicando el sub-proceso $SIREN_{MD}$ en su faceta *con* reutilización (Sección 6.2.2.1) y las relaciones de traza que relacionan los requisitos con los conceptos del dominio generados en el modelo de la aplicación.

Descripción: Una vez obtenida la especificación de requisitos de una nueva aplicación, siguiendo un enfoque de DSDM, el modelo de requisitos que la describe debe transformarse en otro modelo de análisis o diseño. En nuestra propuesta, el modelo de requisitos de la aplicación será la entrada del proceso de transformación que generará el modelo inicial de la aplicación conforme a un DSL concreto (modelo de análisis)-② en Figura 38. Este modelo inicial, generado automáticamente ejecutando las reglas de transformación definidas en el proceso de ingeniería del dominio, será la entrada de la tarea que generará el código de la aplicación siguiendo el enfoque de desarrollo DSM.

Durante el proceso de transformación, que genera el modelo de la aplicación a partir de los requisitos (② en Figura 38), también se registrarán, de forma simultánea, las relaciones de traza entre los requisitos y los conceptos del dominio (conceptos del DSL) generados. Estas trazas se corresponden con las trazas hacia delante (*forward traces*) definidas en la Sección 5.3.2, que relacionan los requisitos con otros artefactos del desarrollo. Por ejemplo, el requisito de iluminación de la Figura 44 estará relacionado con las unidades funcionales que se correspondan con el botón de pulsación y las luces fluorescentes regulables. Vemos, por tanto, que, a diferencia de otras propuestas, nuestro enfoque gestiona de forma automática la información de la trazabilidad relacionada con los requisitos y los artefactos de análisis. El registro explícito de las trazas hacia delante favorece el cumplimiento de las demandas de los estándares de calidad (como CMMI [309]).

Roles implicados:

- Responsable:
 - Analista
- Participantes:
 - Usuario clave

Entradas:

- Reglas de transformación m2m definidas durante el proceso de ingeniería del dominio (Sección 6.2.1.3).
- Modelo de requisitos de una nueva aplicación del dominio.

Entregables:

- Modelo inicial de la aplicación, esto es, modelo conforme al DSL, generado automáticamente a partir del modelo de requisitos.
- Modelo de traza, generado automáticamente, que enlaza los requisitos con los conceptos del dominio generados.

6.2.2.3. Tarea - Fusión de modelos

Objetivos: se trata de fusionar, mediante transformaciones de modelos (*m2m merge* en Figura 38), el modelo que representa la especificación de requisitos de la aplicación y

el modelo que especifica las relaciones de traza hacia delante, entre los requisitos y los conceptos del dominio, en un único modelo de requisitos, conforme a REMM (4 en Figura 38). Mantener las trazas hacia delante en el mismo modelo de requisitos proporciona ciertos beneficios como son:

- *Verificación de requisitos*, se puede validar en el modelo si todos los requisitos están trazados hacia artefactos de análisis, esto es, si todos los requisitos han sido tenidos en cuenta en el modelo de análisis. Esta validación forma parte de la validación del modelo de requisitos (validar que el modelo está bien formado respecto al metamodelo y a las restricciones definidas) y se ejecuta automáticamente cada vez que se salvan los modelos (la herramienta de soporte se presenta en el Capítulo 7). Por tanto, se contribuye a la ejecución de la tarea *T7 Validación de requisitos* del sub-proceso $SIREN_{MD}$.
- *Control de cambios*, al mantener la traza de los requisitos hacia los artefactos de análisis permite marcar las trazas como “sospechosas” cuando se producen cambios en los modelos de análisis. De esta forma se puede estudiar si los cambios introducidos siguen siendo compatibles con los requisitos establecidos o hay que estudiar el impacto de los cambios que se han producido.

Descripción: La fusión de modelos se llevará a cabo de manera automática, mediante transformaciones m2m. Estas transformaciones están definidas de manera general e independiente del dominio, dado que los metamodelos implicados son fijos (REMM y el metamodelo de trazabilidad que veremos en la Sección 6.3.1.2). Nótese que estas transformaciones se han representado en la Figura 43 como guía (al igual que las transformaciones m2t para generar los documentos de requisitos), mientras que las transformaciones m2m que son la entrada de la tarea *Transformación de modelos*, se han representado como un producto ejecutable dado que son el resultado de una tarea del proceso de ingeniería del dominio.

Roles implicados:

- Responsable:
 - Analista
- Participantes:
 - Usuario clave

Entradas:

- Modelo de requisitos de la aplicación.
- Modelo de trazabilidad generado automáticamente en la transformación del modelo de requisitos al modelo de la aplicación.
- Reglas de transformación m2m, previamente definidas, para fusionar el modelo de requisitos con el modelo de trazabilidad.

Entregables:

- Modelo de requisitos de la aplicación, conforme a REMM, que incluye las trazas hacia delante.

6.2.2.4. Tarea - Generar la aplicación

Objetivos: Se trata de refinar (si es necesario) el modelo inicial de la aplicación, generado automáticamente, y obtener el código de la aplicación. De manera que podemos decir que se puede obtener el código de la aplicación automáticamente a partir de la especificación de requisitos.

Descripción: El modelo de la aplicación generado podría ser refinado, u optimizado, manualmente utilizando el editor de modelos implementado como herramienta de soporte al DSL. Una vez especificado el modelo de la aplicación final, siguiendo el enfoque de DSM, que promueve la transformación de modelos de dominio a código, utilizamos el generador de código para obtener el código de la aplicación (❸ en Figura 38). Tanto el editor de modelos como el generador de código son productos obtenidos como resultado de la tarea de *Análisis Específico de Dominio* del proceso de ingeniería del dominio (Sección 6.2.1.2).

Roles implicados:

- Responsable:
 - Analista
- Participantes:
 - Usuario clave

Entradas:

- Editor de modelos conformes al DSL.
- Definición de transformaciones modelo a código.
- Modelo inicial de la aplicación, generado automáticamente en la tarea *Transformación de Modelos* (Sección 6.2.2.2).

Entregables:

- Código de la aplicación, conforme a los requisitos especificados.

6.2.2.5. Tarea - Generar la matriz de trazabilidad

Objetivos: se trata de generar automáticamente mediante transformaciones m2t la matriz de traza representada en el modelo de trazabilidad. Esta matriz servirá a modo de informe para analizar las relaciones establecidas entre los requisitos y los conceptos del DSL, verificar los requisitos y analizar el impacto de los cambios.

Roles implicados:

- Responsable:
 - Analista
- Participantes:
 - Usuario clave

Entradas:

- Modelo de trazabilidad generado automáticamente en la tarea *Transformación de Modelos* (Sección 6.2.2.2).
- Definición de transformaciones modelo a texto.

Entregables:

- Documento que incluye la matriz de traza, que muestra las relaciones entre los requisitos y los conceptos de dominio generados.

6.2.3. Criterios de calidad

Por desgracia, es difícil, si no imposible, definir un método general que sea válido para la definición de las reglas de transformación para cualquier dominio de aplicación. En consecuencia, estas reglas de transformación se tienen que definir *ad-hoc* para cada dominio, extrayendo el conocimiento de los expertos en la etapa de ingeniería del dominio. Sin embargo, la ventaja es que, una vez que se han definido las reglas, se pueden utilizar para transformar cualquier modelo de requisitos de una aplicación o producto, derivado a partir del modelo de requisitos reutilizables del dominio. El conocimiento de los expertos del dominio se almacena en forma de reglas de transformación, o lo que es lo mismo, las reglas de transformación encapsulan el modo en el que los expertos del dominio razonan cuando construyen un modelo del dominio a partir de la especificación de requisitos. Aunque podría parecer que no merece la pena el coste de definir las reglas de transformación, las ventajas que se consiguen una vez que el trabajo está hecho compensan el esfuerzo realizado. Las ventajas son:

- *Repetitiva*: una vez que se han definido las reglas de transformación se pueden aplicar tantas veces como se necesite aunque el experto en el dominio no esté presente.
- *Velocidad*: el modelo inicial de la aplicación se construye más rápido que si éste se construye desde cero utilizando el editor de modelos.
- *Fiabilidad*: el modelo de la aplicación siempre incluye todos los elementos del dominio relevantes. Los diseñadores humanos son más vulnerables a las condiciones del entorno.
- *Calidad*: el modelo de la aplicación generado se construye siguiendo las recomendaciones de los expertos en el dominio (encapsuladas en las reglas de transformación). Además, la calidad de las propias reglas se puede mejorar corrigiendo las deficiencias que se detecten durante su aplicación.

Teniendo en cuenta que el editor del DSL no proporciona ni heurísticas ni guías metodológicas con las que construir los modelos, y que las reglas de transformación se pueden considerar como un modelo de conocimiento para el dominio, es razonable pensar que la calidad de los modelos generados aplicando las transformaciones será mucho *mejor*, que la de los modelos contruidos manualmente a partir de la misma especificación de requisitos. Con el objeto de establecer claramente qué queremos decir cuando hablamos de que la calidad es *mejor*, es necesario definir los criterios de calidad para los modelos generados. Mohagheghi [217] identifica cinco propiedades de calidad básicas de los modelos, que hemos adoptado como marco de calidad de nuestra propuesta:

- *Correcta*: el modelo incluye elementos, y relaciones entre ellos, correctos y no viola las reglas y convenciones establecidas. Por tanto, cubre tanto la corrección *sintáctica* (sintaxis correcta o modelo bien formado) y la corrección *semántica* (significado y relaciones adecuadas relativas al conocimiento del dominio).

- *Completa*: el modelo tiene toda la información necesaria y tiene el nivel de detalle suficiente de acuerdo a los objetivos del modelado.
- *Consistente*: los modelos no contienen contradicciones. El proceso de transformación que genera los modelos a diferentes niveles de abstracción produce modelos consistentes (consistencia vertical) puesto que los modelos de salida mantienen las propiedades de los modelos de entrada.
- *Comprensible*: los modelos pueden ser entendidos por los usuarios que son los potenciales destinatarios de los mismos.
- *Restringido*: el modelo se ajusta al propósito del modelado y al tipo de sistema, y está restringido por los objetivos del modelado, tales como definirse al nivel de abstracción adecuado, y no contener información que no es necesaria.

Dado que el conocimiento de los expertos se recoge en las reglas de transformación, es de esperar que los modelos generados, que se obtienen como resultado del proceso de transformación, alcancen todas estas propiedades de calidad puesto que: 1) el modelo obtenido será conforme al metamodelo destino y no contendrá errores sintácticos y semánticos, y 2) se seleccionarán el conjunto mínimo de conceptos, y relaciones entre ellos, adecuados a partir de la experiencia de los expertos en el modelado de los requisitos del dominio. Esta hipótesis de trabajo ha sido validada por los expertos en el dominio para el caso concreto de los sistemas domóticos, quienes han comprobado que los modelos generados, mediante la ejecución de las reglas de transformación, cumplen los criterios de calidad que hemos establecido (Sección 6.3.1.1).

6.2.4. Discusión: transformación vertical u horizontal

Pensamos que merece la pena destacar que la pieza clave de ReToCo es el proceso de transformación de los modelos de requisitos en modelos de la aplicación puesto que ha sido considerado como casi imposible de conseguir por la comunidad de DSDM (Sección 1.1.3.4). Este hecho, que podría parecer contradictorio, se ha conseguido decrementando la distancia semántica entre los modelos fuente y destino, que se ha conseguido, a su vez, elevando el nivel de abstracción del modelo destino. Consideramos que el nivel de abstracción del DSL es lo suficientemente alto para poder trazarse desde los modelos de requisitos. Además, las reglas de transformación pueden considerarse como modelos de conocimiento que nos permiten buscar en el espacio de la solución (modelo de la aplicación conforme a un DSL) a partir de los requisitos que forman parte del espacio del problema.

En relación con los niveles de abstracción de los modelos fuente (requisitos) y destino (modelo de aplicación conforme a un DSL), es cuestionable si el nivel de abstracción del modelo destino se ha elevado tanto que la especificación de requisitos y el modelo de la aplicación específico de dominio se puede considerar que se encuentren al mismo nivel de abstracción. En este caso, de acuerdo con la taxonomía propuesta por Mens y Van Gorp [213], estaríamos hablando de transformaciones horizontales en lugar de transformaciones verticales. Sin embargo, estamos convencidos de la relevancia de ambos modelos y de la conveniencia de mantener los dos pues se corresponden con niveles de abstracción diferentes. Los motivos que sustentan este convencimiento son:

- Es conveniente que todos los procesos soporten el enfoque de requisitos tradicional [280], dado que: 1) los requisitos textuales son el formato preferido en

la industria [63] y 2) las especificaciones de requisitos textuales son la pieza clave del proceso de IR que se utiliza como medio de comunicación entre los desarrolladores y los clientes. Por tanto, es difícil que un modelo reemplace el rol de las especificaciones de requisitos textuales. Esta afirmación está en la línea de la experiencia publicada por Davis [72], quien establece que los modelos de análisis deberían complementar, en lugar de reemplazar, una especificación de requisitos en lenguaje natural.

- Los modelos conformes al DSL contienen información que no pertenece a la especificación de requisitos. Se tienen que tomar decisiones durante la construcción de los modelos conformes al DSL: seleccionar los conceptos del DSL (por ejemplo, en un sistema domótico las unidades funcionales que representan el motor de una persiana controlada mediante botones de pulsación), establecer los enlaces correctos que relacionen dichos conceptos (por ejemplo, enlazar las unidades funcionales del motor con un enlace tipo move para poder subir y bajar las persianas), determinar los valores correctos de los parámetros (si los tuviera, por ejemplo el tiempo que tarda en subirse por completo la persiana). Este tipo de información no forma parte de la especificación de requisitos propiamente sino que forma parte del conocimiento de los expertos en el dominio. Sobre este punto se incide en la Sección 6.3.1.1.

6.3. Transformaciones de modelos en el proceso ReToCo

En la sección anterior hemos presentado el modelo de proceso de ReToCo. Este proceso, siguiendo la filosofía del DSDM, se sustenta en la definición de metamodelos y de transformaciones de modelos. En las siguientes secciones vamos a profundizar en la definición de las transformaciones tanto m2m, en la Sección 6.3.1, como m2t, en la Sección 6.3.2.

6.3.1. Transformaciones modelo a modelo a partir de modelos de requisitos

De acuerdo al modelo de proceso presentado en la Figura 43, en nuestra propuesta las transformaciones m2m se van a utilizar para: generar los modelos conformes a un DSL (Sección 6.3.1.1) a partir del modelo que representa la especificación de requisitos de la aplicación, generar el modelo de traza que relaciona los requisitos con los conceptos del dominio que se han generado automáticamente (Sección 6.3.1.2) y fusionar el modelo de requisitos con el modelo de traza (Sección 6.3.1.3).

Se pueden utilizar diferentes lenguajes para definir las reglas de transformación entre los metamodelos, tales como ATL [1], QVT [259] o RubyTL [299]. Hemos elegido RubyTL en nuestra propuesta porque facilita la creación de parte de los modelos de manera imperativa, esto es, la creación de distintos elementos y las relaciones entre ellos durante el proceso de transformación. Además, el informe de errores que proporciona RubyTL resulta más comprensible que los que proporcionan los otros lenguajes, con lo que facilita la depuración del código en la implementación de las reglas. RubyTL es un lenguaje de transformación de modelos basado en reglas híbrido, que permite especificar la correspondencia entre metamodelos, implementando como un DSL

embebido en el lenguaje Ruby. RubyTL tiene una parte declarativa, basada en reglas y vínculos (*binding*). Los vínculos son un tipo de asignación especial que nos permite describir qué es transformado a partir de qué, mientras que las reglas se encargan de resolverlos (esto es, una regla se llama implícitamente para resolver un vínculo). La parte imperativa del lenguaje se proporciona por el hecho de que RubyTL está embebido en Ruby, por tanto, se puede utilizar código escrito en Ruby en la definición de las reglas.

6.3.1.1. Generación de modelos conformes a un DSL

La definición de la reglas de transformación m2m que tomen como entrada los modelos de requisitos y los transformen en modelos de dominio conformes a un DSL, es el resultado de una de las tareas implicadas en el proceso de ingeniería del dominio presentado en la Sección 6.2.1. Dado que esta tarea se tiene que realizar *ad-hoc* para cada dominio, no es posible definir un proceso, ni unas reglas de transformación, aplicables de forma general para cualquier dominio. Por tanto, a lo largo de esta sección nos vamos a referir al caso de estudio de los sistemas domóticos para explicar cómo se han definido las transformaciones m2m. Este caso de estudio se presentará con más detalle en el Capítulo 8.

Siguiendo el marco general de transformación de modelos de la Figura 37, su adaptación al caso concreto de los sistemas domóticos es el que se muestra en la Figura 45. En esta figura podemos ver cómo los modelos de requisitos escritos en el lenguaje LER (descrito por el metamodelo REMM, Sección 5.2) se transforman en modelos de la aplicación que representan un sistema domótico, escritos en el lenguaje MetaDomo, que está descrito por el metamodelo Habitation [148] (Capítulo 8). La definición de la transformación (*REMM2Habitation*) se escriben en el lenguaje RubyTL [299] y se ejecutan en el entorno Eclipse, en el marco de la herramienta REMM-Studio⁺ (Capítulo 7).

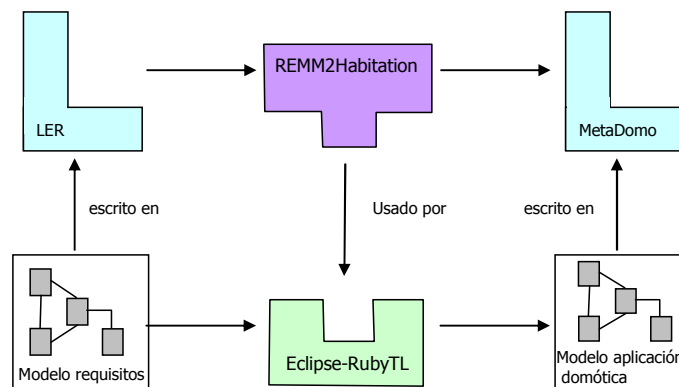


Figura 45 Transformación m2m desde los modelos de requisitos.

De manera general, la idea subyacente a la definición de la reglas de transformación es implementar una especie de sistema experto que encapsule el conocimiento de los expertos en el dominio. Al elicitar los requisitos al mismo tiempo que se está realizando el análisis específico del dominio, los requisitos se pueden escribir

de forma restringida, de manera que permitan su transformación a los conceptos del dominio automáticamente a través de las reglas. Como ya hemos visto, para restringir la escritura de los requisitos se pueden utilizar los requisitos parametrizados. El uso de parámetros nos va a ayudar a definir las reglas de transformación, pues el valor de cada parámetro determina un caso particular de diseño en el modelo de dominio. En consecuencia, es imprescindible en este proceso la participación de los expertos en el dominio (*usuarios clave*), que detallen las decisiones de diseño que hay que tener en cuenta en el modelado de una aplicación del dominio.

Por ejemplo, en el caso de estudio de los sistemas domóticos, los expertos en el dominio saben que cuando hay que modelar el *sistema de iluminación* los elementos que están implicados en el modelado son: 1) el *mecanismo* utilizado para activar la acción que se va a realizar, tales como botón de pulsación, detector de presencia o sensor de iluminación; 2) la *acción*, que puede ser simplemente encender y apagar la luz o regular la intensidad; y 3) el *tipo* de luz que se va a instalar (fluorescente o no). Las unidades funcionales que se van a utilizar en el modelo del dominio que represente la aplicación domótica serán diferentes dependiendo del mecanismo, la acción y el tipo que se elija. Esta información será la que se implemente en las reglas de transformación. En consecuencia vemos como, el conocimiento experto, se necesita tanto para especificar el requisito de iluminación de manera que se pueda transformar en unidades funcionales de los sistemas domóticos (conceptos del DSL), como para implementar las reglas de transformación que creen las unidades funcionales adecuadas según el valor de los parámetros que se establezca durante la especificación de requisitos de la aplicación (Sección 6.2.2.1). Por un lado, el requisito de iluminación se especificará como un requisito parametrizado (Figura 40) de acuerdo con los conceptos del dominio definidos en el DSL (mecanismo, acción y tipo). Por otro lado, las reglas de transformación establecen qué unidades funcionales del DSL definido para los sistemas domóticos (Habitation) se tendrán que crear cuando este requisito se incluya en el modelo de requisitos. La definición completa del DSL Habitation se detalla en el Capítulo 8. La regla analiza los valores dados a los parámetros y crea las unidades funcionales que correspondan. Por ejemplo, si se selecciona como mecanismo un botón de pulsación (PUSH_BUTTON) al instanciar el requisito de iluminación, se generará la unidad funcional correspondiente a este mecanismo (PB) en el modelo inicial de la aplicación. La Figura 46 muestra los conceptos de los sistemas domóticos relacionados con el requisito de iluminación (una de las reglas de transformación para los requisitos de iluminación se describe más abajo).

Si durante el proceso de ingeniería de la aplicación, mediante la reutilización sistemática de requisitos (❶ en Figura 38, o el sub-proceso SIREN_{MD} en Figura 43), se selecciona el requisito de iluminación y se instancia con los valores adecuados para los parámetros, por ejemplo, uso de un botón de pulsación, detector de presencia y sensor de iluminación para regular tres puntos de luz, el modelo que representa la aplicación domótica, generado automáticamente ejecutando la regla de transformación Requirement2DimmableLight (Figura 47, 48 y 49), es el que se muestra en la Figura 50. En esta figura podemos ver que, además de las unidades funcionales que habíamos indicado en la Figura 46, es necesario: 1) incluir otras unidades funcionales como los controladores de los mecanismos elegidos y los puntos de luz (DMI, DMO, SWI) o el temporizador (TM) para el detector de presencia, que controla cuándo se tiene que

apagar la luz después de llevar un tiempo sin detectar presencia; y 2) asignar valores a los parámetros asociados a las unidades funcionales, por ejemplo $TMtemp = 5$ en el temporizador (TM) establece que se tiene que apagar la luz cinco segundos después de no detectar presencia. Este conocimiento no forma parte de los requisitos sino que forma parte del conocimiento implícito en el dominio y proporcionado por los expertos. Este conocimiento se recoge en las reglas de transformación.

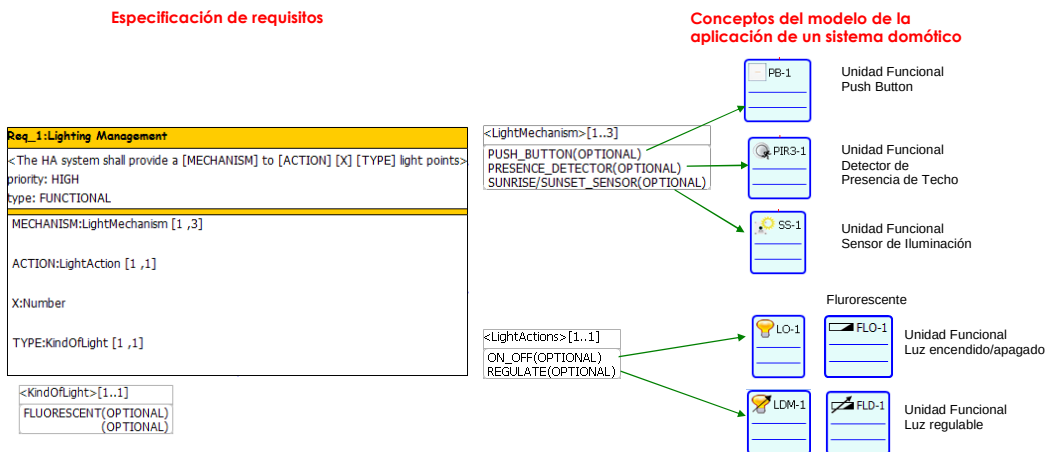


Figura 46 Relaciones entre los parámetros del requisito de iluminación y las unidades funcionales de los sistemas domóticos.

La definición de la regla de transformación Requirement2DimmableLight (Figura 47, 48 y 49) establece una correspondencia entre un requisitos de producto, que representa el requisito parametrizado instanciado (REMM::ProductRequirement) y la unidad funcional en el modelo de los sistemas domóticos destino (MetaDomoTarget::StandardFUnitInstance). Para que esta regla pueda aplicarse (condiciones especificadas en la sección filter) se tiene que cumplir que: 1) el requisitos sobre el que se va a aplicar es un requisitos de iluminación (requirement.name.include?"Light"); 2) el requisito es la instanciación de un requisito parametrizado (requirement.parameterInstances.length != 0) y la acción que se va a ejecutar sobre las luces es la de regular la intensidad (requirement.getEnumValue("ACTION") == "REGULATE"). Si se cumplen estas condiciones se ejecutará la regla de transformación que, además de generar un punto de luz regulable, creará el resto de elementos necesarios para modelar el sistema de iluminación de una aplicación domótica (Figura 50).

```

rule 'Requirement2DimmableLight' do
  from REMM::ProductRequirement
  to   MetaDomoTarget::StandardFUnitInstance
      filter do |requirement|
        requirement.name.include?"Light" and
        requirement.parameterInstances.length != 0 and
        requirement.getEnumValue("ACTION") == "REGULATE"
      end
  mapping do |requirement, functionalUnit|
    # Guardamos en un array todos los artefactos que se van
    # creando a partir de la definición del requisito.
    # Al final crearemos una traza desde el requisito a
    # cada uno de los artefactos creados
    artifacts = Array.new

    # Obtenemos el número de luces que se van a regular
    numFU = requirement.getValue("X")

    # Obtenemos los mecanismos para regular las luces
    mecanismos = requirement.getEnumValues("MECHANISM")

    #-----
    # CREAR LOS PUNTOS DE LUZ REGULABLES Y SUS CONTROLADORES
    #-----
    # Guardamos en un array todos los controladores de las
    # luces que se tienen que conectar con los mecanismos
    luces = Array.new
    # La unidad funcional de la luz regulable tiene como
    # referencia FLD si es FLUORESCENT y LDM si es normal
    paramValue = requirement.getEnumValue("TYPE")
    if (paramValue == "FLUORESCENT")
      type = "FLD"
    else
      type = "LDM"
    end
    # Inicializamos la unidad funcional asociada al requisito
    functionalUnit.inicializar(type)

    # Creamos un artefacto que referencie a la unidad funcional
    # para las trazas forward en el modelo de traza
    artifacts << Util.createArtifact(functionalUnit)

    # Creamos el controlador del punto de luz
    DMO_FU = Util.createStandardFUnit("DMO")
    # Añadimos la unidad funcional al metamodelo y
    # a los artefactos
    functionalUnit.owner.hasFUnitInstance<<DMO_FU
    artifacts << Util.createArtifact(DMO_FU)
    # Conectamos el pto de luz al controlador
    canal2_Link=
      Util.createLink(DMO_FU,functionalUnit,"hwDimmer")
    functionalUnit.owner.hasFUnitLink<<canal2_Link
  end
end

```

Figura 47 Regla de transformación Requirement2DimmableLight (1/3).


```

if (mecanismos.include?("PUSH_BUTTON"))
  # Hay que poner un doble enlace porque el botón permite
  # encender/apagar y también regular
  canal3_Link =
    Util.createLink(DMO_FU, functionalUnit, "hwSwitch")
  functionalUnit.owner.hasFUnitLink<<canal3_Link
end
# Añadimos el controlador a las luces que se han creado
luces << DMO_FU
# Si hay que crear más de un punto de luz
if (numFU > 1)
  index = (numFU-1).to_i
  index.times do
# Creamos tantos puntos de luz como indique el parámetro -1
  luz_FU = Util.createStandardFUnit(type)
  functionalUnit.owner.hasFUnitInstance<<luz_FU
  artifacts << Util.createArtifact(luz_FU)
  DMO_FU = Util.createStandardFUnit("DMO")
  functionalUnit.owner.hasFUnitInstance<<DMO_FU
  artifacts << Util.createArtifact(DMO_FU)
  canal1_Link =
    Util.createLink(DMO_FU, luz_FU, "hwDimmer")
  functionalUnit.owner.hasFUnitLink<<canal1_Link
  if (mecanismos.include?("PUSH_BUTTON"))
    #hay que poner un doble enlace
    canal2_Link =
      Util.createLink(DMO_FU, luz_FU, "hwSwitch")
    functionalUnit.owner.hasFUnitLink<<canal2_Link
  end
  luces << DMO_FU
end
end #FIN DE CREACION DE PUNTOS DE LUZ Y SUS CONTROLADORES
# -----
#                      CREAR MECANISMOS Y SUS CONTROLADORES
# -----
dimmerInputControls = Array.new
switchInputControls = Array.new
for mecanismo in mecanismos
  mecanismoRef = Util.getFunctionalUnitRef(mecanismo)
  mecanismo_FU = Util.createStandardFUnit(mecanismoRef)
  functionalUnit.owner.hasFUnitInstance<<mecanismo_FU
  artifacts << Util.createArtifact(mecanismo_FU)
  if (mecanismo != "PRESENCE_DETECTOR")
    # Creamos la unidad funcional del controlador
    DMI_FU = Util.createStandardFUnit("DMI")
    functionalUnit.owner.hasFUnitInstance<<DMI_FU
    artifacts << Util.createArtifact(DMI_FU)
    # Conectamos el mecanismo y su controlador
    canal1_Link =
      Util.createLink(mecanismo_FU, DMI_FU, "hwSwitch")
    functionalUnit.owner.hasFUnitLink<<canal1_Link
    dimmerInputControls<<DMI_FU
  end

```

Figura 48 Regla de transformación Requirement2DimmableLight (2/3).

```

    if (mecanismo == "PUSH_BUTTON")
        switchInputControls<<DMI_FU
    end
else # El mecanismo es un detector de presencia
    SWI_FU = Util.createStandardFUnit("SWI")
    functionalUnit.owner.hasFUnitInstance<<SWI_FU
    artifacts << Util.createArtifact(SWI_FU);
    # Conectamos el mecanismo a su controlador
    canal1_Link =
        Util.createLink(mecanismo_FU, SWI_FU, "hwSwitch")
    functionalUnit.owner.hasFUnitLink<<canal1_Link
    switchInputControls<<SWI_FU
    # Hay que crear el temporizador
    TM_FU = Util.createStandardFUnit("TM")
    if (TM_FU.hasParameters)
        #Borramos los parámetros creados por defecto
        TM_FU.hasParamInstan.clear
        # Los parámetros se especifican en los req. hijos
        TM_FU.hasParamInstan = requirement.children
    end
    functionalUnit.owner.hasFUnitInstance<<TM_FU
    artifacts << Util.createArtifact(TM_FU)
    link1 = Util.createLink(SWI_FU, TM_FU, "switch")
    functionalUnit.owner.hasFUnitLink<<link1
    switchInputControls<<TM_FU
end
end

# -----
# CONECTAR EL CTRL DEL MECANISMO - CTRL DE CADA PUNTO DE LUZ
# -----
for inputControl in switchInputControls
    for luzControl in luces
        # Enlace tipo "switch" para encender y apagar
        accionLink =
            Util.createLink(inputControl, luzControl, "switch")
        functionalUnit.owner.hasFUnitLink<<accionLink
    end
end
for inputControl2 in dimmerInputControls
    for luzControl2 in luces
        # Enlace tipo "dimmer" para regular la intensidad
        accionLink =
            Util.createLink(inputControl2, luzControl2, "dimmer")
        functionalUnit.owner.hasFUnitLink<<accionLink
    end
end
# Creamos las trazas forward
for artifact in artifacts
    trace = Util.createTrace(requirement, artifact)
    matrix.traces << trace
    matrix.artifacts << artifact
end
end #end_mapping
end #end_rule

```

Figura 49 Regla de transformación Requirement2DimmableLight (3/3).

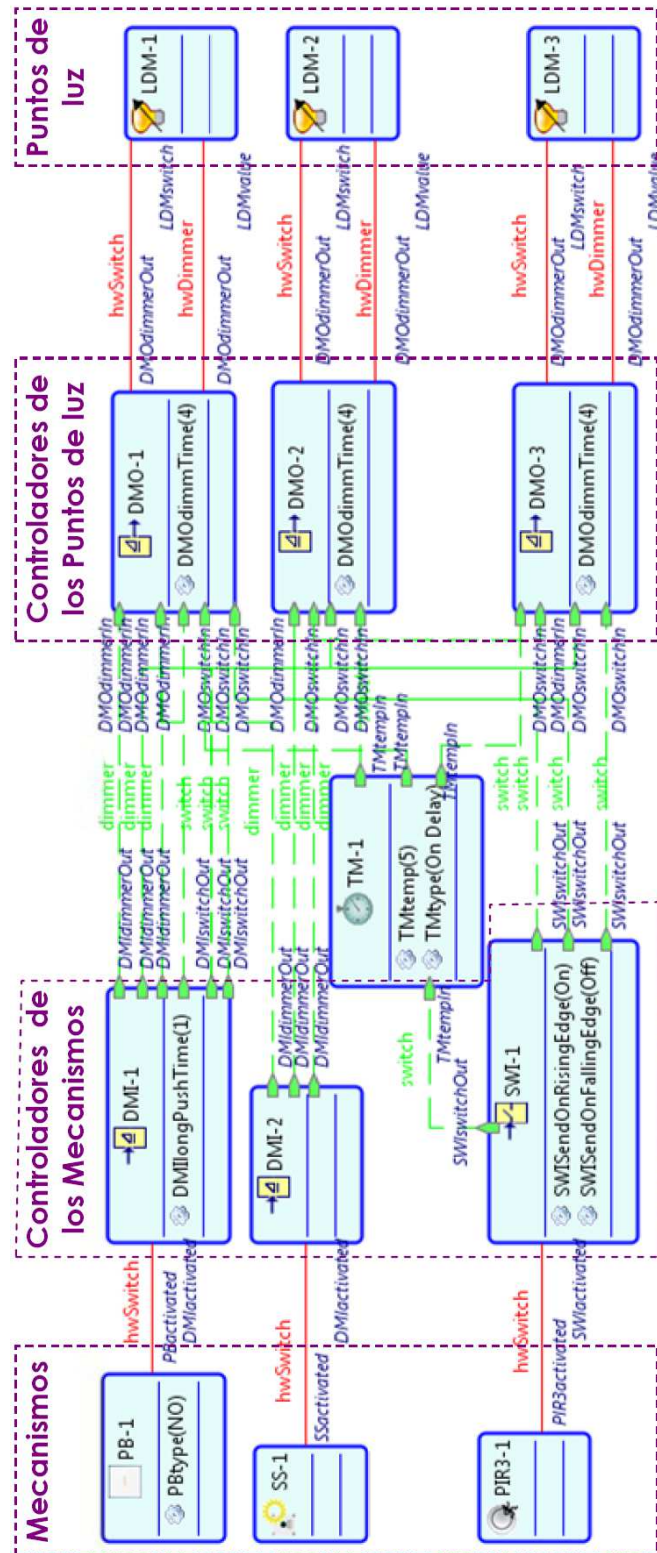


Figura 50 Modelo de la aplicación domótica que se corresponde con el requisito de iluminación: El sistema domótico dispondrá de un botón de pulsación, un detector de presencia y un sensor de iluminación para regular tres puntos de luz.

El mismo proceso de definición de requisitos parametrizados e implementación de la regla de transformación, se ha seguido para la definición del catálogo de requisitos reutilizables en el dominio de los sistemas domóticos. Este catálogo, que hemos denominado HAREL (*Home Automation Requirements Specification Language*), incluye: los requisitos relacionados con el aire acondicionado (splits o fancoil), las persianas (venecianas o tradicionales), toldos y sistemas de alarma (intrusión, humo, fuego y agua), además de los requisitos de iluminación (luces fijas o regulables). En consecuencia, la definición de transformación (*REMM2Habitation*) incluye, de forma análoga a la aquí presentada (*requirement2DimmableLight*, Figura 47 y siguientes), las reglas de transformación de cada uno de los requisitos en los modelos domóticos que los representan: *requirement2AirConditioning*, *requirement2VenetianBlind*, *requirement2Alarm*, *requirement2SunBlind*, *requirement2Shutter*, *requirement2FixedLight*. La definición de transformación *REMM2Habitation*, también especifica la regla que lanza la ejecución del resto de reglas de transformación (*top rule*, Figura 51). Esta regla *top* establece que cada requisito de producto funcional especificado en el modelo de requisitos de la aplicación, esto es, los requisitos reutilizados a partir del catálogo de requisitos reutilizables HAREL, se transformará en las correspondientes unidades funcionales de una aplicación domótica. Nótese que la correspondencia es uno a muchos dado que, como hemos visto para el sistema de iluminación, a partir de la especificación de un requisito se crean un conjunto de unidades funcionales y enlaces que las relacionan.

```

top_rule 'catalog2application' do
  from REMM::ProductCatalog
  to   MetaDomoTarget::DomoApplication
  mapping do |catalog, application|
    application.hasFUnitInstance = catalog.functionalReq
  end
end

```

Figura 51 RubyTL top rule.

La especificación completa de los requisitos, junto con los modelos domóticos que se generan automáticamente a partir de cada uno de ellos, se detalla en el Capítulo 8. Es importante destacar que, todos los posibles modelos que se pueden generar en función de las distintas instanciaciones de los parámetros de los requisitos, han sido validados por los expertos en el dominio, en el grupo de investigación del DSIE de la UPCT, y cumplen los criterios de calidad establecidos en la Sección 6.2.3. Los modelos generados son sintáctica y semánticamente *correctos*, puesto que son conformes al metamodelo y se han construido utilizando las unidades funcionales y los enlaces más adecuados (basándonos en el conocimiento de los expertos). Además, los modelos obtenidos son *consistentes* con los requisitos seleccionados y cumplen con la funcionalidad requerida (*completos*), son *comprensibles* para los expertos en el dominio de los sistemas domóticos que desarrollarán este tipo de aplicaciones y no contienen información que no es necesaria (*restringidos*).

6.3.1.2. Generación del modelo de traza

Cuando se ejecuta la transformación de modelos *REMM2Habitation* (tarea *Transformación de Modelos* del proceso de ingeniería de la aplicación de ReToCo, Figura 43 y Sección 6.2.2.2), al mismo tiempo que se genera el modelo de la aplicación, se crea el modelo de traza que liga los requisitos con los conceptos del DSL que se han ido creando. El modelo de traza que se obtiene, es conforme al metamodelo TraceMM (*Forward Trace MetaModel*) que se muestra en la Figura 52. Este metamodelo se ha definido de manera genérica, lo que significa que se puede utilizar para enlazar cualquier requisito con cualquier artefacto del desarrollo de análisis o diseño, en concreto, con los conceptos del DSL del dominio objeto de estudio. En nuestro caso de estudio, las reglas de transformación que se han definido, conectarán los requisitos de producto con los conceptos de los sistemas domóticos definidos en el DSL Habitation. Por ejemplo, el requisito de iluminación instanciado en la sección anterior (Sección 6.3.1.1) para especificar que la aplicación contará con 3 puntos de luz regulables mediante un botón de pulsación, un detector de presencia y un sensor de iluminación, se trazaría a las unidades funcionales que se han creado (Figura 50): PB-1, PIR3-1, SS-1, LDM-1, LDM-2, LDM-3 y los controladores.

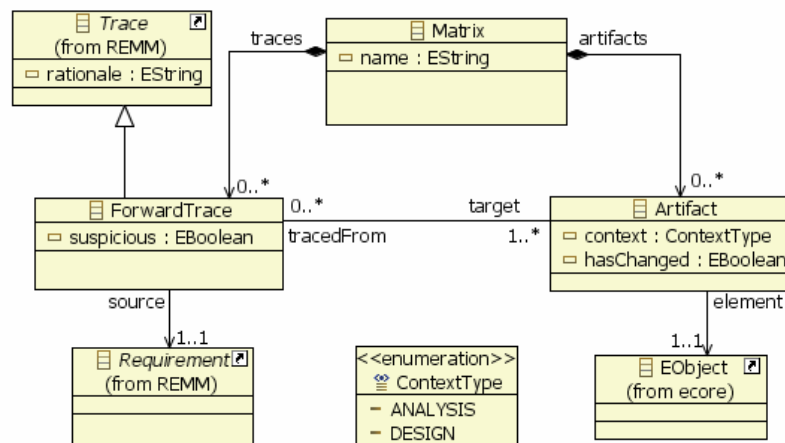


Figura 52 Metamodelo para representar las trazas hacia delante (TraceMM).

Si repasamos el código de la regla de transformación *Requirement2DimmableLight* que se presenta en la Figura 47, Figura 48 y Figura 49, podemos encontrar el detalle de la construcción del modelo de traza de esa parte de la definición de transformación *REMM2Habitation* que se extrae en la Figura 53. El mismo código se repetirá en todas las reglas de transformación para registrar las relaciones de traza entre los requisitos y las unidades funcionales creadas.

Para representar el modelo de requisitos de forma “amigable” se ha definido una transformación *m2t* que genera la matriz de trazabilidad en formato HTML (Sección 6.3.2.2, Figura 60). La generación de la matriz de trazabilidad es una de las tareas del proceso de ingeniería de la aplicación de ReToCo (Figura 43, Sección 6.2.2.5).

6.3.1.3. Generación del modelo de requisitos con trazas hacia delante

De acuerdo con la tarea presentada en la Sección 6.2.2.3, *Fusión de modelos*, del proceso de ingeniería de la aplicación de ReToCo (Figura 43), se ha definido una transformación m2m (*trace2remm* en la Figura 54), que se ejecutará una vez que se ha obtenido el modelo de traza, con el objeto de unir en un único modelo el modelo de requisitos de la aplicación y las trazas hacia delante generadas.

```
transformation 'remm2habitation'

# modelo de traza que se va a crear durante la transformación
matrix = TraceMM::Matrix.new
rule 'Requirement2DimmableLight' do
  from REMM::ProductRequirement
  to   MetaDomoTarget::StandardFUnitInstance
  filter do |requirement|
    ...
  end
  mapping do |requirement, functionalUnit|
    # Guardamos en un array todos los artefactos que se van
    # creando a partir de la definición del requisito.
    # Al final crearemos una traza desde el requisito a cada
    # uno de los artefactos creados
    artifacts = Array.new
    ...
    artifacts << Util.createArtifact(functionalUnit)
    ...
    # Para cada unidad funcional que tengamos que crear
    # se añade a la colección de artefactos
    # Por ejemplo, creamos el controlador del punto de luz
    DMO_FU = Util.createStandardFUnit("DMO")
    artifacts << Util.createArtifact(DMO_FU)
    ...
    # Cuando hayamos creado todas las unidades funcionales

    # Creamos las trazas forward que ligen el requisito
    # con los artefactos que se han creado
    # para añadirlas a la matriz de trazabilidad
    for artifact in artifacts
      trace = Util.createTrace(requirement, artifact)
      # Añadimos al modelo la traza y el artefacto
      matrix.traces << trace
      matrix.artifacts << artifact
    end
  end #end_mapping
end #end_rule
...
```

Figura 53 Creación del modelo de traza en la definición de transformación REMM2Habitation.

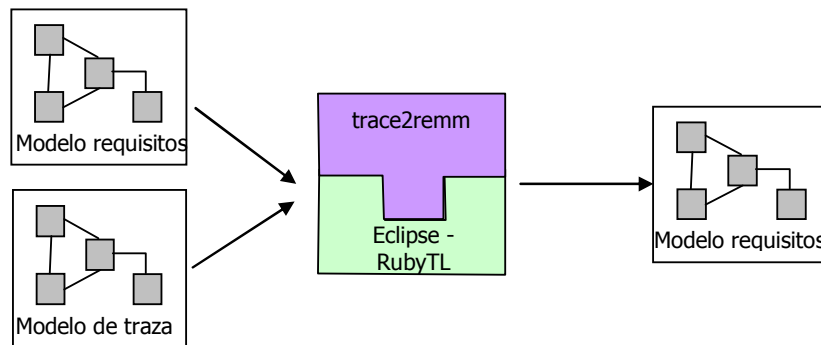


Figura 54 Definición de transformación para la fusión de modelos.

En la Figura 55 y la Figura 56 se muestra un extracto de la definición de transformación *trace2remm*, implementada para la fusión de modelos. La definición de las reglas de transformación es sencilla, puesto que se trata de crear los mismos elementos que se encuentran en el modelo de requisitos origen, en el modelo de requisitos destino, por tanto, hay una equivalencia 1 a 1 entre los conceptos, dado que los dos modelos son conformes al mismo metamodelo, REMM (por ejemplo, *requirement2requirement* en Figura 56). Además, el metamodelo de traza TraceMM (Figura 52) es un subconjunto del metamodelo REMM (Capítulo 5) y, en consecuencia, también existe una correspondencia 1 a 1 entre los conceptos del metamodelo de traza y los definidos en REMM (por ejemplo, *artifact2artifact* y *ForwardTrace2ForwardTrace* en Figura 56).

```
transformation 'trace2remm'

sourceCatalog = REMMSource::ProductCatalog.all_objects.first

top_rule 'matrix2catalog' do
  from TraceMM::Matrix
  to REMMTarget::ProductCatalog
  mapping do |matrix, targetCatalog|
    targetCatalog.name = sourceCatalog.name
    targetCatalog.purpose = sourceCatalog.purpose
    targetCatalog.scope = sourceCatalog.scope
    targetCatalog.overview = sourceCatalog.overview
    targetCatalog.description = sourceCatalog.description
    targetCatalog.productReq = sourceCatalog.requirements
    targetCatalog.intertraces = sourceCatalog.intertraces
    # dataTypes, stakeholders, testsCases, externalObjects,
    # glossary
    #Se añade la información del metamodelo de traza
    targetCatalog.artifacts = matrix.artifacts
    targetCatalog.forwardtraces = matrix.traces
  end
end
```

Figura 55 Extracto de la definición de transformación *trace2remm* (1/2).


```

rule 'artifact2artifact' do
  from TraceMM::Artifact
  to   REMMTarget::Artifact
  mapping do |source, target|
    target.context = source.context
    target.hasChanged = source.hasChanged
    target.set 'element', source.element
    target.tracedFrom = source.tracedFrom
  end
end
rule 'ForwardTrace2ForwardTrace' do
  from TraceMM::ForwardTrace
  to   REMMTarget::ForwardTrace
  mapping do |tsource, tttarget|
    tttarget.suspicious = tsource.suspicious
    tttarget.rationale = tsource.rationale
    tttarget.source = tsource.source
    tttarget.target = tsource.target
  end
end
rule 'requirement2requirement' do
  from REMMSource::ProductRequirement
  to   REMMTarget::ProductRequirement
  mapping do |source, target|
    target.id = source.get 'id'
    target.name = source.name
    target.text = source.text
    target.abstractionLevel = source.abstractionLevel
    target.priority = source.priority
    target.rationale = source.rationale
    target.type = source.type
    target.set 'reusedFrom', source.reusedFrom
    target.status = source.status
    target.isSourceOf = source.isSourceOf
    target.isTargetOf = target.isTargetOf
    # igual complementedBy, proposedBy, source, uses, validatedBy
    target.treacedTo = #establecemos la traza forward
      REMMTarget::ForwardTrace.all_objects.find{
        |trace| trace.source == source}
    if ! source.parameterInstances.nil?
      target.parameterInstances = source.parameterInstances
    end
  end
end
rule 'param2param' do
  from REMMSource::ParameterInstance
  to   REMMTarget::ParameterInstance
  mapping do |source, target|
    target.set 'instanceOfParameter',
      source.instanceOfParameter
    target.parameterValue = source.parameterValue
  end
end

```

Figura 56 Extracto de la definición de transformación trace2remm (2/2).

6.3.2. Transformaciones modelo a texto a partir de los modelos de requisitos

En el DSDM la definición de transformaciones modelo a texto (m2t) se utilizan para la generación del código de la aplicación en algún lenguaje de programación (como Java), o para generar la documentación. En el caso del proceso ReToCo, se hace uso de las transformaciones m2t para generar la especificación textual de requisitos, representada en el modelo de requisitos (en el marco del subproceso SIREN_{MD}), y para generar la representación textual del modelo de traza, esto es, la matriz de trazabilidad que se genera a modo de informe. Estas dos transformaciones se presentan en las Secciones 6.3.2.1 y 6.3.2.2, respectivamente.

6.3.2.1. Generación del documento de requisitos

Dado que uno de los principales entregables del proceso de IR es el documento con la especificación textual de requisitos, se ha definido una transformación m2t para generar dicha documentación de forma automática a partir de los modelos de requisitos (tarea T6 en el subproceso SIREN_{MD}, Sección 4.7). Para la definición de la transformación se ha utilizado el lenguaje MOFScript [7], que actualmente es uno de los lenguajes candidatos en el proceso de RFP (*Request For Proposal*) del OMG para la definición de un lenguaje estándar para la transformación de modelos MOF a texto. MOFScript cubre todos los aspectos necesarios en el contexto de la generación de texto en Ingeniería del Software, incluyendo: generación de texto a partir de modelos basados en MOF, definición de mecanismos de control (como bucles y estructuras condicionales), manipulación de valores de tipo cadena de texto, obtener expresiones que referencian a los elementos del modelo, producir ficheros de salida, trazabilidad entre los modelos y el texto generado y facilidad de uso.

A partir del modelo de requisitos, en realidad, se genera un conjunto de documentos HTML que contienen la especificación de requisitos. De esta forma, se puede navegar fácilmente por la especificación de requisitos y se facilita la validación de la misma (tarea T7 del subproceso SIREN_{MD}). La generación de documentos es diferente en función del modelo de requisitos origen que se esté transformando. De acuerdo a las facetas de reutilización propuestas en SIREN, y que soporta el metamodelo REMM, estos modelos de requisitos pueden ser: 1) modelo de requisitos *para* reutilización, en este caso la especificación de requisitos contiene un repositorio (o colección) de catálogos de requisitos reutilizables (pertenecientes a diferentes perfiles o dominios), o bien, 2) modelo de requisitos *con* reutilización, en este caso la especificación de requisitos representa un catálogo de producto, esto es, los requisitos de un producto o aplicación concreta. En las siguientes subsecciones se detallan los documentos generados para cada uno de estos modelos de requisitos.

Documentación generada para los modelos de requisitos para reutilización

En la Figura 57 se muestra gráficamente el conjunto de documentos generados a partir de un modelo de requisitos *para* reutilización y las relaciones de navegación que existen entre ellos. Vemos como, a partir del fichero con el contenido del repositorio, se podría navegar por cada uno de los catálogos de requisitos reutilizables que contenga. Desde todos los ficheros generados se puede volver tanto al fichero que contiene el

índice del catálogo (líneas rojas en la Figura 57) como al índice del repositorio. Esta relación de navegación no se muestra en el diagrama para favorecer la legibilidad. El contenido de cada fichero es el siguiente:

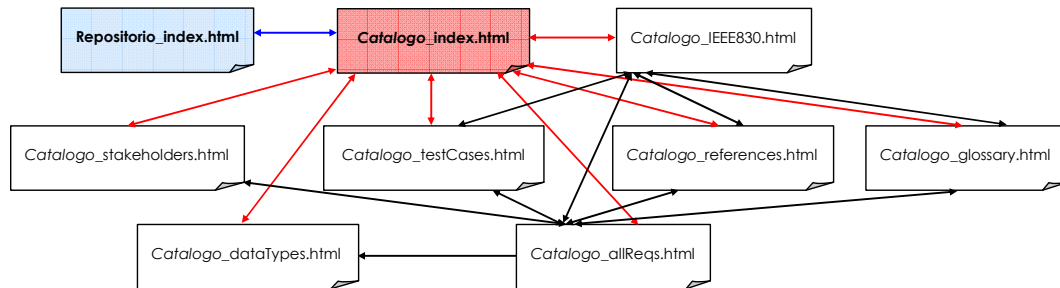


Figura 57 Transformación m2t a partir de modelos de requisitos *para* reutilización.

- `Repositorio_index.html`: contiene el listado con los nombres de los catálogos de requisitos reutilizables contenidos en un repositorio de requisitos (véase el ejemplo que se muestra en la Figura 59). Cada nombre de catálogo es un enlace al fichero desde el que se puede navegar por el contenido del catálogo. En consecuencia, la estructura de ficheros que sigue se repetirá por cada catálogo que incluya.
- `Catalogo_index.html`: contiene el índice de las distintas vistas del catálogo, esto es, el enlace a los ficheros que vamos a explicar a continuación. La palabra *Catalogo* aparece en cursiva porque será sustituida por el nombre del catálogo de requisitos en el momento de la transformación tanto en este fichero como en los que siguen.
- `Catalogo_allReqs.html`: contiene el listado de todos los requisitos definidos en el catálogo. Para cada requisito se muestra el detalle de la información asociada, siempre que se haya establecido, esto es, los valores de sus atributos (tipo, prioridad, etc.) y los elementos con los que está relacionado (parámetros, stakeholder, referencias, etc.). Los valores de los elementos con los que se relaciona siempre son enlaces al fichero donde se describe el detalle de dicho elemento. Por ejemplo, si un requisito utiliza un término del glosario, dicho término aparece como un enlace al fichero donde se encuentra la definición de este término. En el caso de los requisitos parametrizados se especifica el nombre del parámetro y el nombre del tipo. Este nombre será un enlace al fichero donde se muestra el detalle sobre el tipo de datos (véase el ejemplo que se muestra en la Figura 59).
- `Catalogo_dataTypes.html`: contiene el listado de todos los tipos de datos utilizados en el catálogo (el concepto *catálogo* se describe en la Sección 5.4). Por defecto, sólo contendrá los tipos de datos `NumberType` y `StringType`, que siempre están disponibles para cualquier catálogo. En el caso de que el catálogo contenga requisitos parametrizados cuyos parámetros sean de algún tipo de dato enumerado, la descripción de cada uno de estos tipos tendrá el siguiente formato: `nombreTipoEnumerado[min, max] = {Literal1[*], Literal2[*], ...}`. El `*` es opcional. Si un literal lo lleva significa que es obligatorio (Figura 59).

- *Catalogo_stakeholders.html*: contiene el listado de los stakeholders que han participado en la especificación de requisitos (el concepto *stakeholder* se describe en la Sección 5.3.1). Para cada uno de ellos se muestra sus atributos (nombre y posición en la empresa) y se lista el conjunto de requisitos que ha propuesto. El listado de requisitos muestra sólo el identificador y el nombre, pero es un enlace a la descripción completa del requisito en el fichero *Catalogo_allReqs.html* (Figura 59).
- *Catalogo_references.html*: contiene el listado de todas las referencias utilizadas, que amplían la especificación de requisitos (el concepto *objeto externo* se describe en la Sección 5.3.1). Para cada una de ellas muestra sus atributos y lista el conjunto de requisitos que la cita. El listado de requisitos muestra sólo el identificador y el nombre, pero es un enlace a la descripción completa del requisito en el fichero *Catalogo_allReqs.html*.
- *Catalogo_glossary.html*: contiene el glosario de términos utilizados en el catálogo, esto es, el conjunto de términos específicos junto con sus definiciones y sinónimos (el concepto *glosario* se describe en la Sección 5.3.1). Para cada término se lista el conjunto de requisitos que lo usa. El listado de requisitos muestra sólo el identificador y el nombre, pero es un enlace a la descripción completa del requisito en el fichero *Catalogo_allReqs.html*. Nótese cómo esta estructura permite validar si los requisitos han utilizado el mismo término de forma consistente.
- *Catalogo_testCases.html*: contiene la descripción de los casos de prueba definidos para validar los requisitos (el concepto *caso de prueba* se describe en la Sección 5.3.1). Para cada caso de prueba se lista también los requisitos que se pretende validar con él. El listado de requisitos muestra sólo el identificador y el nombre, pero es un enlace a la descripción completa del requisito en el fichero *Catalogo_allReqs.html*.
- *Catalogo_IEEE830.html*: contiene la especificación de requisitos del software (SRS, *Software Requirements Specification*) en el formato estándar IEEE-830 [131] (conforme a las guías de SIREN, sección 4.3.5). Para poder llevar a cabo esta transformación m2t se ha establecido una correspondencia entre los conceptos incluidos en REMM y las secciones propuestas por la plantilla del estándar (Tabla 41). Hay dos consideraciones que es necesario tener en cuenta en la definición de esta correspondencia. En primer lugar, de acuerdo con las recomendaciones del estándar IEEE-830, el objetivo de la plantilla del SRS es proporcionar “la especificación de un producto software, programa o conjunto de programas concretos que realizan ciertas funciones en un entorno específico”. Por tanto, dado que el objetivo de un modelo de requisitos *para* reutilización es la representación de un repositorio de catálogos de requisitos reutilizables, estos catálogos no incluyen la especificación de requisitos de un producto concreto sino de una familia de productos, por tanto, para este tipo de catálogos, se omite las secciones de alcance, descripción y perspectiva del producto, que se especificarán para cada producto concreto (apartados destacados en rojo en la Tabla 41). En segundo lugar, algunas de las características soportadas por REMM no tienen una correspondencia clara con ninguna de las secciones propuestas en el estándar, como por ejemplo las reglas

del negocio. Hemos considerado interesante incluir nuevas secciones en la plantilla para recoger esta información, de acuerdo con las recomendaciones del propio estándar, añadiéndolas al final de la sección 3 del documento (secciones destacadas en azul en la Tabla 41).

En la Figura 59 se muestra un ejemplo sencillo de algunos de los ficheros generados para un modelo de requisitos *para* reutilización. El modelo representa un repositorio con dos catálogos de requisitos reutilizables: seguridad y TPV (*Terminal de Punto de Venta*). El catálogo de requisitos de seguridad muestra dos requisitos parametrizados, propuestos por Judith, que es la responsable de seguridad, en relación con las copias de seguridad.

Documentación generada para los modelos de requisitos con reutilización

En el caso de la transformación m2t a partir de modelos de requisitos *con* reutilización (Figura 58), se genera un único fichero *Catalogo_index.html*, a partir del cual se puede navegar por el contenido de la especificación de requisitos del producto o la aplicación (requisitos, stakeholders, referencias, glosario de términos y casos de prueba). En este caso, no es aplicable la generación del fichero con los tipos de datos puesto que, en una especificación de un nuevo producto, los requisitos parametrizados que se hayan reutilizado se habrán instanciado con los valores adecuados para el proyecto actual. La estructura y contenido de los ficheros generados es la misma que se ha explicado para los modelos de requisitos creados en la faceta *para* reutilización, salvo en el caso del documento de especificación conforme al estándar IEEE-830, donde ahora sí se puede especificar la descripción, alcance y perspectiva general del producto que se está especificando (secciones de la plantilla destacadas en rojo en la Tabla 41).

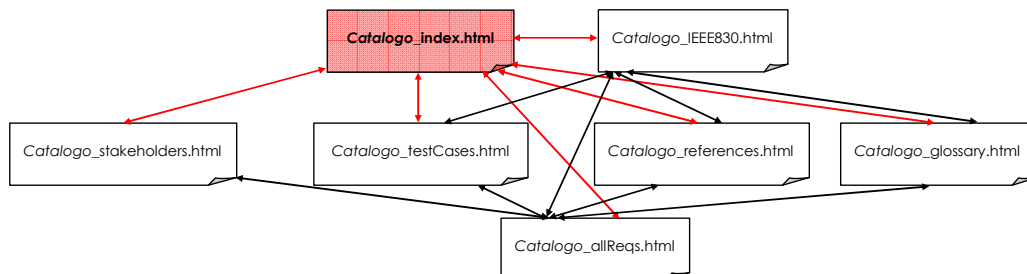


Figura 58 Transformación m2t a partir de modelos de requisitos *con* reutilización.

Por último, cabe destacar que, aunque en nuestra propuesta hemos mostrado cómo generar el SRS conforme a la plantilla del estándar IEEE-830, de forma similar se pueden definir diferentes transformaciones m2t para obtener, por ejemplo: 1) el documento de visión que contenga los requisitos del negocio y las características del sistema, 2) el documento de requisitos del sistema de acuerdo a las recomendaciones del estándar IEEE-1233 [128], 3) un documento de requisitos del software siguiendo una plantilla diferente (como VOLERE [291]), o 4) el documento de requisitos en otros formatos como el estándar ReqIF (*Requirements Interchange Format*) [260] que permita la exportación de los requisitos a otras herramientas de gestión de requisitos. Para cada uno de estos casos habría que establecer una correspondencia entre los conceptos

definidos en REMM y las plantillas del documento de especificación de requisitos que se va generar, de forma análoga a lo que se ha definido en la Tabla 41.

Plantilla IEEE-830	Conceptos de REMM
1. Introduction	
1.1 Purpose	Catalog.purpose
1.2 Scope	<i>ProductCatalog.scope</i>
1.3 Definitions, acronyms, abbreviations	Glossary - Terms
1.4 References	ExternalObject
1.5 Overview	<i>ProductCatalog.overview</i>
2. Overall description	<i>ProductCatalog.description</i>
3. Specific requirements	
3.1 External interfaces	ReqType::FUNCTIONALITY_INTEROPERABILITY ReqType::FUNCTIONALITY_ACCURACY
3.2 Functions	ReqType::FUNCTIONAL
3.3 Performance requirements	ReqType::EFFICIENCY
3.4 Design constraints	ReqType::CONSTRAINT
3.4.1 Standard compliance	ReqType::FUNCTIONALITY_COMPLIANCE ReqType::RELIABILITY_COMPLIANCE ReqType::EFFICIENCY_COMPLIANCE ReqType::PORTABILITY_COMPLIANCE ReqType::USABILITY_COMPLIANCE ReqType::MAINTAINABILITY_COMPLIANCE
3.5 Software system attributes	
3.5.1 Reliability	ReqType::RELIABILITY
3.5.2 Availability	ReqType::RELIABILITY_AVAILABILITY
3.5.3 Security	ReqType::FUNCTIONALITY_SECURITY
3.5.4 Maintainability	ReqType::MAINTAINABILITY
3.5.5 Portability	ReqType::PORTABILITY
3.5.6 Usability	<i>ReqType::USABILITY</i>
3.6 Other requirements	ReqType::NON_FUNCTIONAL
3.6.1 Reglas del negocio	<i>AbstractionLevelType::BUSINESS_RULES</i>
3.7 Requirements interdependences	<i>InterReqTrace</i>

Tabla 41 Correspondencia entre la plantilla del IEEE-830 y REMM.

6.3.2.2. Generación de la matriz de traza

La definición de las transformaciones m2t para generar los informes de trazabilidad (tarea *Generar la matriz de trazabilidad* en el proceso de ingeniería de la aplicación, Figura 43, Sección 6.2.2.5) ha sido desarrollada por el grupo de investigación DSIE de la UPCT, en el marco de la colaboración para la aplicación del proceso ReToCo al dominio de los sistemas domóticos. Los informes generados para este caso de estudio suponen una adaptación de las plantillas definidas en un trabajo anterior [301], que permite al analista generar informes detallados a partir de los modelos de traza. La tecnología utilizada por el grupo DSIE para la implementación de estas transformaciones m2t, es el lenguaje basado en plantillas JET (*Java Emitter Template*) [5]. JET permite generar salidas de texto (HTML, Java, XML, etc.) a partir de modelos EMF (*Eclipse Modelling Framework*).

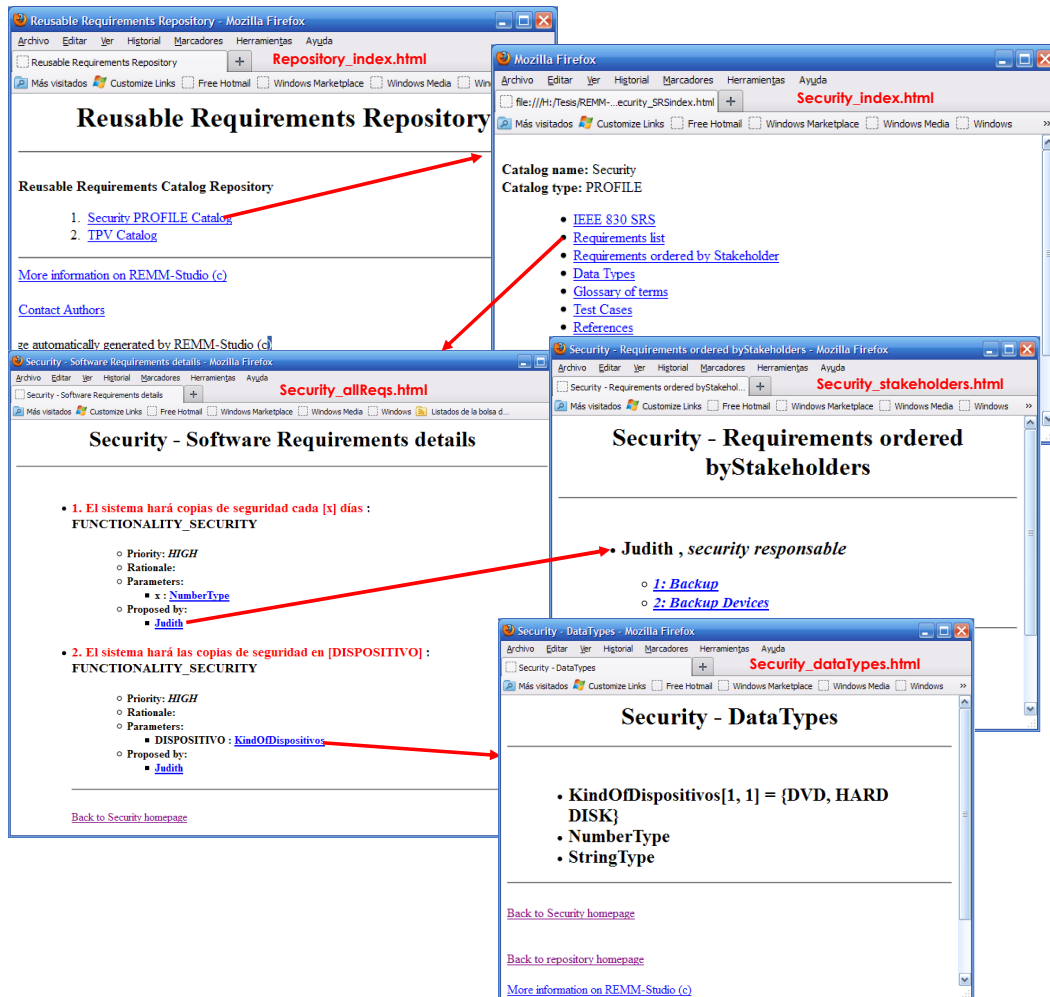


Figura 59 Ejemplo de ficheros generados para un modelo de requisitos *para* reutilización.

En el caso particular de los sistemas domóticos, el informe de traza generado permite analizar cómo se ha tenido en cuenta los requisitos en el proceso de generación automática de los modelos de la aplicación domótica, puesto que en el informe se representa la matriz de traza que incluye las relaciones entre cada requisito y los dispositivos domóticos generados en el modelo de la aplicación. En la Figura 60 se muestra una captura de pantalla de parte del fichero HTML generado a partir del modelo de traza que relaciona el requisito de iluminación con las unidades funcionales creadas en el modelo de la aplicación (LDM-1, DMO-1, LDM-2, etc.) de la Figura 50.

Source Link ▼ ▲		Target Link ▼ ▲	
Metamodel	Model Element	Metamodel	Model Element
REMM	<u>ProductRequirement</u> - text => The HA system shall provide a {PUSH_BUTTON, PRESENCE_DETECTOR, SUNRISE/SUNSET_SENSOR} to REGULATE 3.0 light points - id => 2 - name => Lighting Management	MetaDomo	<u>StandardFUnitInstance</u> - FUInstanceName => Light Dimmable LDM-1 - FUInstanceDes => Light Dimmable - FUInstanceRef => LDM-1
REMM	<u>ProductRequirement</u> - text => The HA system shall provide a {PUSH_BUTTON, PRESENCE_DETECTOR, SUNRISE/SUNSET_SENSOR} to REGULATE 3.0 light points - id => 2 - name => Lighting Management	MetaDomo	<u>StandardFUnitInstance</u> - FUInstanceName => Dimming Out DMO-1 - FUInstanceDes => Dimming Out - FUInstanceRef => DMO-1
REMM	<u>ProductRequirement</u> - text => The HA system shall provide a {PUSH_BUTTON, PRESENCE_DETECTOR, SUNRISE/SUNSET_SENSOR} to REGULATE 3.0 light points - id => 2 - name => Lighting Management	MetaDomo	<u>StandardFUnitInstance</u> - FUInstanceName => Light Dimmable LDM-2 - FUInstanceDes => Light Dimmable - FUInstanceRef => LDM-2
REMM	<u>ProductRequirement</u> - text => The HA system shall provide a {PUSH_BUTTON, PRESENCE_DETECTOR, SUNRISE/SUNSET_SENSOR} to REGULATE 3.0 light points - id => 2 - name => Lighting Management	MetaDomo	<u>StandardFUnitInstance</u> - FUInstanceName => Dimming Out DMO-2 - FUInstanceDes => Dimming Out - FUInstanceRef => DMO-2
REMM	<u>ProductRequirement</u> - text => The HA system shall provide a {PUSH_BUTTON, PRESENCE_DETECTOR, SUNRISE/SUNSET_SENSOR} to REGULATE 3.0 light points - id => 2 - name => Lighting Management	MetaDomo	<u>StandardFUnitInstance</u> - FUInstanceName => Light Dimmable LDM-3 - FUInstanceDes => Light Dimmable - FUInstanceRef => LDM-3
REMM	<u>ProductRequirement</u> - text => The HA system shall provide a {PUSH_BUTTON, PRESENCE_DETECTOR, SUNRISE/SUNSET_SENSOR} to REGULATE 3.0 light points - id => 2 - name => Lighting Management	MetaDomo	<u>StandardFUnitInstance</u> - FUInstanceName => Dimming Out DMO-3 - FUInstanceDes => Dimming Out - FUInstanceRef => DMO-3

Figura 60 Informe de trazabilidad generado automáticamente mediante una transformación m2t.

6.4. Conclusiones y aportaciones al estado del arte

En este capítulo hemos presentado un proceso completo de DSDM, desde los requisitos hasta el código, basado en la reutilización de requisitos, que hemos denominado ReToCo. Como cualquier proceso de DSDM, la propuesta gira en torno a un metamodelo, REMM en este caso, para la especificación de modelos de requisitos (Capítulo 5), y las transformaciones de modelos para el desarrollo del software. Las transformaciones soportadas por el proceso ReToCo, incluye: 1) transformaciones *modelo a modelo*, para generar el modelo de la aplicación conforme a un DSL y el modelo de traza que enlaza los requisitos con los conceptos creados del DSL; y 2) transformaciones *modelo a texto*, para generar los documentos de requisitos, informes de trazabilidad y el código de la aplicación.

El resumen de nuestra propuesta, utilizando el mismo marco de comparación que se definió en la revisión sistemática sobre transformaciones a partir de modelos de requisitos (Sección 3.3, Tabla 12), se puede ver en la Tabla 42. El objetivo de esta revisión

era estudiar el estado de arte en cuanto a las propuestas de transformación automáticas (o semiautomáticas) de modelos de requisitos en otros modelos del desarrollo, analizando si estas transformaciones están definidas en el marco de un proceso de DSDM global y si estas transformaciones tenían en cuenta o no las especificaciones de requisitos textuales. A la vista de los resultados obtenidos en la revisión, podemos destacar que, hasta donde nosotros sabemos, la propuesta presentada en esta tesis doctoral contribuye al estado del arte en los siguientes aspectos:

- Se trata del único enfoque propuesto en el marco del DSDM que conocemos que da soporte a los dos aspectos implicados en la reutilización de requisitos, modelado *para* y *con* reutilización.
- Se trata de la única propuesta que conocemos que propone un proceso completo (desde los requisitos hasta el código), de propósito general, que parte de modelos de requisitos, que representan la especificación de requisitos textuales, donde se establecen explícitamente las relaciones entre ellos. Las relaciones de traza se tienen en cuenta durante la reutilización sistemática de requisitos. El proceso completo está soportado por la herramienta REMM-Studio⁺, que se presenta en el capítulo siguiente, y se ha instanciado para el caso concreto de los sistemas domóticos. La instanciación del proceso a este dominio ha sido evaluada por los desarrolladores de este tipo de aplicaciones (expertos del dominio) como detallaremos en el Capítulo 8.

MDA	Requisitos textuales	Modelo origen	Modelo destino	Tipo	Lenguaje	Herramienta	Validación	Dominio
No	Si	Modelo de requisitos	Modelo de aplicación	A	RubyTL	REMM-Studio	Sistemas domóticos	Propósito general
			Modelo de traza	A	RubyTL			
			SRS	A	MOFScript			
			Informe traza	A	JET			

Tabla 42 Resumen de transformaciones de modelos a partir de modelos de requisitos.

Capítulo 7

REMM-Studio⁺: un entorno integrado de Ingeniería de Requisitos dirigido por modelos y basado en reutilización

7.1. Introducción

De acuerdo con Pressman [279], las *herramientas CASE* (*Computer-Aided Software Engineering*) ayudan a los profesionales y directivos del área de la Ingeniería del Software en las actividades relacionadas con el proceso software. El uso de herramientas, que reducen la cantidad de esfuerzo necesario para producir un entregable o llevar a cabo algún hito del proyecto, reporta beneficios sustanciales. Además, lo que es aún más importante es que las herramientas pueden proporcionar a los ingenieros diferentes puntos de vista de la información, lo que mejora su comprensión sobre el trabajo que realiza. Este hecho conduce a mejoras en la toma de decisiones y en la calidad del software.

El objetivo de la presente tesis doctoral es la integración de los requisitos en el *Desarrollo de Software Dirigido por Modelos* (DSDM), para lo que se ha propuesto un proceso completo de desarrollo (ReToCo, Capítulo 6). Por tanto, es necesario analizar las herramientas disponibles, tanto en el ámbito de la Ingeniería de Requisitos (IR) como en el ámbito del DSDM, para evaluar hasta qué punto nos pueden servir para dar soporte a dicho proceso. El eje central del proceso ReToCo es el subproceso de IR dirigido por modelos SIREN_{MD} (Sección 4.7) que requiere de: 1) un editor de modelos de requisitos que permita representar visualmente las especificaciones de requisitos tanto *para* como *con* reutilización y 2) una herramienta para ejecutar las transformaciones (modelo a modelo y modelo a texto) a partir de los modelos de requisitos creados, para generar el modelo inicial conforme a un DSL y la especificación textual de requisitos.

En consecuencia, antes de presentar el contenido y organización de este capítulo (Sección 7.1.3), debemos analizar, por un lado, el soporte que las herramientas de gestión de requisitos proporcionan para la reutilización y modelado de los requisitos (Sección 7.1.1), y, por otro lado, cómo se integran los requisitos en las herramientas que soportan el

DSDM (Sección 7.1.2). Las conclusiones que se extraigan del estudio servirán para justificar el interés de desarrollar una nueva herramienta como REMM-Studio⁺.

En principio, las herramientas comerciales dedicadas a la gestión de requisitos se centran en una representación textual de los requisitos y no proporcionan ninguna facilidad para el modelado. Hay que analizar si, con la nueva corriente del DSDM, ha cambiado esta tendencia. En cuanto a las herramientas de soporte al DSDM, la tendencia ha sido que se centren en los modelos y las transformaciones de modelos dejando los requisitos a un lado. A partir de las especificaciones de requisitos, los analistas generan, de forma manual y en base a su experiencia, el primer modelo del desarrollo, cuya definición y construcción sí que es soportada por las herramientas de modelado. A continuación, vamos a analizar si este conocimiento que tenemos a priori es o no cierto.

7.1.1. Herramientas de gestión de requisitos

Las *herramientas de gestión de requisitos* (RMT, *Requirements Management Tool*) almacenan los requisitos y los atributos asociados a cada uno de ellos en bases de datos multiusuarios (en esta sección seguimos a Wiegers [360] para describir las propiedades y beneficios de las RMT). Estas herramientas sirven de apoyo a un proceso de IR bien definido y proporcionan mucha flexibilidad en la gestión de cambios de los requisitos y en el uso de los requisitos como base para el diseño, las pruebas y la gestión de proyectos. Además, permiten importar requisitos, definir valores de los atributos, filtrar y mostrar el contenido de la base de datos, exportar los requisitos en varios formatos, definir relaciones de traza y conectar los requisitos con elementos almacenados en otras herramientas de desarrollo de software.

Los beneficios esperados del uso de las RMT son mayores conforme el proyecto avanza y el equipo va olvidando los detalles acerca de los requisitos. Las tareas en las que puede contribuir son: 1) gestión de versiones y cambios; 2) análisis del impacto de los cambios siguiendo las trazas entre diferentes tipos de requisitos, entre los requisitos y diferentes subsistemas y entre requisitos individuales y componentes del sistema relacionados (por ejemplo, diseños, módulos de código, test, documentación de usuario); 3) controlar el estado de los requisitos y, en consecuencia, el estado del proyecto (por ejemplo, el 17% de los requisitos no están implementados o el 55% de los requisitos han sido verificados); 5) control de accesos para usuarios o grupos de usuarios que podrían estar geográficamente dispersos (las bases de datos utilizan bloqueos a nivel de requisitos para permitir la actualización concurrente); 6) discutir con los stakeholders cuestiones relacionadas con los requisitos electrónicamente; y 7) reutilización de requisitos.

La reutilización de requisitos que se propone en el listado anterior, se refiere a la capacidad de almacenar el requisito una única vez en la base de datos para poder referenciarlo posteriormente, siempre que sea aplicable, en otros proyectos o partes de la descripción del producto, evitando de esta forma requisitos duplicados [360]. De los estudios realizados sobre el soporte a la reutilización de las herramientas RMT comerciales (Sección 4.6.1) se concluye que las RMT analizadas (IBM Rational RequisitePro [124], Borland CALIBER [36], IBM Rational DOORS [123]) no soportan los aspectos considerados como claves para la reutilización de requisitos, incluyendo:

definición de requisitos parametrizados, reutilización de relaciones de traza, o gestión de requisitos exclusivos. La capacidad de reutilización de las herramientas se limita a copiar y pegar los requisitos entre proyectos. En general, las RMT no soportan la reutilización sistemática de requisitos [208, 339], ni la especificación de la variabilidad en los requisitos [28, 32]. Para cubrir estas limitaciones, existen algunos enfoques que proponen extender las RMT comerciales como DOORS, con los mecanismos de variabilidad [306], o RequisitePro, con capacidades de reutilización, como es el caso de SirenTool (Sección 4.6).

En cuanto al soporte a la representación visual de las especificaciones de requisitos de estas mismas herramientas (RequisitePro, CALIBER y DOORS), cuando se ofrece, tiene que ver con el uso de modelos que complementan la especificación textual de requisitos, por ejemplo, casos de uso. Este es el caso de CALIBER y DOORS. Por un lado, CALIBER combina escenarios de negocio visuales, la definición de guiones (*storyboarding*) y la simulación interactiva a través de una interfaz navegable. Por otro lado, en DOORS se puede instalar un complemento (*DOORS Analyst*) que permite integrar en un mismo entorno los modelos (UML y SysML) y los requisitos. La representación visual de la especificación textual de los requisitos, al estilo de los modelos que se pueden definir conformes a REMM, no es soportada por ninguna de las RMT analizadas. No obstante, todas las herramientas soportan la integración con otras herramientas de desarrollo, lo que permite trazar los requisitos con otros modelos del desarrollo, aunque no sea en un entorno integrado.

Por último, cabría analizar el metamodelo de requisitos soportado por las RMT. En general, se trata de un metamodelo simple, puesto que normalmente sólo incluyen el concepto de requisito (que se caracteriza por un conjunto de atributos) y las relaciones de traza “sub-requisito” y “requisitos asociados” [304]. Hasta donde nosotros sabemos, ninguna de las RMT comerciales soporta un conjunto amplio de relaciones entre requisitos como el propuesto en REMM (Capítulo 5).

7.1.2. Herramientas de soporte al DSDM

Las herramientas CASE tradicionales que soportan la edición de modelos y su almacenamiento en un repositorio, han evolucionado para dar soporte al DSDM. En este tipo de procesos de desarrollo, no sólo son necesarios los modelos, sino que se requiere la definición y ejecución de transformaciones de modelos. De acuerdo con Kleppe *et al.* [170], los elementos que forman parte de un entorno completo que soporte el proceso MDA (*Model Driven Architecture*, la propuesta de DSDM del OMG), que se pueden generalizar para cualquier proceso de DSDM, son los que se muestran en la Figura 61, e incluyen:

- *Editor de modelos*: editor para poder construir y modificar los modelos.
- *Herramienta para validar los modelos*: se debe cuidar que los modelos que se utilizan para generar otros modelos estén extremadamente bien definidos. La herramienta de validación puede comprobar los modelos contra un conjunto de reglas predefinidas o definidas por el usuario.
- *Repositorio de modelos*: constituye la base de datos de los modelos. En el repositorio se almacenan y consultan los modelos utilizando, por ejemplo, XMI.

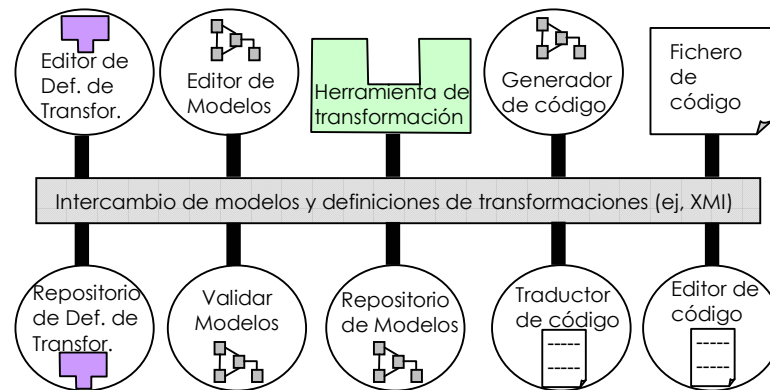


Figura 61 Funcionalidad de una herramienta que soporte el DSDM [170].

- *Editor de definiciones de transformaciones*: editor para poder construir y modificar las reglas de transformación.
- *Repositorio de definiciones de transformaciones*: para almacenar las reglas de transformación.
- *Editor de código*: funciones comunes a un entorno de desarrollo de software interactivo (IDE, *Interactive Development Environment*) como son la compilación, depuración y edición del código fuente.
- *Ficheros de código*: aunque el código se puede considerar que es un modelo, normalmente se mantienen como ficheros de texto. Dado que hay herramientas que no “entienden” los ficheros de texto se necesita:
 - o *Generador de código*: lee el código en el repositorio de modelos y genera un fichero de código en formato de texto.
 - o *Traductor de código*: que es capaz de leer ficheros de texto y almacenarlos en un repositorio de modelos, en un formato basado en modelos, que puedan utilizar otras herramientas.

Según Kleppe *et al.* [170], todas estas funciones pueden estar soportadas de dos formas: específica del lenguaje o genérica. Por ejemplo, una herramienta específica del lenguaje podría proporcionar únicamente un editor de modelos para UML y un generador de código de UML a C#. Mientras que, un editor de modelos genérico permitiría al usuario editar cualquier modelo, en tanto que esté disponible el lenguaje utilizado para la definición de los modelos. En consecuencia, en este tipo de herramientas se tiene que dar soporte a la definición de lenguajes.

Atendiendo al aspecto del DSDM al que dan soporte las herramientas, éstas se pueden clasificar fundamentalmente como sigue (Tabla 43):

- *Herramientas para el desarrollo de aplicaciones*: esto es, herramientas que soportan: 1) la edición de modelos, habitualmente UML y/o SysML, y 2) las transformaciones de modelos, fundamentalmente transformaciones modelo a código. Siguiendo el enfoque de DSDM, a partir de los modelos de análisis se puede llegar a generar el código de la aplicación mediante transformaciones de modelos. Además, estas herramientas proporcionan mecanismos para la extensión de lenguajes mediante la definición de perfiles UML. Es el caso de las

herramientas: *Artisan Studio*, *Innovator Modeling Platform*, *MagicDraw*, *Modelio*³, *ObjectiF* y *Rational Rhapsody*.

- Herramientas para el modelado específico de dominio (DSM, *Domain Specific Modeling*): que incluye las herramientas que soportan la definición de lenguajes en la forma de metamodelos (DSL, *Domain Specific Language*), y la definición de transformaciones a partir de los modelos conformes a este lenguaje. Los editores de modelos se pueden generar de forma automática a partir de la definición del metamodelo. Es el caso de las herramientas: *IBM Rational Software Architect*, *Microsoft Visual Studio Visualization and Modeling SDK*⁴, *MetaEdit*⁴ o los proyectos *Eclipse GMF/EMF/openArchitectureWare* (oAW).

Herramienta	Desarrollador	Página oficial
Desarrollo de aplicaciones		
<i>Artisan Studio</i>	Atego	http://www.atego.com/products/artisan-studio/
<i>Innovator Modeling Platform</i>	MID	http://www.mid.de/en/products
<i>MagicDraw</i>	No Magic	http://www.nomagic.com/
<i>Modelio</i>	ModelioSoft	http://www.modeliosoft.com/en/products/features.html
<i>ObjectiF</i>	MicroTOOL	http://www.microtool.de/objectif/en/
<i>Rational Rhapsody</i>	IBM	http://www-01.ibm.com/software/awdtools/rhapsody/
Modelado específico de dominio		
<i>Rational Software Architect</i>	IBM	http://www-01.ibm.com/software/rational/products/swarchitect/
<i>Visual Studio Visualization and Modeling SDK</i>	Microsoft	http://archive.msdn.microsoft.com/vsvmsdk
<i>MetaEdit</i>	MetaCase	http://www.metacase.com/
<i>GMF/EMF/</i>	Eclipse project	http://www.eclipse.org/modeling/
<i>openArchitectureWare</i>	Eclipse project	http://www.eclipse.org/workinggroups/oaw/
Mixtas		
<i>Enterprise Architect</i>	Sparx Systems	http://www.sparxsystems.com.au/
<i>Together Architect</i>	Borland	http://techpubs.borland.com/together/
Transformación de modelos		
<i>AndroMDA</i>	Open source	http://www.andromda.org/docs/index.html
<i>Actifsource</i>	Open source	http://www.actifsource.com/
<i>ATL, QVT</i>	Eclipse Project	http://www.eclipse.org/m2m/
<i>Acceleo, JET</i>	Eclipse Project	http://www.eclipse.org/modeling/m2t/
<i>MOFScript</i>	Eclipse Project	http://www.eclipse.org/gmt/mofscript/

Tabla 43 Herramientas para el DSDM.

- Herramientas de desarrollo *mixtas* se trata de las herramientas que soportan tanto el desarrollo de aplicaciones utilizando lenguajes estándar (como UML), como la definición de nuevos lenguajes, proporcionando capacidades para el

³ Sucesor de *Objecteering*, que fue la primera herramienta de soporte a MDA para UML mediante la técnica de perfiles (*profile*). *Objecteering* cubre el ciclo de modelado completo, incluyendo la gestión de requisitos, producción de documentación e informes y generación de código.

⁴ Es el sucesor de DSL SDK, también llamado DSL Tools.

metamodelado. Este es el caso de las herramientas: *Enterprise Architect* o *Together Architect*.

- Herramientas para *transformaciones*: ofrecen la capacidad de definir y ejecutar transformaciones de modelos a código (como AndroMDA y actifsource), modelo a modelo (como ATL, QVT o RubyTL [299]) o modelo a texto (como MOFScript, JET o Acceleo).

En la Tabla 43 hemos recogido sólo una pequeña muestra del amplio conjunto de herramientas disponibles, seleccionadas bien por su popularidad o bien por la amplia experiencia y reconocimiento de las empresas que las han desarrollado. Hay disponibles distintas listas, que amplían la de la Tabla 43, como son: la lista de herramientas de código abierto (*openSource*) que propone OpenSource-IT.com [261], el conjunto que ha recopilado Xavier Seignard [311] y el listado de productos que mantiene el OMG que son conformes a su propuesta MDA [252].

De acuerdo con la clasificación de herramientas propuesta en la Tabla 43, podríamos decir que utilizaríamos las herramientas de soporte al DSM y las de transformación de modelos, con el objeto de implementar un entorno para el desarrollo de aplicaciones siguiendo el enfoque de DSDM, esto es, para dar soporte a un proceso de DSDM. Así, nos interesa saber si las herramientas DSDM para el desarrollo de aplicaciones prestan atención a la especificación de requisitos textual, de manera que soporten un proceso de desarrollo completo, desde los requisitos hasta el código. Hemos analizado este tipo de herramientas (las que se muestran en la Tabla 43), para comprobar si soportan la representación visual de la especificación de requisitos textuales (modelos de requisitos) y si se sincronizan con alguna de las herramientas RMT que facilitan el proceso de IR.

Los resultados del estudio se muestran en la Tabla 44. En ella podemos ver que, prácticamente todas las herramientas, incluyen la gestión de requisitos textuales, lo que les permite trazar los requisitos con el resto de artefactos del desarrollo. La única excepción es *ObjectiF*, que modela los requisitos únicamente mediante la definición de casos de uso. Esto constituye un resultado interesante, que contradice la idea preconcebida a la que apuntábamos en la introducción, sobre la exclusión de los requisitos en las herramientas de DSDM.

Por otro lado, cinco de las ocho herramientas analizadas (63% aprox.) soportan la representación visual de los requisitos textuales, bien mediante el diagrama de requisitos de SysML o algún diagrama propio basado en éste. No obstante, las dos herramientas que no permiten la representación gráfica de los requisitos (MagicDraw y Together Architect) están sincronizadas con herramientas comerciales de gestión de requisitos, por lo que los requisitos textuales se pueden trazar a los elementos definidos en el resto de modelos del desarrollo. La conexión con las RMT comerciales la ofrecen también más de la mitad de las herramientas estudiadas. Por último, destacar que todas las herramientas ofrecen la posibilidad de generar la documentación a partir de los modelos, en diferentes formatos predefinidos o en algún formato configurable por el usuario.

Los resultados obtenidos son buenos desde el punto de vista de la IR en general y la trazabilidad de requisitos en particular. No obstante, los mismos problemas que ya hemos presentados respecto a los diagramas de requisitos SysML (Sección 1.1.3.2) y las herramientas RMT (Sección 7.1.1) son aplicables a las herramientas de DSDM, puesto que se apoyan en ellos para el proceso de IR. En resumen, los metamodelos de requisitos

soportados por las herramientas de DSDM son pobres y no soportan las cuestiones claves de la reutilización de requisitos.

Herramienta	Especificación textual de requisitos	Modelos de requisitos	Conexión con RMT	Genera Doc. a partir de los modelos
<i>Artisan Studio</i>	✓	Modelo de requisitos SysML	Rational DOORS	✓
<i>Innovator Modeling Platform</i>	✓	Modelo de requisitos SysML	---	✓
<i>MagicDraw</i>	✓	---	IRqA [353] (todas las versiones) Borland CALIBER (versiones Architect y Enterprise) Rational RequisitePro (MagicRQ plugin) Rational DOORS (MagicRQ plugin)	✓
<i>Modelio</i>	✓	Modelo de requisitos basados en SysML	---	✓
<i>ObjectiF</i>	✗	Casos de uso	---	✓
<i>Rational Rhapsody</i>	✓	Modelo de requisitos SysML	Rational DOORS Rational RequisitePro	✓
<i>Enterprise Architect</i>	✓	Modelo de requisitos basados en SysML	Rational DOORS (MDG Link)	✓
<i>Together Architect</i>	✓	---	Borland CALIBER	✓

Tabla 44 Gestión de requisitos en las herramientas para el desarrollo de aplicaciones siguiendo el enfoque de DSDM.

7.1.3. Objetivo y organización del capítulo

De acuerdo con Pohl [277] la creación y gestión de las relaciones de traza entre los artefactos de IR requiere una herramienta que le dé soporte de forma adecuada. Además, los diferentes tipos de relaciones de traza se deben soportar técnicamente, por tanto, el pre-requisito es que se hayan definido los tipos de relaciones y los artefactos de IR en un metamodelo. El propio metamodelo se puede utilizar para derivar la herramienta de soporte. Ese es el procedimiento que hemos seguido para el desarrollo de la herramienta que soporte el proceso de IR dirigido por modelos SIREN_{MD} (Sección 4.7). En primer lugar, hemos definido el metamodelo REMM, que constituye el lenguaje

para la representación de especificaciones de requisitos textuales y, a partir de él, hemos generado semi-automáticamente el entorno REMM-Studio⁺. Decimos semi-automáticamente puesto que, aunque la mayor parte del código se ha generado automáticamente (gracias a la tecnología de implementación que hemos utilizado, Sección 7.2), este código ha tenido que ser adaptado y ampliado con nuevas funcionalidades.

En la presente tesis doctoral tratamos de contribuir al estado del arte de las herramientas de soporte al DSDM e IR, proponiendo un nuevo entorno de desarrollo de IR dirigido por modelos centrado en la reutilización de requisitos. La herramienta, que hemos denominado REMM-Studio⁺, incluye: 1) una serie de editores de modelos (Sección 7.3), que permiten construir y modificar los modelos de requisitos *para* y *con* reutilización, 2) una herramienta para validar los modelos (Sección 7.4) y 3) una herramienta que permita ejecutar las transformaciones de modelos (Sección 7.5). En primer lugar se presenta la tecnología utilizada para la implementación de la herramienta (Sección 7.2) y a continuación se detalla la funcionalidad implementada en cada una de las herramientas que integra REMM-Studio⁺. Por último, presentamos las conclusiones y aportaciones al estado del arte (Sección 7.6).

7.2. Arquitectura de la herramienta

El consorcio OMG ofrece un conjunto de estándares relacionados con el MDSD, en concreto, con su propuesta en este campo, MDA. Entre estos estándares se incluye MOF (*MetaObject Facility*) [259] como lenguaje de meta-meta-modelado, esto es, como lenguaje para describir metamodelos. Actualmente, la implementación más ampliamente utilizada de MOF es un *plug-in* de Eclipse que se denomina *Eclipse Modelling Framework* (EMF) [43]. Realmente, EMF soporta un subconjunto de MOF, llamado EMOF (*Essential MOF*), pero permite a los diseñadores crear, manipular y almacenar tanto modelos como metamodelos. Esta es la razón por la que muchas iniciativas relacionadas con el DSDM se están desarrollando en el marco de Eclipse y EMF, por ejemplo, los proyectos GMP (*Graphical Modelling Project*) [80], anteriormente denominado GMF (*Graphical Modeling Framework*), o GMT (*Generative Modelling Technologies*) [79]. Así, el uso extendido de Eclipse por la comunidad DSDM, junto con el hecho de que es un entorno gratuito, de código abierto e independiente de la plataforma, nos ha llevado a elegirlo como entorno de desarrollo para implementar REMM-Studio⁺.

Habitualmente, el proyecto EMF se utilizaba junto con GEF (*Graphical Editing Framework*), para proporcionar una representación visual a los modelos de dominio expresados en EMF. En este contexto, el proyecto GMF surge para simplificar el uso combinado de EMF y GEF en la construcción de aplicaciones de modelado (editores de modelos) en Eclipse. Podríamos decir que GMF establece un puente entre EMF y GEF donde: EMF proporciona la noción de metamodelado, GEF es responsable de la representación gráfica y GMF permite visualizar y editar elementos EMF en diagramas GEF [338]. Además, GMF está diseñado para eliminar la necesidad de que el cliente gestione la persistencia de los diagramas y le permite centrarse en gestionar la lógica del negocio en el editor.

Actualmente, el proyecto GMF se ha renombrado como GMP. GMP proporciona un conjunto de componentes generativos e infraestructura del tiempo de ejecución para el desarrollo de editores gráficos basados en EMF y GEF [80]. Este proyecto incluye: *GMF Tooling* y *GMF Runtime*, que hemos utilizado para el desarrollo de REMM-Studio+. A continuación pasamos a describir cada uno de estos marcos de desarrollo en base a la documentación ofrecida por la página oficial de Eclipse [80] y la presentación de Tikhomirov y Shatalin [338].

- *GMF Tooling*: proporciona un enfoque dirigido por modelos para generar editores gráficos totalmente funcionales en Eclipse. Está formado por: 1) un conjunto de modelos que definen el editor de diagramas, que capturan los aspectos gráficos, de herramientas y metamodelado, y 2) un generador de código extensible, capaz de adaptar el código generado para que cumpla los requisitos específicos.

La Figura 63 muestra los principales componentes y modelos que se utilizan durante un desarrollo basado en GMF, particularizado al caso del editor de modelos de requisitos *para* reutilización que hemos denominado `REMMf.diagram` (*REMM models for reuse*). El esquema es el mismo para el resto de editores generados: `REMMb.diagram`, editor de modelos *con* reutilización y `REMMextra.diagram`, editor de elementos de REMM que no son requisitos propiamente (stakeholders, casos de prueba, referencias u objetos externos y glosario de términos). Todos los editores comparten el mismo modelo de dominio, REMM, y definen el resto de modelos de forma específica. El contenido de cada modelo es el siguiente:

- o *Modelo de dominio* (`*.ecore`): contiene la definición completa del metamodelo de requisitos REMM utilizando el lenguaje EMOF (Capítulo 5). Representa los datos con los que queremos trabajar (requisito, stakeholder, caso de prueba, etc.). Además, EMF proporciona la capacidad para almacenar la información de los modelos, por defecto utiliza XMI (*XML Metada Interchange*) para hacer persistente la definición de los modelos.

Una vez que se especifica el metamodelo, se puede generar las clases de implementación en el lenguaje Java a partir de él. Para ello, primero se crea el modelo generador (`*.genmodel`), con el que se puede generar, de forma automática, el código Java correspondiente al modelo EMF. Además, el código generado puede ser extendido a mano, si es necesaria su adaptación. En la implementación de REMM-Studio+, antes de continuar con el proyecto GMF, generamos el código Java de las clases definidas en el metamodelo REMM como se muestra en la Figura 62.

- o *Modelo de definición gráfica* (`*.gmfgraph`): contiene la información relativa a los elementos gráficos (figuras, etiquetas, nodos, enlaces, etc.) que aparecerán en un entorno de ejecución basado en GEF. Estos componentes gráficos no tienen una relación directa con los elementos del modelo de dominio para los que proporcionarán la representación gráfica y la capacidad de edición.

- o *Modelo de definición de las herramientas* (*.gmftool): se utiliza para diseñar la paleta que mostrará, en el editor, los elementos que pueden crearse en los modelos.

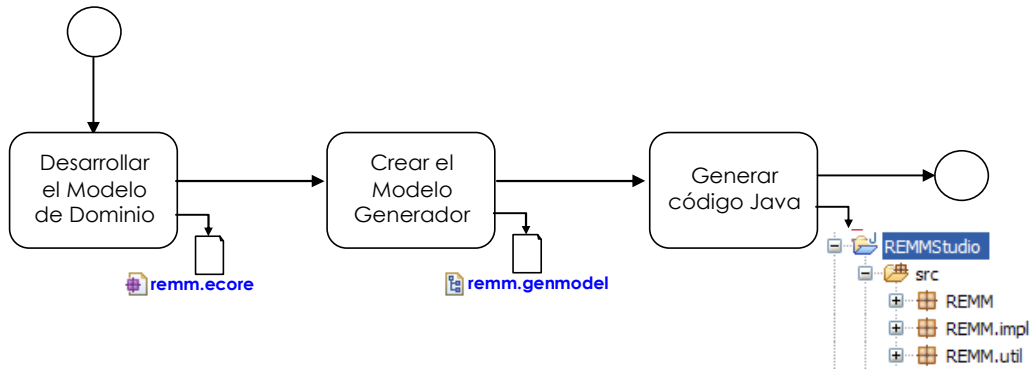


Figura 62 Flujo de trabajo de un proyecto EMF.

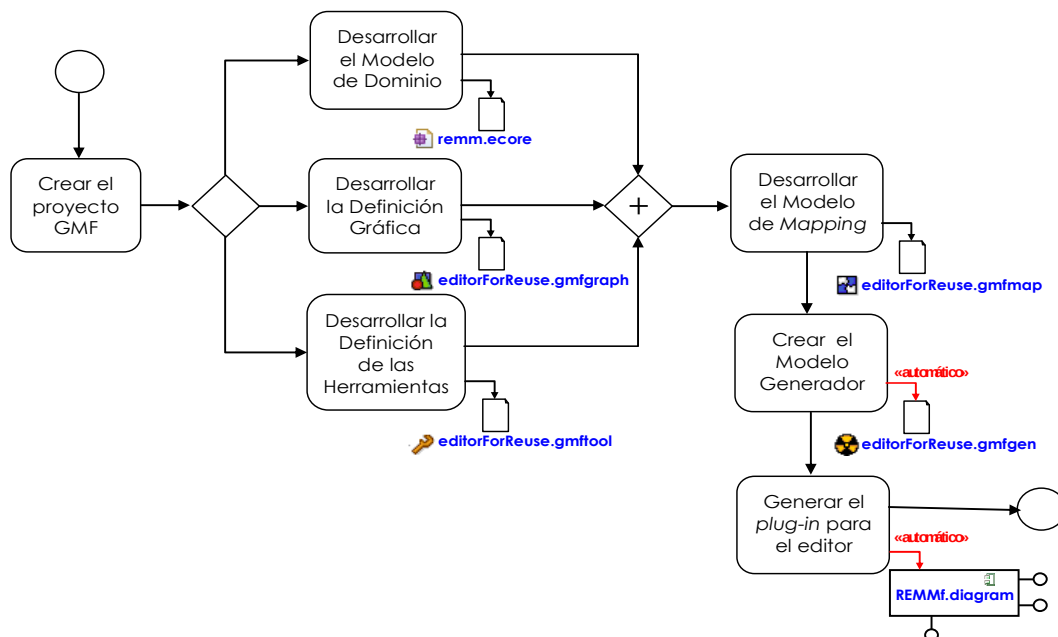


Figura 63 Marco para la generación del editor de modelos de requisitos para reutilización.

- o *Modelo de correspondencia o mapping* (*.gemfmap): enlaza las definiciones gráficas y de la paleta de herramientas con los elementos del modelo de dominio. Manteniendo esta separación entre los modelos se favorece la reutilización, puesto que la misma definición gráfica y de herramientas se puede utilizar para diferentes modelos de dominio.
- o *Modelo generador* (*.gmfgen): Una vez que se ha definido el modelo de correspondencia, se crea automáticamente un modelo generador. Este

modelo se puede configurar para establecer, por ejemplo, la extensión de los ficheros que contendrán la representación gráfica de los modelos, la activación o no de las reglas de validación definidas, el nombre del *plugin* que se va generar, entre otras. Finalmente, el modelo *gmfg* se utiliza para generar el código del *plug-in* del editor de modelos (*.diagram), también automáticamente. El entorno de ejecución (*GMF Runtime*) hará de puente entre la notación (diagrama) y el modelo de dominio (metamodelo) cuando un usuario trabaja con el editor, y también proporciona persistencia y sincronización entre ambos.

- *GMF Runtime* proporciona muchas características que tendrían que implementarse a mano si se utilizaran directamente EMF y GEF para el desarrollo. *GMF Runtime* está formado por un conjunto de componentes para editores gráficos como, por ejemplo, impresión de modelos, exportación de imágenes o barras de herramientas. Podría verse como una librería que simplifica la creación de los diagramas. Además, ofrece un conjunto de puntos de extensión que se utilizan por el código generado para extender el comportamiento de los diagramas en tiempo de ejecución. Estos puntos de extensión son los que se han utilizado en REMM-Studio⁺ para ampliar las funcionalidades de los editores y adaptar el código generado a las necesidades de los editores de modelos de requisitos. La relación entre *GMF Tooling* y *GMF Runtime* se muestra en la Figura 64.

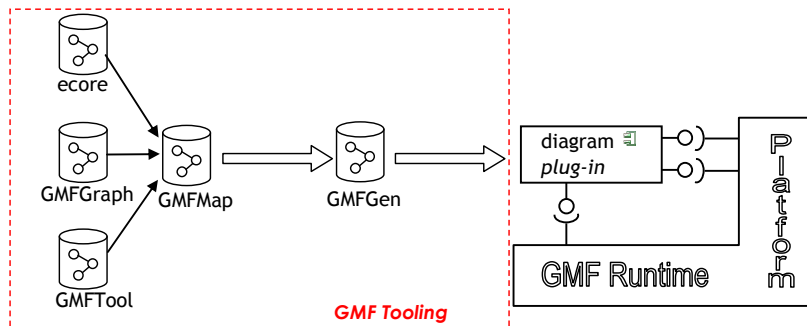


Figura 64 Conexión de *GMF Tooling* y *GMF Runtime*[338].

Una vez creados los editores de modelos, los elementos que faltan para terminar de configurar la arquitectura de REMM-Studio⁺ (Figura 65), son los módulos para las transformaciones a partir de los modelos de requisitos creados con dichos editores. Para dar soporte al proceso ReToCo (Capítulo 6), las transformaciones que se tienen que poder ejecutar son transformaciones: 1) modelo a texto, para generar la especificación textual de requisitos (SRS, *Software Requirements Specification*), y 2) modelo a modelo, para generar el modelo de conforme a un DSL (en nuestro caso de estudio el modelo de una aplicación domótica) a partir del modelo de requisitos con reutilización.

7.3. Editores de modelos

Para el modelado de los requisitos se necesitan fundamentalmente dos editores diferentes, aquel que permita la creación de modelos de requisitos *para* reutilizar (REMMf, Sección 7.3.2) y el que permita la construcción de modelos de requisitos *con* reutilización (REMMb, Sección 7.3.3). Estos editores permiten la definición de los requisitos y el establecimiento de las relaciones entre ellos (relaciones entre-requisitos). La definición del resto de artefactos implicados en la especificación de requisitos (stakeholders, casos de prueba, objetos externos, glosario de términos), con los que también se relacionan los requisitos (relaciones extra-requisitos), se crean en un editor desarrollado para este fin, para favorecer la legibilidad de los modelos (REMMextra, Sección 7.3.1).

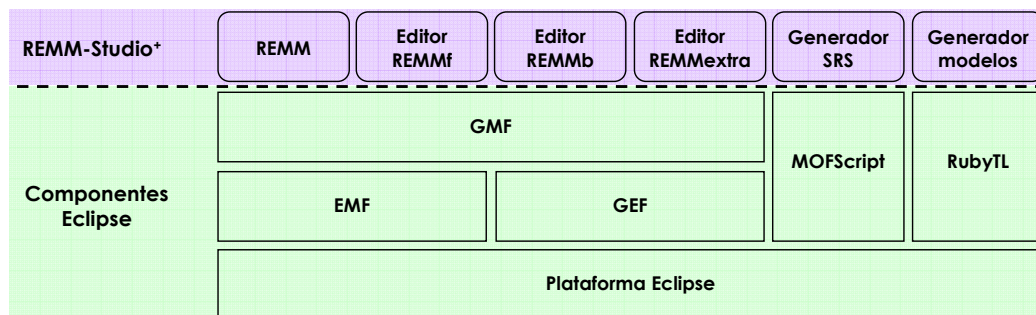


Figura 65 Arquitectura de REMM-Studio+.

Nos vamos a inspirar en el caso de estudio propuesto por Leffingwell y Widring [186], con el objeto de manejar un conjunto reducido de elementos, que sean los suficientemente significativos para ilustrar las funcionalidades soportadas por los editores. El caso de estudio tiene que ver con la definición de los requisitos de la unidad central de control (*Central Control Unit*) del proyecto denominado HOLIS (*HOME Lighting automation System*), un sistema innovador para automatizar la iluminación de residenciales de lujo. Los requisitos propuestos por Leffingwell y Widring se van a generalizar en un catálogo de requisitos reutilizables relacionados con la automatización de los sistemas de iluminación, utilizando el editor REMMf. Después, se reutilizarán para la definición de los requisitos de una aplicación concreta dentro de esta familia, la iluminación de residenciales de lujo, utilizando REMMb.

7.3.1. REMMextra: editor de los artefactos de IR relacionados con los requisitos

En la Figura 66 se muestra el editor de los artefactos de IR que se relacionan con los requisitos, que hemos denominado REMMextra. Como el resto de editores de modelos, está formado por cinco paneles, que son:

- 1) *Explorador de ficheros*, donde se muestran los ficheros que conforman el proyecto. El contenido del fichero seleccionado se mostrará tanto en la zona de modelado como en el panel visor del fichero.

- 2) *Visor del fichero*, muestra el esquema completo del modelo y sombrea la zona que se está viendo en la zona de modelado.
- 3) *Zona de modelado*, se utiliza para definir el modelo de requisitos que contiene los artefactos que se ofrecen en la paleta.
- 4) *Paleta de artefactos*, muestra los elementos que se pueden crear en el modelo.
- 5) *Ventana de propiedades*, muestra los atributos del elemento del modelo que se ha seleccionado. Alguno de los atributos se representan gráficamente en el modelo, pero el resto se configuran desde esta ventana de propiedades.

En la Figura 66 se muestra el proyecto TesisExample que contiene el modelo de requisitos *para* reutilización, conforme a REMM, que hemos denominado repositorio.remm. Este modelo tiene asociadas distintas vistas gráficas que se representan utilizando los diferentes editores, el modelo de requisitos propiamente dicho, que se representa en el editor REMMf (repositorio.remmf_diagram, Sección 7.3.2), y un modelo de artefactos de IR relacionados con los requisitos por cada catálogo que contiene el repositorio. Como veremos en la sección siguiente, el repositorio de requisitos contiene dos catálogos y por eso en el explorador aparecen dos ficheros con extensión .remmextra_diagram.

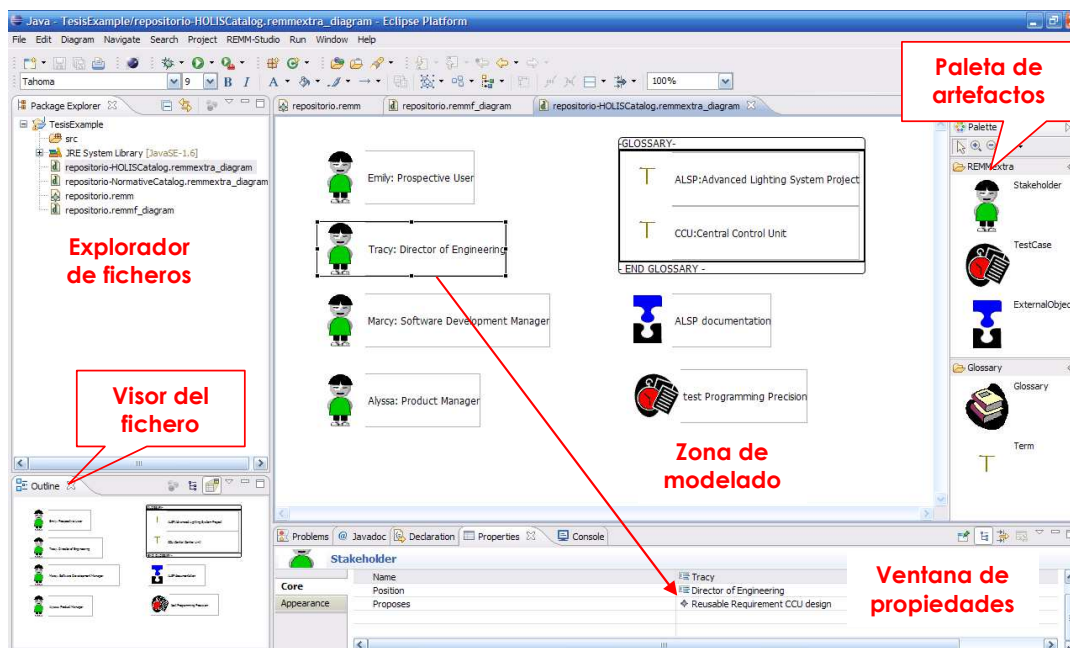


Figura 66 REMMextra: editor de artefactos de IR relacionados con los requisitos.

En el modelo de la Figura 66 se han definido los stakeholders implicados en la especificación de requisitos y para cada uno de ellos, además del nombre y el cargo en la empresa, se puede establecer el conjunto de requisitos que propone. En concreto, Tracy, Director de Ingeniería, ha impuesto la restricción de diseño que establece que el diseño de la unidad de control se tiene que copiar de la definida en el proyecto ALSP (Req_6 en la Figura 68). El término ALSP se ha definido en el glosario de términos y

también se ha definido como un objeto externo, puesto que la documentación de este proyecto se referencia para complementar la especificación de requisitos actual. Por último, en este editor se definen también los casos de prueba, como el que comprueba si la precisión del programador de la unidad de control es del tiempo establecido (por ejemplo, un minuto). Al igual que ocurre con los stakeholders, cada término del glosario, objeto externo o caso de prueba tiene asociado el conjunto de requisitos con los que se relaciona.

7.3.2. REMMf: editor de modelos de requisitos para reutilización

El modelo de requisitos *para* reutilización representa la especificación de un repositorio de catálogos de requisitos reutilizables (Sección 5.4). Los catálogos del repositorio pueden ser de dos tipos: dominio o perfil. Para ilustrar estos conceptos, hemos definido en el repositorio: 1) el *catálogo de dominio HOLIS*, que contiene los requisitos genéricos de un sistema de iluminación automática para el hogar (Figura 69), y el 2) *catálogo de perfil Normative* (Figura 70), que contiene los requisitos legales y normativos de la empresa, que deben cumplirse en los proyectos que desarrolla. En realidad, a modo de ejemplo, mostramos sólo algunos de los requisitos de cada uno de los catálogos. Como vimos en la sección anterior, cada uno de los catálogos tendrá asociado un conjunto de artefactos de IR (stakeholders, casos de prueba, etc.) que definimos con el editor REMMextra. La Figura 67 muestra la ventana del editor REMMf, que contiene el esqueleto inicial del modelo de requisitos, donde se ha especificado que el repositorio contiene dos catálogos: HOLIS y Normative.

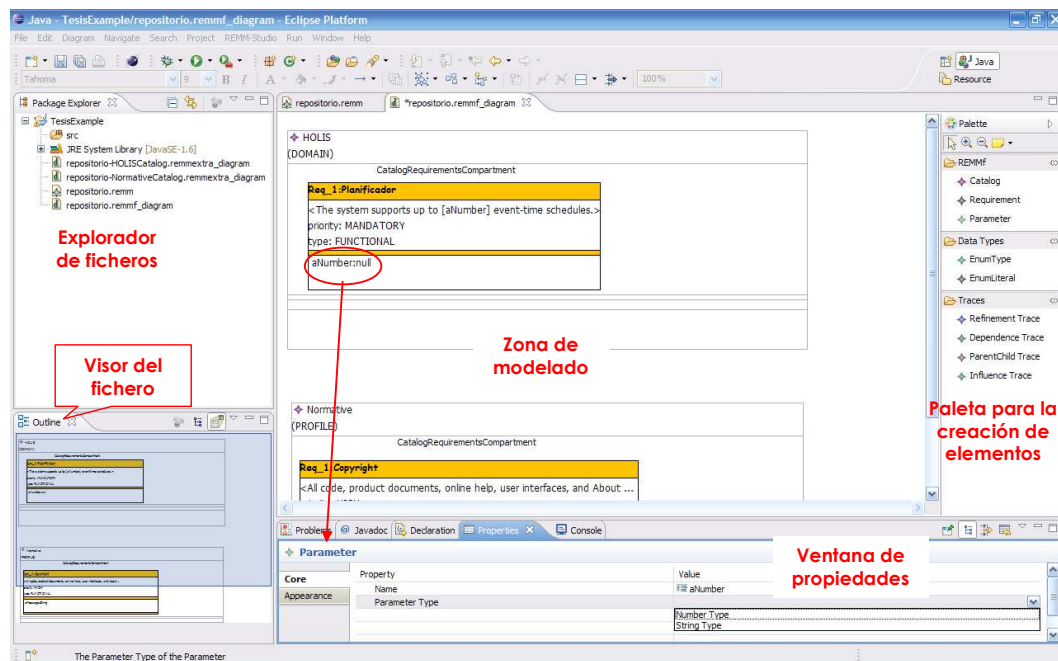


Figura 67 REMMf, editor de modelos *para* reutilización.

Merece la pena incidir en el hecho de que cada editor gráfico representa una vista del mismo modelo de requisitos (que se recoge de manera completa en el fichero `.remm`), lo que permite que los elementos descritos en uno de los editores también se reflejen en los otros. Por ejemplo, en la Figura 68 se muestran parte de las propiedades (atributos) de la restricción de diseño especificada por el requisito Req_6. En la lista de propiedades hemos señalado aquellas que referencian a elementos que se han definido con el editor REMMextra, como son: 1) los términos del glosario que se han utilizado en la especificación del requisito; 2) el stakeholder que lo ha propuesto; y 3) el objeto externo que establece la ubicación de la documentación del proyecto ALSP al que se hace referencia.

Uno de los elementos claves en el modelado de requisitos *para* reutilización, es la definición de *requisitos parametrizados* que especifican puntos de variación (Sección 5.4). Los parámetros de un requisito se pueden crear de forma manual, desde la paleta que ofrece el editor, o de forma automática, cuando se redacta el texto del requisito el editor crea un parámetro por cada cadena de texto que esté comprendida entre corchetes (“[” y “]”), con el tipo del parámetro igual a `null` (Figura 67). Una vez creados los parámetros, se puede configurar el tipo adecuado a través de la ventana de propiedades. Por defecto, los tipos que hay disponibles en todos los catálogos son el numérico (`NumberType`) y las cadenas de texto (`StringType`), aunque se pueden definir nuevos tipos enumerados (`EnumType`) utilizando la paleta. Un tipo enumerado representa un conjunto finitos de valores (`EnumLiteral`). En la Figura 69 se muestra el tipo enumerado `FormatClockType` que representa los posibles formatos del reloj {12h o 24h}. La cardinalidad del enumerado se ha definido con valor máximo 2, lo que significa que cuando se reutilice este requisito, en el momento de la instanciación del parámetro, se puede seleccionar uno o los dos valores del conjunto de literales.

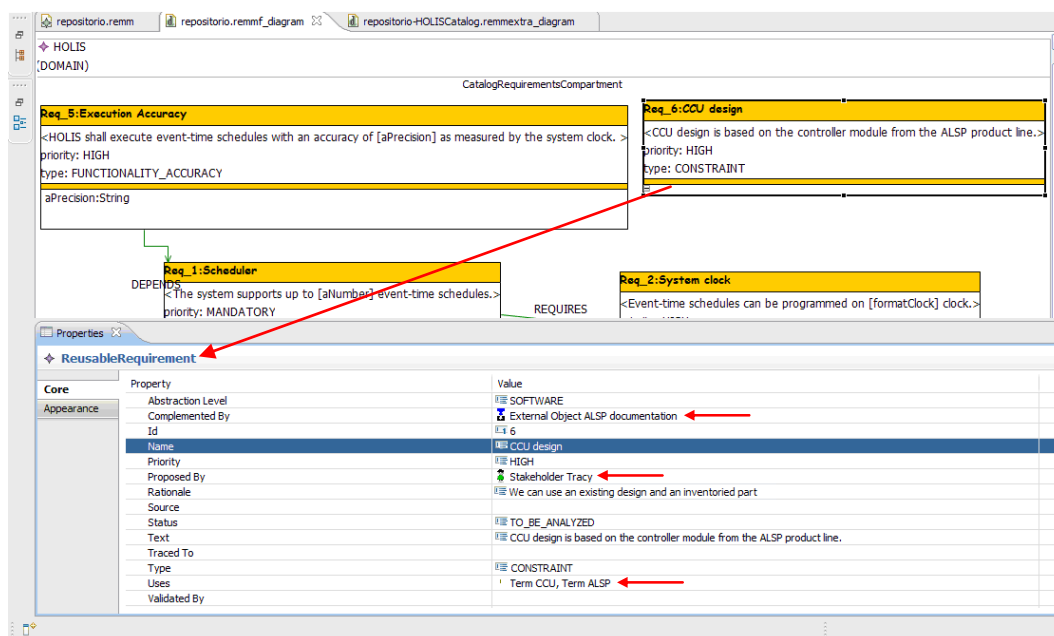


Figura 68 Propiedades de un requisito sincronizadas con el editor REMMextra.

Por otro lado, los requisitos reutilizables que se definan (parametrizados o no) se relacionarán entre ellos mediante relaciones de traza: refinamiento, dependencia, padre-hijo e influencia (Sección 5.3.3). En la Figura 69 y la Figura 70 se muestran ejemplos de estas trazas. Además, podemos ver en la Figura 69 que es posible establecer una condición sobre las trazas que determina qué se tiene que cumplir para que la traza exista, en la Figura 69 se establece que el número de eventos soportados tiene que ser mayor que cero. Las condiciones se tienen que definir conformes a la gramática presentada en la Sección 5.5.1, de no ser así se informará del error al validar el modelo.

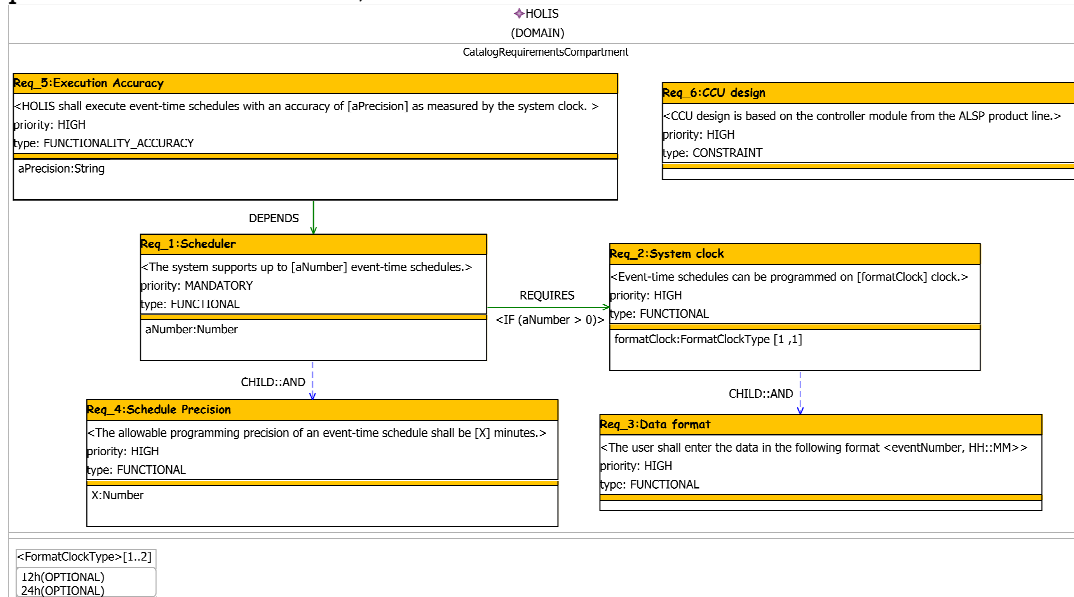


Figura 69 Catálogo HOLIS en el repositorio.

Por último, la variabilidad en los modelos de requisitos para reutilización también se puede expresar utilizando relaciones de traza exclusivas, que representan que dos requisitos no pueden ser reutilizados, en la especificados de un nuevo producto, al mismo tiempo, sino que sólo puede seleccionarse uno de los dos. La Figura 70 muestra un ejemplo en el catálogo Normative. Nótese que en este caso, a diferencia del resto de trazas, el enlace que se establece entre los dos requisitos es bidireccional, puesto que los requisitos son mutuamente exclusivos.

7.3.3. REMMb: editor de modelos de requisitos con reutilización

Un modelo de requisitos *con* reutilización representa la especificación de requisitos de un nuevo producto o aplicación, que se define reutilizando los requisitos previamente establecidos en un repositorio de requisitos reutilizables. Los aspectos claves del editor que da soporte a este tipo de modelos, y que hemos denominado REMMb (Figura 71), son: el apoyo a la selección de requisitos y la instanciación de los requisitos parametrizados que se reutilicen.

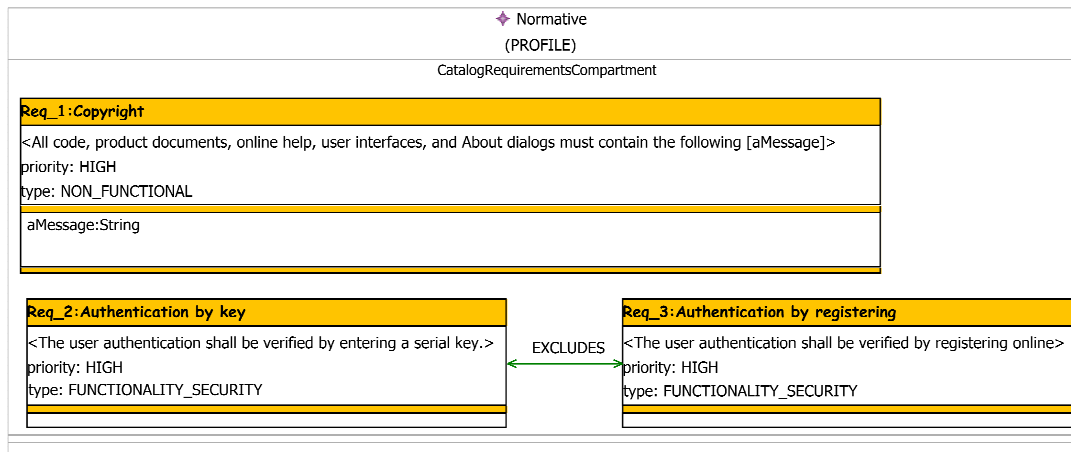


Figura 70 Catálogo Normativo del repositorio.

La Figura 71 muestra la ventana principal del editor REMMb. En el editor se ha desplegado el menú *popup* que contiene las opciones de menú que dan soporte a la reutilización de requisitos.

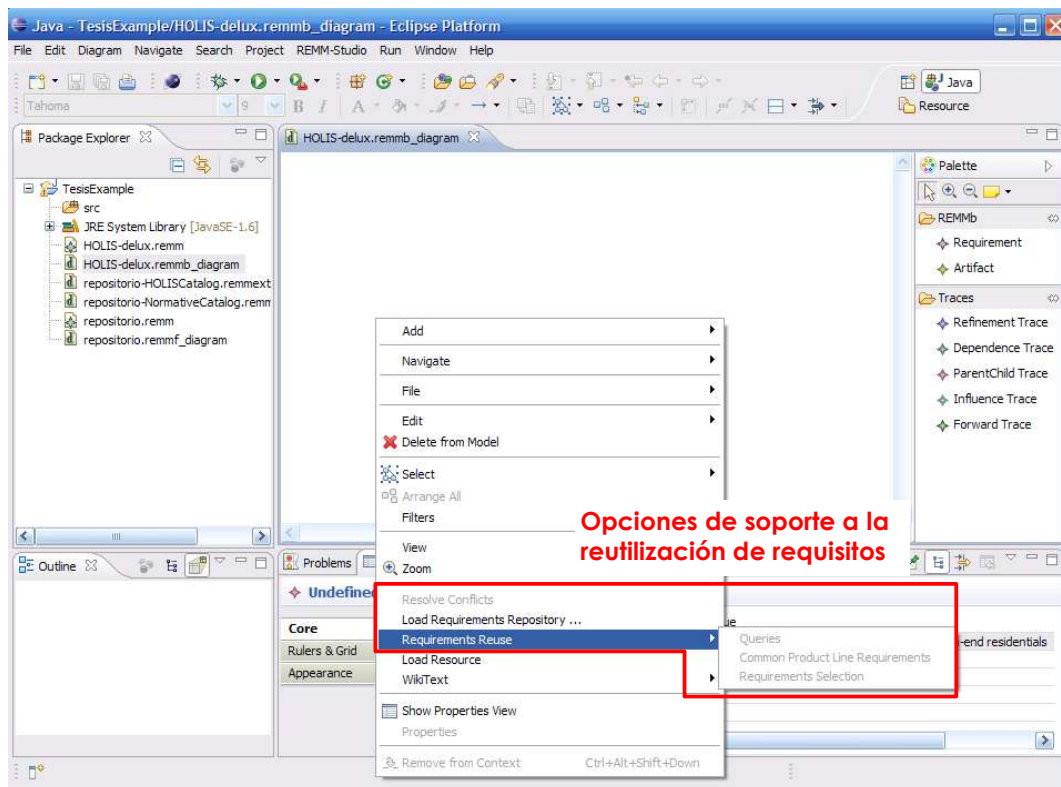


Figura 71 REMMb, editor de modelos de requisitos *con* reutilización.

En la figura se puede ver que estas opciones, salvo la opción “Cargar un Repositorio de Requisitos” (*Load Requirements Repository*), están todas desactivadas, esto es debido

a que no se puede seleccionar ningún requisito si no se ha establecido previamente el repositorio que se va a utilizar. Una vez que se ha cargado un repositorio, las opciones del menú pasan a estar activas. En realidad, la carga de un repositorio es equivalente a la opción ofrecida por el editor de “Cargar un recurso” (*Load Resource*), la única diferencia es que utilizando la opción de cargar un repositorio se comprueba que el fichero seleccionado contiene, efectivamente, un modelo de requisitos que representa un repositorio de catálogos de requisitos reutilizables.

El submenú “Reutilización de Requisitos” (*Requirements Reuse*), ofrece las distintas opciones para la selección de requisitos de algún catálogo definido previamente. Estas opciones son: búsqueda de requisitos en función del valor de sus atributos, selección de los requisitos comunes de una línea de productos (SPL) o selección manual de los requisitos a partir del listado completo de los requisitos de un catálogo. A continuación detallamos cada una de estas opciones.

- *Búsqueda de requisitos (Queries)*: a través de un asistente se selecciona el catálogo del repositorio sobre el que se va a hacer la búsqueda, el atributo que se va a utilizar, y el valor de dicho atributo. La ventana para la inserción del valor del atributo se adapta en función del tipo del atributo, por ejemplo, si el tipo es una cadena de texto se proporciona un campo de texto libre en el que insertar la cadena que se va a buscar: literalmente o utilizando el “*”, para indicar cualquier cadena que contenga el valor que se está especificando. Por ejemplo, en la Figura 72, se busca en el catálogo HOLIS todos los requisitos que contenga en su texto la cadena *event-time*. El resultado de la búsqueda es el que se muestra también en la Figura 72, donde vemos que, no sólo se han reutilizado los requisitos, sino también las relaciones de traza que se habían definido entre ellos.
- *Selección de los requisitos comunes de una SPL (Common Product Line Requirements)*: dado que los requisitos de un catálogo de dominio se pueden entender como la especificación de requisitos de una SPL (tras su etapa de Ingeniería del Dominio), esta opción simplifica la búsqueda de los requisitos comunes a todos los productos de la SPL. Estos requisitos habrán sido marcados con prioridad obligatoria en la especificación de los requisitos del catálogo de dominio. Nótese que al mismo resultado llegaríamos con la opción de menú anterior buscando todos los requisitos que tengan el valor MANDATORY en el atributo *priority*. La ventaja de esta opción, frente a la anterior, es que, en este caso, el asistente sólo muestra los catálogos de dominio en la lista de los catálogos contenidos en el repositorio.
- *Selección de Requisitos (Requirements Selection)*: mediante esta opción de menú, es posible seleccionar manualmente los requisitos contenidos en los catálogos. Para facilitar la selección, la ventana de diálogo (Figura 73) muestra los requisitos de manera jerárquica haciendo explícitas las relaciones padre-hijo existentes (Sección 5.3.3). Además, el asistente lleva el control de las relaciones padre-hijo que son obligatorias (hijos AND) de manera que, cuando se selecciona un requisito padre, automáticamente se seleccionan los requisitos hijos obligatorios, y no permite que se deseccione un requisito padre mientras algún requisito hijo obligatorio esté seleccionado. En la Figura 73 se muestra el modelo de requisitos resultado de la selección de los requisitos marcados en la ventana de diálogo. Podemos observar que, como en el caso de la Figura 72, no sólo se han

reutilizado los requisitos, sino también las relaciones de traza que existen entre ellos.

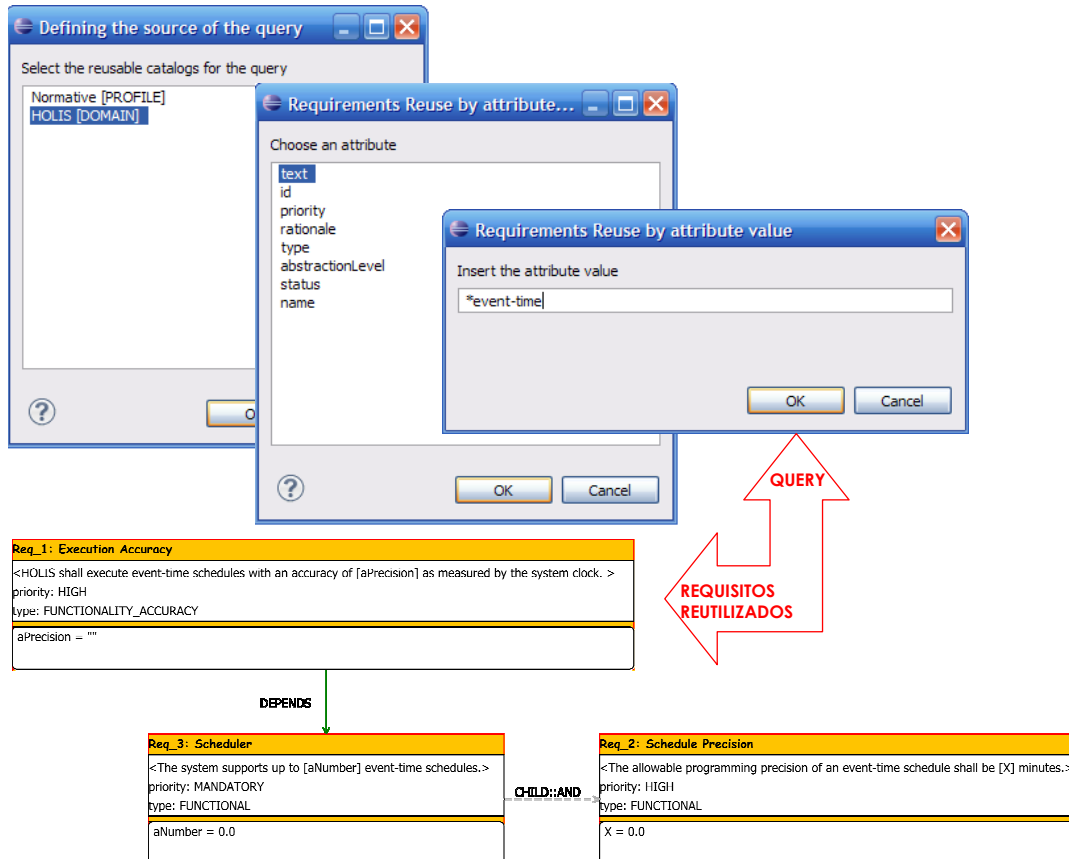


Figura 72 Búsqueda de requisitos en función del valor de sus atributos.

Además de reutilizar los requisitos desde el menú contextual del editor, es posible reutilizar los requisitos en el momento en el que se está definiendo un nuevo requisito de producto. En la Figura 74 se muestra esta circunstancia, se ha añadido un nuevo requisito utilizando la paleta del editor, y el menú contextual asociado al requisito incluye la opción de menú "Reutilización de requisitos" (*Requirements Reuse*). A través de una ventana de diálogo, parecida a la de la Figura 73, que muestra un listado plano del contenido del repositorio, se podrá seleccionar el requisito a partir del cual se quiere definir este nuevo requisito de producto. Una vez seleccionado, en el modelo de requisitos actual, no sólo se copian los valores de este requisito sino que se reutilizan todos los requisitos "que cuelgan", en sentido figurado, de él. Por ejemplo, si seleccionamos el requisito Req_1 del catálogo HOLIS (Figura 69) el resultado sería el modelo que se muestra en la Figura 74.

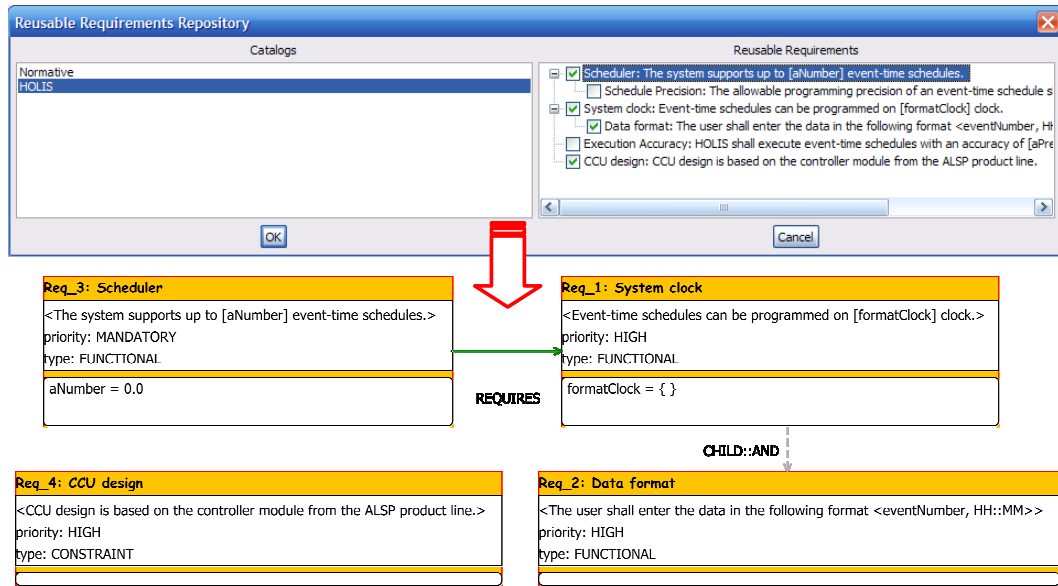


Figura 73 Selección de requisitos para su reutilización.

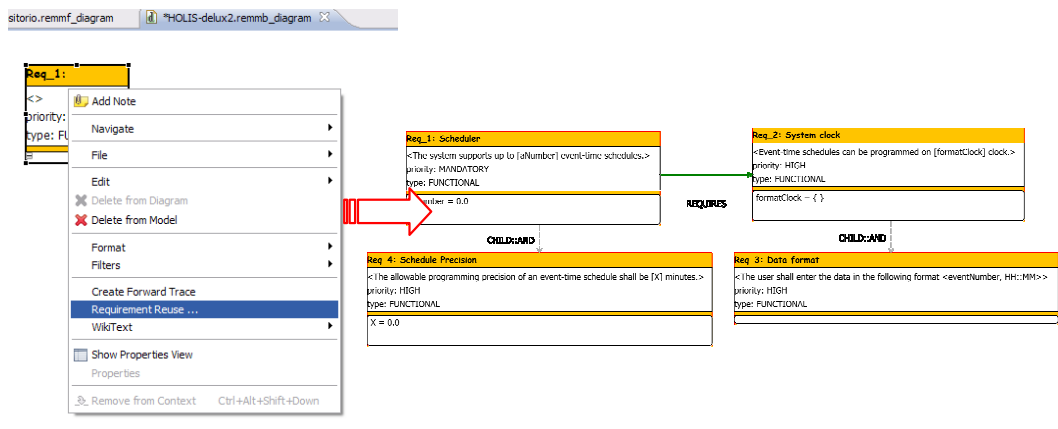


Figura 74 Selección de un único requisito para su reutilización.

Una vez reutilizados los requisitos, bien sea mediante el menú contextual del editor o del requisito de producto, aquellos que contengan algún parámetro (requisitos parametrizados) tendrán que ser instanciados. Esto es, hay que definir un valor para cada uno de los parámetros de acuerdo a los requisitos del nuevo producto o aplicación que se esté especificando. La instanciación de los parámetros se hace a través de una nueva pestaña en la ventana de propiedades implementada para este fin. El campo destinado a la lectura del nuevo valor se adapta en función del tipo del parámetro siendo: un campo de texto si el tipo es numérico o una cadena de texto, un desplegable con los valores de un enumerado si la cardinalidad máxima que se ha definido para el tipo es 1, o una ventana de diálogo con una lista multi-selección, en el caso de que la cardinalidad máxima del enumerado sea mayor que 1. En este último caso, se controla que los valores seleccionados no superen la cardinalidad máxima establecida. Por

ejemplo, en la Figura 75 se muestra como el parámetro del requisito Req_2 (reutilización del requisito Req_1 del catálogo HOLIS Figura 69) lo hemos instanciado con el valor 256, de manera que se permite la planificación de hasta 256 eventos. El valor establecido para el parámetro se sustituye también en el texto del requisito. En la Figura 75 se muestra también cómo se está definiendo el valor para el parámetro del requisito Req_1 a través de la pestaña de la ventana de propiedades. Al validar el modelo se comprueba que todos los requisitos hayan sido instanciados como veremos en la Sección 7.4.

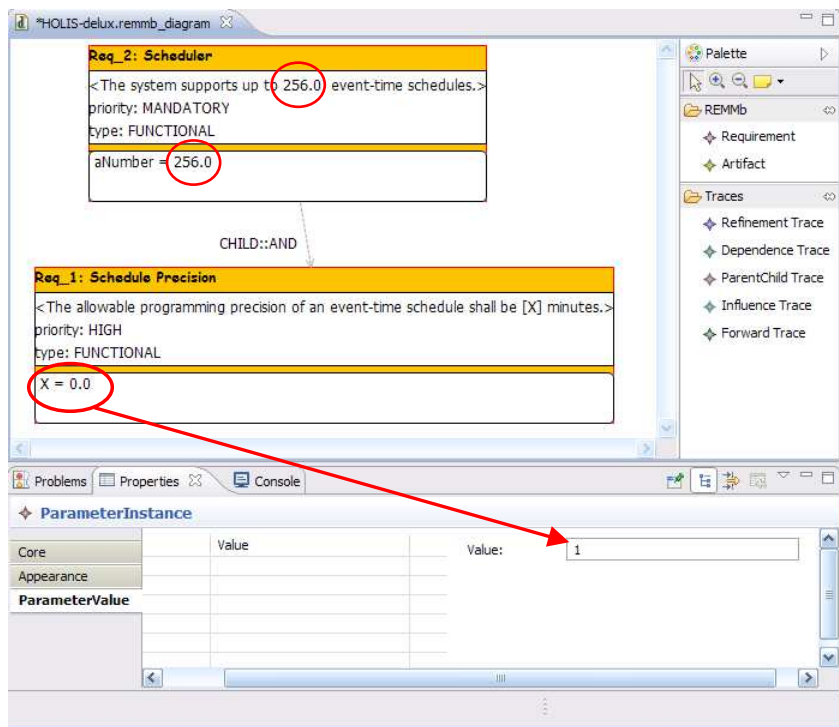


Figura 75 Instanciación de los requisitos parametrizados.

Por último, la opción de menú “Resolución de Conflictos” (*Resolve Conflicts*) en Figura 71, sirve para informar si se han seleccionado dos requisitos que se han definido como exclusivos en el catálogo de requisitos reutilizables. En el ejemplo de la Figura 76 se muestra cómo se han intentado reutilizar dos requisitos exclusivos. Podría parecer que la información obtenida mediante esta opción de menú es redundante, puesto que el error se ha marcado directamente en la relación de traza que une los requisitos, al validar el modelo. No obstante, este informe detecta el error aún cuando no exista explícitamente la relación de traza entre los requisitos.

7.3.4. Trazabilidad entre requisitos en REMMb y REMMf

Tanto en el editor REMMb como en el editor REMMf, se han definido dos nuevas vistas que permiten representar matricialmente y en forma de árbol, las relaciones de traza entre los requisitos (Figura 77). Cada tipo de traza tiene asociado un icono

diferente para favorecer la comprensión. Estas vistas complementan el informe, generado automáticamente mediante transformaciones de modelos (Sección 6.3.2), que muestran las trazas hacia delante, esto es, las relaciones de traza desde los requisitos a otros artefactos del desarrollo.

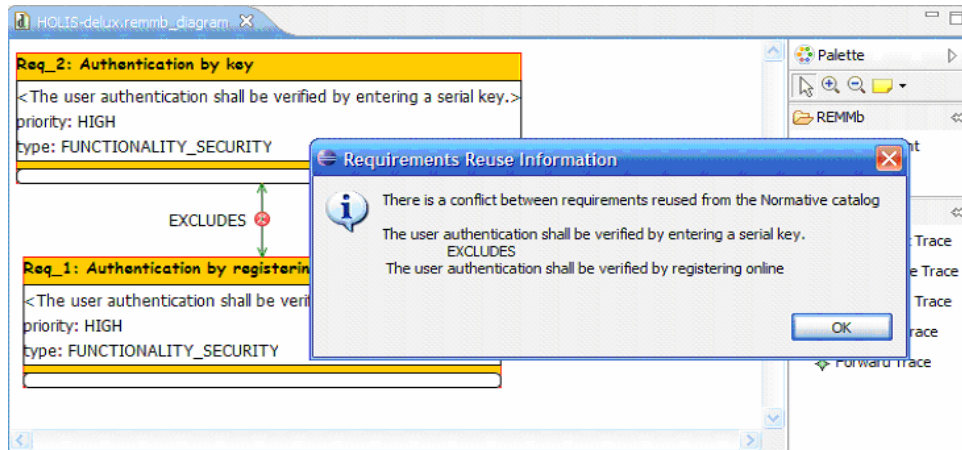


Figura 76 Informe sobre conflictos entre los requisitos reutilizados

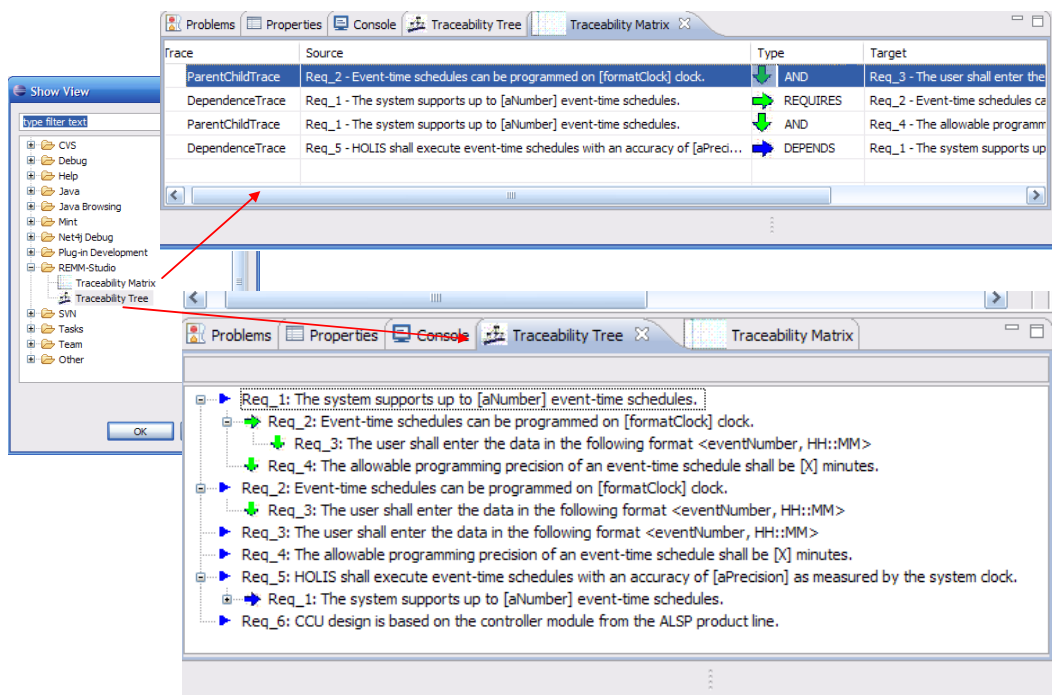


Figura 77 Árbol y matriz de traza entre-requisitos.

7.4. Herramienta para validar los modelos

En el metamodelo REMM se han definido reglas de validación OCL y Java que garantizan que los modelos de requisitos que se van a crear están bien formados (Sección 5.5). Una vez activadas estas reglas en el modelo generador (gmfggen, Figura 63), hemos programado que se comprueben cada vez que se salvan los modelos. A continuación mostramos algunos ejemplos en los que se ha detectado el incumplimiento de alguna de las reglas establecidas:

- 1) En la Figura 78 se muestra un extracto del catálogo de requisitos HOLIS en el que se ha detectado dos errores: 1) el enumerado `formatClockType` contiene dos literales obligatorios cuando la cardinalidad que se ha definido establece un máximo de 1 (Regla R5 en Sección 5.5.1), y 2) el tipo del parámetro `formatClock` no se ha establecido (restricción dada por la cardinalidad de la relación en el metamodelo, todo parámetro tiene que tener especificado el tipo).
- 2) En la Figura 76 se muestra un ejemplo en el que se ha detectado que hay dos requisitos que se han reutilizado que están relacionados mediante una relación de traza de dependencia exclusiva en un catálogo de producto (Regla R4 en Sección 5.5.1).
- 3) En la Figura 79 se muestra el error detectado en la relación entre dos requisitos porque el valor establecido para el parámetro hace que la evaluación de la condición sea `false` (Regla R11 en Sección 5.5.1).

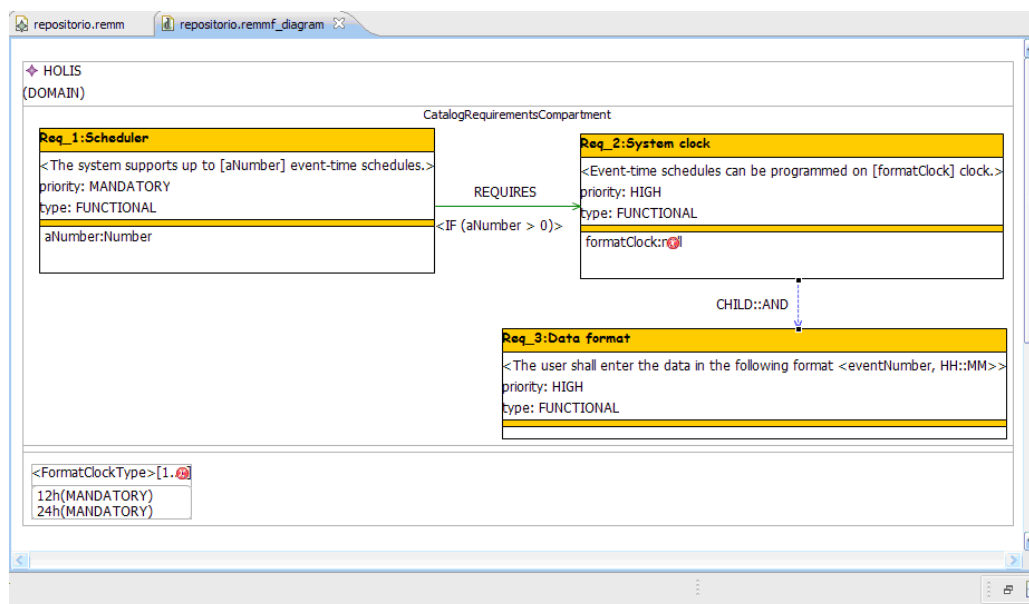


Figura 78 Errores encontrados en el modelo de requisitos para reutilización.

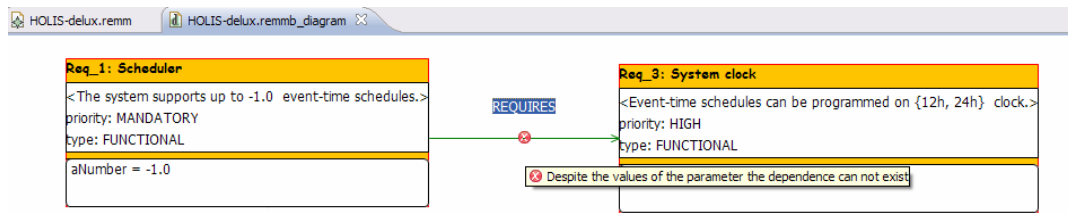


Figura 79 Error de validación del modelo debido al valor del parámetro.

7.5. Ejecución de las transformaciones de modelos

La versión actual de la herramienta REMM-Studio⁺ no dispone de un asistente que lleve a cabo las transformaciones de forma transparente al usuario (está prevista su inclusión en una versión posterior). La ejecución de las transformaciones debe configurarse y ejecutarse de forma manual. En el caso de las transformaciones modelo a texto, para generar la especificación textual de requisitos (Sección 6.4.1), debemos configurar las preferencias del *plugin* de MOFScript (Figura 80) para establecer: 1) la ubicación del metamodelo de requisitos (REMM.ecore); 2) el directorio donde se encuentran los modelos de entrada de la transformación (en realidad, el directorio que se indique es irrelevante puesto que luego se selecciona con un asistente el fichero que contiene el modelo de entrada); y 3) el directorio donde se encuentra la definición de la transformación. REMMf2SRS.m2t es el fichero que contiene las reglas de transformación a partir de los modelos de requisitos *para* reutilización, se ha establecido como directorio de salida, esto es, como directorio donde se van a guardar los ficheros generados durante la transformación, la carpeta repositorio-doc. REMMb2SRS.m2t es el fichero que contiene las reglas para la transformación del modelo de requisitos *con* reutilización y se ha establecido como directorio de salida la carpeta SRS.

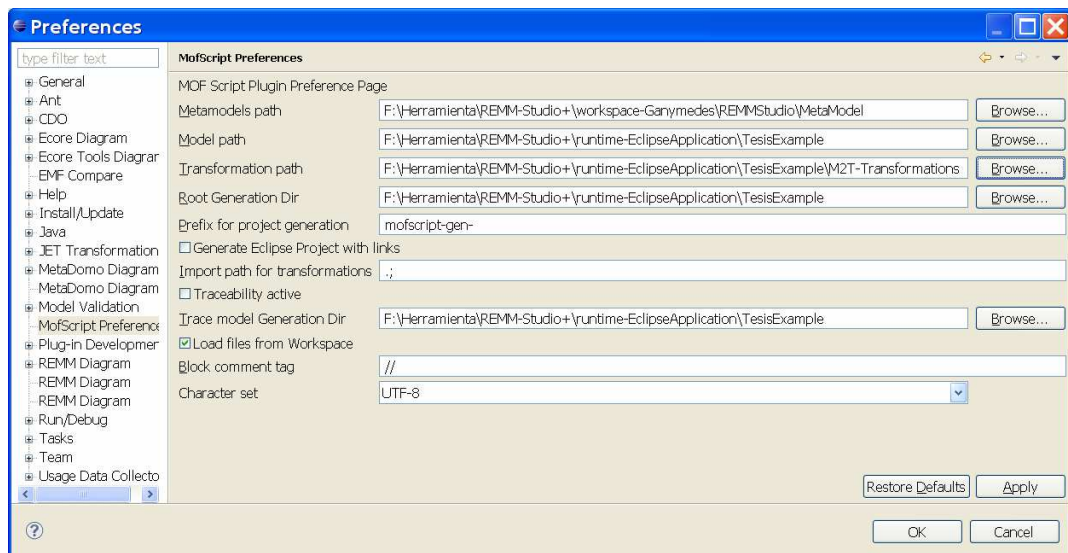


Figura 80 Configuración de las preferencias del plug-in MOFScript.

La Figura 81 muestra la ejecución de la transformación REMMf2SRS desde el entorno REMM-Studio⁺. En la figura se ve como se selecciona el fichero que contiene el modelo de requisitos (`repositorio.remm`) que se va a transformar. Como resultado de la ejecución, se genera la estructura de ficheros HTML explicada en la Sección 6.4.1 como se muestra en la Figura 82.

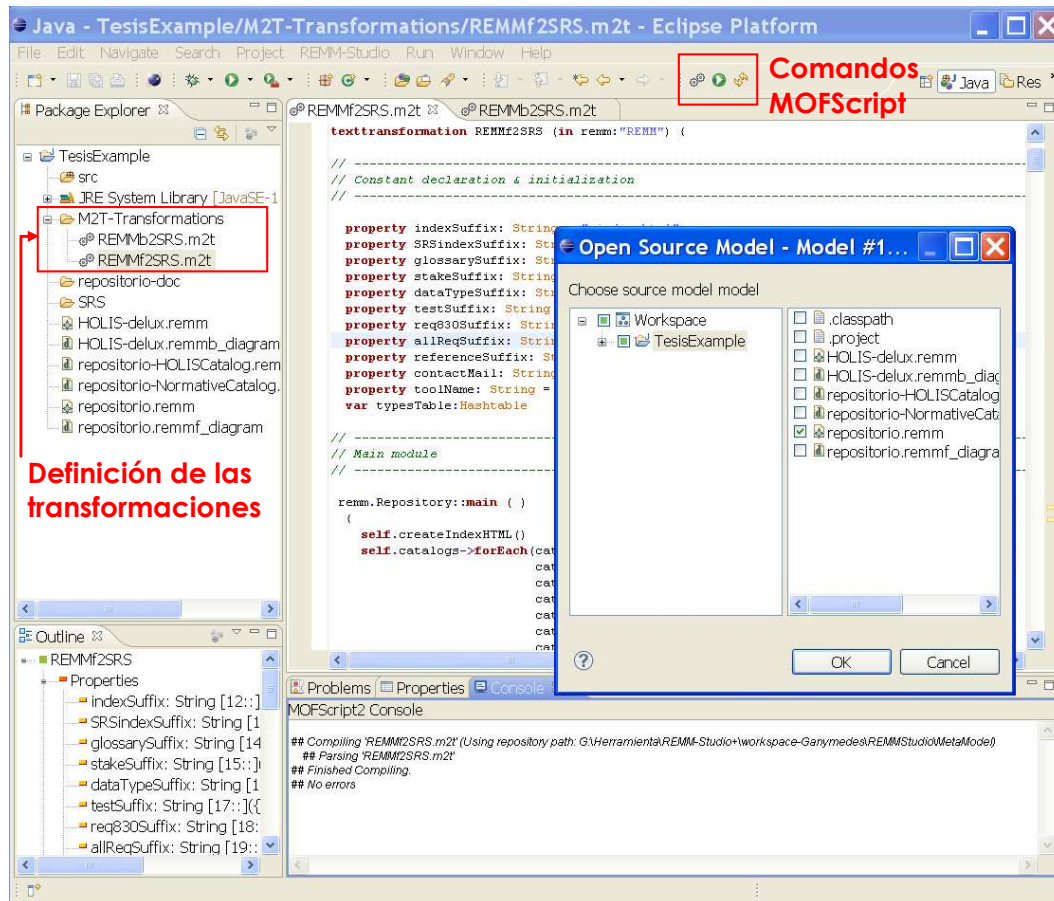


Figura 81 Ejecución de la transformación modelo a texto.

Para este caso de estudio sencillo no se han definido transformaciones modelo a modelo. Como vimos en la Sección 6.3, estas transformaciones tienen que definirse *ad-hoc* para cada dominio donde queramos aplicarlas. Además, este proceso requiere que se haya llevado a cabo un trabajo de modelado específico de dominio mediante el cual se haya definido el metamodelo que represente el lenguaje para poder especificar modelos del dominio en cuestión. La ejecución de estas transformaciones se presentan en el capítulo siguiente como parte de la instanciación del proceso ReToCo (Sección 6.2) al dominio de los sistemas domóticos.

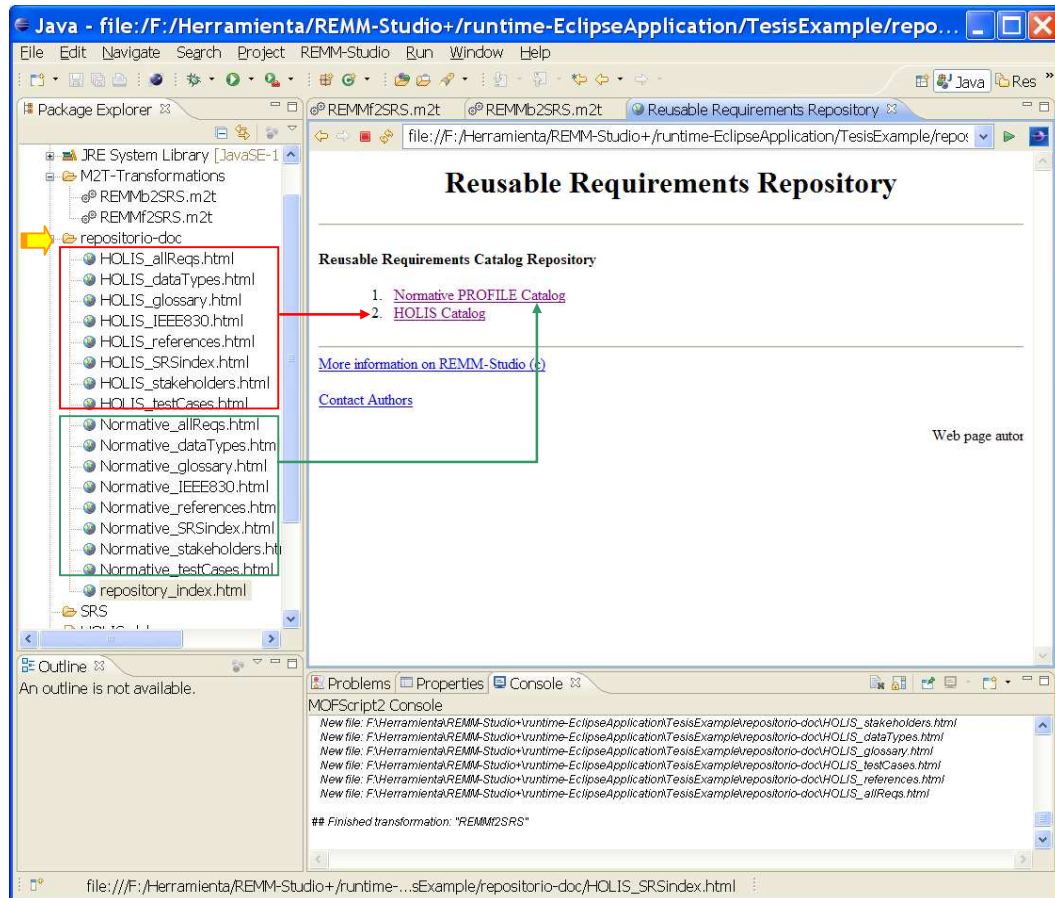


Figura 82 Resultado de la ejecución de la transformación modelo a texto.

7.6. Conclusiones y aportaciones al estado del arte

En este capítulo hemos presentado la herramienta REMM-Studio⁺, un entorno integrado para dar soporte a un proceso de IR dirigido por modelos como SIREN_{MD} (Sección 4.7). REMM-Studio⁺ ha sido desarrollada a partir del metamodelo REMM (Capítulo 5), con el objeto de proporcionar editores para el modelado de especificaciones de requisitos textuales. Dado que REMM presta especial atención a la reutilización de requisitos y el modelado de la variabilidad, REMM-Studio⁺ incluye editores para las dos facetas de la reutilización: modelado *para* reutilización y modelado *con* reutilización. Además, REMM-Studio⁺ proporciona: 1) un editor para la especificación de los artefactos de IR con los que se relacionan los requisitos (para favorecer la legibilidad de los modelos),

2) una herramienta para validar los modelos y comprobar, no sólo que son conformes al metamodelo REMM sino que cumplen las reglas OCL y Java definidas para cada editor (Sección 5.5) y 3) herramientas para la transformación de los modelos de requisitos en la especificación textual de requisitos (transformación modelo a texto) y en modelos conformes a un DSL (transformaciones modelo a modelo).

La herramienta desarrollada como parte de la presente tesis doctoral, REMM-Studio⁺, está en la línea de la herramienta SirenTool (Sección 4.6), desarrollada previamente dentro del grupo de investigación al que pertenece esta doctoranda, en cuanto a que proporciona soporte automatizado a la reutilización de requisitos. La mejora frente a SirenTool es la representación gráficamente de los requisitos textuales en modelos, lo que permite integrarlos fácilmente en un proceso de DSDM.

Las aportaciones de REMM-Studio⁺ al estado del arte podemos resumirlas como sigue:

- El metamodelo subyacente, a partir del cual se ha construido la herramienta, es REMM que, como concluíamos en el Capítulo 5, es el único metamodelo propuesto hasta el momento que se preocupa tanto de las relaciones de traza Post-RS (*Post Requirements Specification*) como de las relaciones de traza Pre-RS. La riqueza del modelo de trazas definidas en REMM mejora las soportadas por las herramientas de IR y DSDM actuales.
- Las herramientas de IR comerciales se centran en la especificación de requisitos del producto actual. En REMM-Studio⁺ se da soporte a las dos facetas implicadas en la reutilización de requisitos, modelado de requisitos *para* reutilización (incluyendo el modelado de la variabilidad) y el modelado de requisitos *con* reutilización (incluyendo el apoyo a la selección e instanciación de los requisitos parametrizados reutilizados).
- Se da soporte a la reutilización sistemática de requisitos, esto es, no sólo se reutilizan los requisitos sino también las relaciones de traza establecidas entre ellos. Las relaciones de traza definidas entre los requisitos se tienen en cuenta en el momento de la reutilización.
- Los modelos de requisitos son validados automáticamente, lo que favorece la etapa de validación de la especificación de requisitos.

Capítulo 8

Caso de estudio: Sistemas domóticos

8.1. Introducción

De acuerdo con Wohlin *et al.* [361], en una disciplina como la Ingeniería del Software es crucial el uso de métodos empíricos (como experimentos controlados, casos de estudio, encuestas o análisis post-mortem), que nos permitan evaluar y validar los resultados de la investigación. En esta tesis doctoral hemos aplicado, el proceso propuesto en el Capítulo 6 a un dominio concreto (o caso de estudio), el de los sistemas domóticos. Además, hemos evaluado, mediante la realización de una encuesta, el grado de satisfacción de los usuarios potenciales de nuestra propuesta, esto es, los expertos en el desarrollo de las aplicaciones domóticas.

En los últimos años, las tecnologías de la información se están integrando en el hogar y la vida cotidiana a gran velocidad. Este proceso ha dado lugar a un nuevo tipo de sistemas reactivos: los *sistemas domóticos* [320]. El objetivo principal de estos sistemas es dotar a las viviendas de un cierto grado de inteligencia que permita mejorar la calidad de vida de sus habitantes. Tareas tales como el encendido y regulación de luces de forma automática, control de la temperatura, corte de agua y gas cuando se detectan fugas o el control de los dispositivos del hogar de forma remota desde el móvil u ordenador con conexión a Internet, son algunas de las aplicaciones típicas del dominio domótico, que integran funciones de confort, ahorro energético, seguridad y comunicaciones.

Los problemas fundamentales en el desarrollo de los sistemas domóticos son: la ausencia de un estándar *de facto* para implementar este tipo de aplicaciones y el uso de herramientas de desarrollo dependientes de la tecnología utilizada. En este contexto, la tecnología se refiere al sistema que permite transferir datos de control de todos los componentes de gestión de edificios (por ejemplo, aire acondicionado, sistema de control de accesos, iluminación). Por un lado, existen varios estándares y protocolos adoptados por las empresas que lideran el mercado, pero según AENOR (*Asociación Española de Normalización y Certificación*) es improbable que se establezca una única tecnología dominante en el campo de la domótica a corto plazo. Destacan, por su alta cuota de mercado y alto crecimiento: la tecnología *KNX*, que ha sido aprobada como estándar

européo (CENELEC⁵ EN50090: *Home and Building Electronic Systems*) e internacional (ISO/IEC⁶ 14543-3-X: *Information technology - Home Electronic Systems Architecture*), y la tecnología *Lonworks*, que también ha sido aprobada como estándar internacional (ISO/IEC 14908: *Information technology - Control network protocol*), europeo (EN14908: *Open Data Communication in Building Automation, Controls and Building Management - Control Network Protocol*) y americano (ANSI/CEA⁷ EIA-709.1 *Control Network Protocol Specification*).

Por otro lado, cada uno de los estándares existentes dispone de sus propias herramientas de desarrollo de software con el que crear las aplicaciones domóticas y programar los dispositivos. Estas herramientas son proporcionadas bien por los fabricantes de dispositivos (en caso de sistemas propietarios) o bien por las asociaciones que dan soporte a la tecnología (en casos de sistemas normalizados). Las herramientas suelen ser dependientes de la plataforma y orientadas a la generación de código, con una sintaxis poco intuitiva y con un bajo nivel de abstracción por lo que se hace necesario un alto grado de especialización por parte de los desarrolladores y se les obliga a trabajar en un espacio muy cercano a la solución.

En consecuencia, en las etapas iniciales del desarrollo de un sistema domótico se debe seleccionar una tecnología concreta, puesto que las herramientas y dispositivos a usar dependen de esta elección. Estos hechos hacen que el desarrollo de aplicaciones domóticas sea totalmente dependiente de la plataforma, siendo muy complicado incrementar el nivel de abstracción y trabajar con conceptos del dominio domótico en lugar de trabajar con elementos de la tecnología.

Tradicionalmente, todo el proceso de desarrollo de un sistema domótico es realizado por un especialista del dominio (ingeniero de proyecto), que: 1) recoge los requisitos del cliente para una instalación (elementos que se desean integrar, funcionalidad requerida, elección de una tecnología concreta) basándose en su propia experiencia, 2) realiza la instalación, 3) coordina el seguimiento y, por último, 4) programa los dispositivos para conseguir la funcionalidad requerida. La mayor limitación de esta forma de proceder es la dificultad de conseguir *artefactos software reutilizables*, prefiriendo, por norma general, una solución eficiente y totalmente a medida, antes que diseñar soluciones generales para facilitar su reutilización. Como consecuencia de esto, cada nuevo sistema debe construirse prácticamente desde cero, aunque su lógica y estructura sean casi idénticas a las de otros sistemas desarrollados previamente (incluso simplemente implementados sobre plataformas diferentes). Otras carencias de las metodologías de desarrollo tradicionales para este tipo de sistemas incluyen la ausencia de una notación única para representar los mismos conceptos funcionales que aparecen en las diferentes plataformas y la falta de gestión de requisitos de la aplicación.

Para solucionar las limitaciones de las metodologías tradicionales de desarrollo de sistemas domóticos, en el grupo DSIE, de la Universidad Politécnica de Cartagena (UPCT), se inició una línea de investigación en este campo que ha culminado con la

⁵ CENELEC es el Comité Europeo para la Estandarización Electrotécnica.

⁶ ISO/IEC es la Comisión Electrotécnica Internacional de la Organización Internacional de Estandarización.

⁷ ANSI/CEA es la Asociación de Consumidores de Electrónica del Instituto Americano de Estandarización.

defensa de dos tesis doctorales a cargo de los doctores D. Manuel Jiménez [147] y D^a. Francisca Rosique [296]. Por un lado, en la tesis del Dr. Jiménez, se define un marco general para el desarrollo de sistemas domóticos aplicando técnicas procedentes de la Ingeniería del Software y se propone un DSL (*Domain Specific Language*) para domótica, que permite elevar el nivel de abstracción y modelar los sistemas domóticos de forma independiente de la plataforma. Por otro lado, la tesis de la Dra. Rosique es una continuación de la tesis anterior, en la que se refina el DSL inicial y se propone una metodología de desarrollo que sigue un enfoque de DSDM (*Desarrollo de Software Dirigido por Modelos*), desde los requisitos hasta el código de la aplicación para una tecnología concreta. En este capítulo vamos a seguir los trabajos realizados por los doctores Jiménez y Rosique para la presentación de los aspectos relacionados con los sistemas domóticos, el DSL y la propuesta de DSDM para el desarrollo de este tipo de sistemas.

Dado el éxito de la aplicación de la tecnología de DSDM en este campo, consideramos que se trataba de un caso de estudio adecuado para aplicar el proceso que se propone en esta tesis doctoral, complementando el proceso definido por el DSIE. El resultado de la instanciación del proceso ReToCo (*From Requirements To Code*), presentado en el Capítulo 6, al dominio de los sistemas domóticos, es fruto de la colaboración con los miembros del grupo DSIE implicados en la línea de investigación de domótica. El grupo DSIE nos ha proporcionado su experiencia en el desarrollo de este tipo de aplicaciones, el DSL (Sección 8.2) y el proceso de transformación de los modelos domóticos al código (Sección 8.3). Nuestra aportación se encuadra en las primeras etapas del desarrollo, definiendo un catálogo de requisitos reutilizables (Sección 8.4) y las transformaciones modelo a modelo (m2m) para generar los modelos domóticos, *automáticamente*, a partir de los requisitos del catálogo (Sección 8.5). De esta forma, una vez que están generados los modelos domóticos se puede continuar con el proceso DSDM propuesto por el DSIE, que permite generar el código de la aplicación. La propuesta conjunta ha sido evaluada con éxito mediante la realización de una encuesta en la que han participado tanto expertos en el desarrollo de sistemas domóticos como expertos en el DSDM. Las características y los resultados de la encuesta se presentan en la Sección 8.6.

8.2. Habitation: un DSL para los sistemas domóticos

Cualquier sistema domótico incorpora un número de elementos presentes en todas las tecnologías y estándares propios de este dominio, tales como controladores, sensores, motores, botones, etc. Estos elementos, denominados *unidades funcionales*, representan la mínima parte funcional en la que se puede descomponer un dispositivo domótico. Estos elementos difieren en la arquitectura y los protocolos que utilizan, pero son idénticos en las capacidades que proporcionan. Esto es, cada plataforma, estándar o fabricante proporciona sus propios dispositivos, aunque todos ellos intentan cubrir las mismas necesidades funcionales. La definición de unidades funcionales, en lugar de dispositivos ligados a una tecnología concreta, es la pieza clave para la especificación de sistemas domóticos independientes de la plataforma final de implementación.

Jiménez *et al.* [148] proponen el DSL Habitation para el desarrollo de sistemas domóticos y muestran cómo el uso del DSL, integrado en un enfoque de DSDM, resuelve los inconvenientes de los procesos de desarrollo tradicionales presentados en la Introducción (Sección 8.1). La sintaxis abstracta del DSL se recoge en un único metamodelo (Figura 83) aunque incluye dos vistas claramente diferenciadas:

- *Vista Catálogo* (destacada en color rosa en la Figura 83), permite la definición y el mantenimiento de un catálogo de unidades funcionales.
- *Vista Aplicación* (destacado en color amarillo en la Figura 83), permite el modelado de una aplicación domótica.

Esta separación: 1) favorece la reutilización de las unidades funcionales, definidas en el catálogo, en diferentes aplicaciones; y 2) asegura que se pueda mantener el catálogo actualizado en todo momento, sin necesidad de modificar el metamodelo.

8.2.1 Vista Catálogo del DSL Habitation

El concepto principal del metamodelo en la Vista Catálogo es el CatalogRoot, que representa el catálogo de elementos reutilizables. Este catálogo está compuesto a su vez por: el *catálogo de unidades funcionales* (FUnitCatalog) y el *catálogo de servicios* (ServiceCatalog). Por un lado, el catálogo de unidades funcionales incluye la definición de las unidades funcionales (FUnitDefinition) que pueden contener parámetros (ParameterDef). Por ejemplo, la definición de la unidad funcional SMS, que representa el dispositivo encargado de enviar un mensaje al teléfono móvil cuando salta la alarma, incluye dos parámetros denominados SMSTelNumber y SMSMessage (ambos de tipo cadena de texto, StringType) que son necesarios para configurar el número de teléfono del destinatario y el mensaje que se va a enviar, respectivamente. Por su parte, el catálogo de servicios incluye la definición de los servicios (ServiceDefinition). Los *servicios* permiten a las unidades funcionales interactuar entre sí, por ejemplo, conmutar (*switch*), regular (*dimmer*), mover (*move*), entre otros. Muchos de estos servicios se repiten entre las unidades funcionales, con lo que la definición del catálogo de servicios permite la reutilización de las definiciones en cualquier unidad funcional.

La Figura 84 muestra un fragmento del modelo que representa el catálogo de unidades funcionales propuesto para el modelado de aplicaciones domóticas. Las definiciones de las unidades funcionales se organizan por categorías (Category) como por ejemplo, 1) la categoría *botones* (Buttons), que incluye la definición de botones de pulsación (PushButton, PB), botones de pulsación para subir (PB-Up) o bajar (PB-Down), o 2) la categoría *luces* (Lights), que incluye la definición de las luces para encender y apagar (Light OnOff, LO), luces regulables (Light Dimmable, LDM) o luces regulables fluorescentes (Fluorescent Dimmable, LFM).

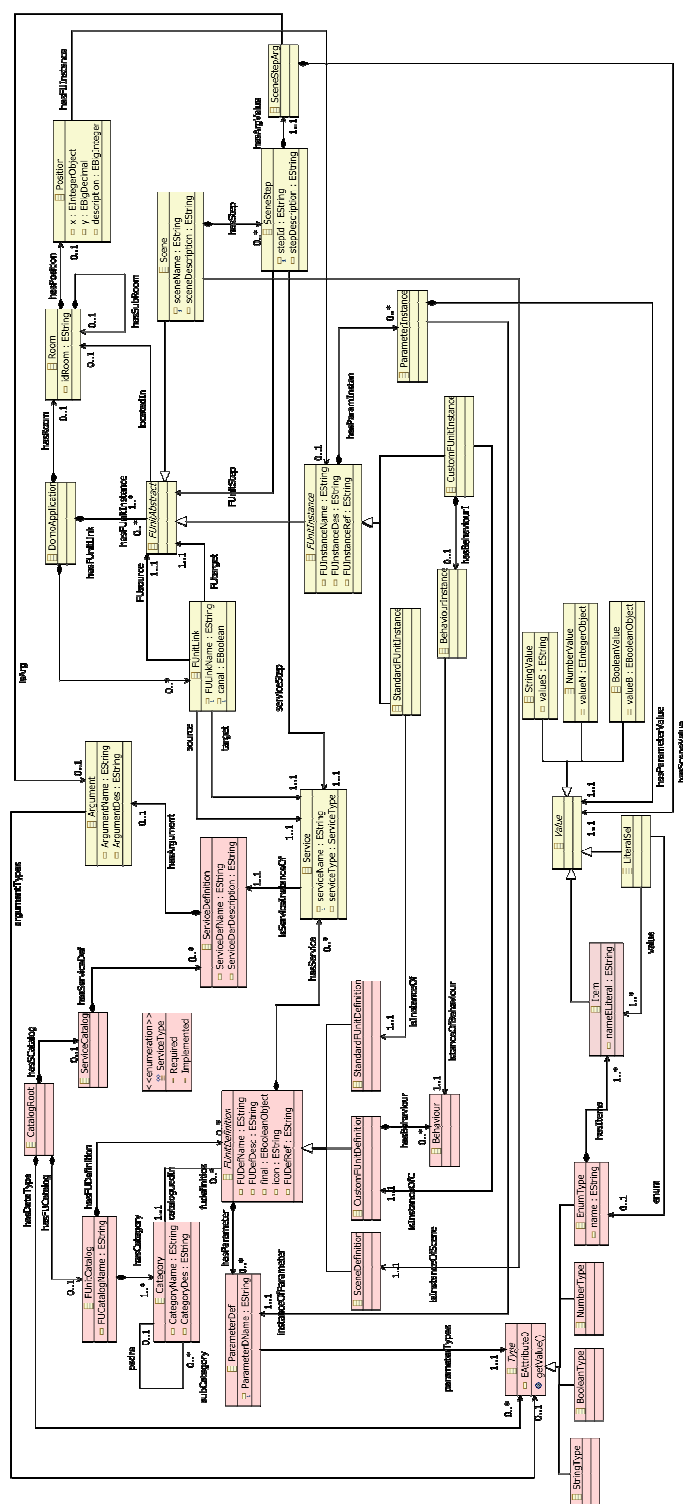


Figura 83 Metamodelo Habitation. Un DSL para sistemas domóticos.

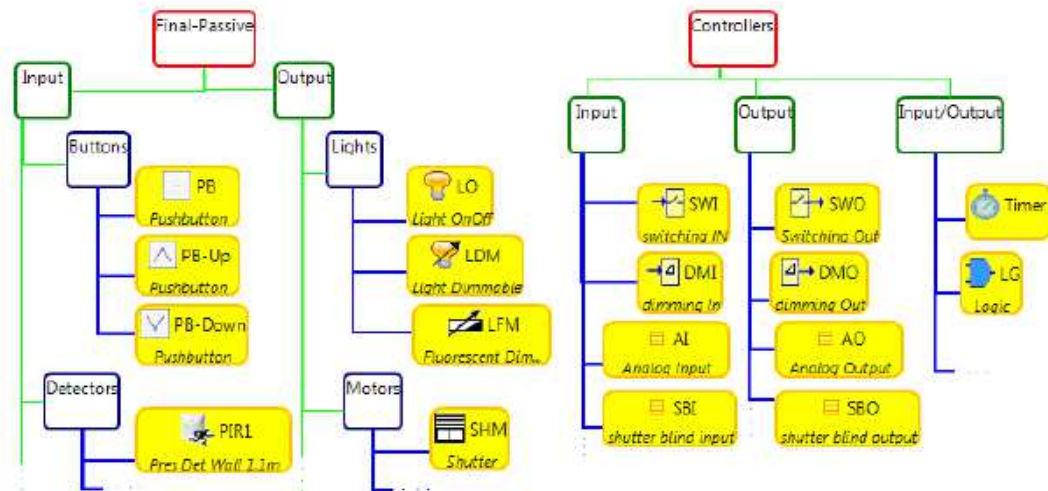


Figura 84 Fragmento del catálogo de unidades funcionales [296].

8.2.2. Vista Aplicación del DSL Habitation

El concepto DomoApplication, en la Vista Aplicación del DSL (Figura 83), representa un modelo de una aplicación domótica. Este modelo está formado, fundamentalmente por unidades funcionales (FUnitAbstract) que se conectan mediante enlaces (FUnitLink) a través de servicios (Service). Básicamente, la especificación de una aplicación domótica se realiza instanciando las unidades funcionales que están predefinidas en el catálogo y conectándola mediante enlaces. Estas instancias de unidades funcionales deben ser configuradas añadiendo los valores necesarios a sus parámetros (ParameterInstance), en el caso de que la definición de la unidad funcional los incluyera. Por ejemplo, la creación en un modelo de una instancia de la unidad funcional SMS implicaría el establecimiento de los parámetros SMSTelNumber y SMSMessage con los valores adecuados, por ejemplo "625356789" y "Peligro. Ha saltado la alarma", respectivamente. En el caso de los controladores (parte derecha de la Figura 84), los parámetros determinan el comportamiento interno de los dispositivos, por ejemplo, DMI (*dimming in*) es el controlador de entrada para regular las luces, su parámetro DMILongPushTime se debe configurar para establecer los segundos que se debe tener pulsado el botón para que el dispositivo detecte una pulsación larga y empiece a regular la intensidad de la luz. En caso de no detectar una pulsación larga simplemente se produce una conmutación, es decir se pasa de encendido a apagado, o viceversa.

Los enlaces entre las unidades funcionales establecen la forma en la que estas unidades van a interactuar con el resto del sistema. En cierto modo, se crea una especie de diagrama de actividades que se inicia a partir de los eventos disparados por las unidades funcionales de entrada, tales como botones o sensores. Al crear un enlace, se indican los servicios que se interconectan. Estos servicios son instancias de las definiciones de los servicios predefinidas en el catálogo (Sección 8.2.1). Estos enlaces

pueden comportarse como enlaces de tipo *canal* (atributo `canal` en la metaclass `FUnitLink`), que representan una conexión hardware (se muestra en rojo en los modelos, ver las figuras de la Sección 8.6), o bien como un enlace normal (se muestra en verde en los modelos, ver las figuras de la Sección 8.6). Por ejemplo, la unidad funcional LO que representa un punto de luz fijo se conecta con la unidad funcional SWO (*output switch*) mediante un enlace tipo canal, utilizando el servicio `hwSwitch`, esta conexión permite que la luz conmute (*switch*) de encendida a apagada, y viceversa.

8.3. Proceso de desarrollo dirigido por modelos para los sistemas domóticos

El grupo DSIE de la UPCT [296] ha propuesto una nueva metodología de DSDM para el desarrollo de sistemas domóticos (Figura 85), que es conforme a la propuesta del OMG, MDA [170]:

- 1) *Nivel CIM*: se definen los requisitos de la aplicación domótica que se va a desarrollar. Relacionados, por ejemplo, con la seguridad, motorización, iluminación y climatización (HVAC son las siglas de *Heating, Ventilation and Air Conditioning*)
- 2) *Nivel PIM*: se diseña el sistema domótico utilizando el DSL Habitation, que incluye conceptos del dominio independientes de la tecnología de implementación que se va utilizar.
- 3) *Nivel PSM*: se define un modelo integrado de componentes que permite la generación de código para una tecnología específica, en concreto para la plataforma KNX/EIB.

Siguiendo la filosofía de DSDM, los modelos definidos en cada nivel son conformes a un metamodelo. Además, se han definido transformaciones de modelos automáticas para pasar del nivel PIM al PSM y generar el código a partir del modelo PSM. Entre el nivel CIM y el PIM la correspondencia se establece de manera manual. A nivel CIM se ha definido un catálogo de requisitos de domótica que están enlazados *explícitamente* con fragmentos de modelos conformes al DSL. Estos fragmentos representan o implementan dicho requisito. De esta forma, reutilizando los requisitos definidos en el catálogo, se pueden reutilizar los fragmentos de los modelos (conformes al DSL) que están relacionados con los requisitos, que habrá que componer manualmente utilizando el editor. En la Figura 86 se muestra gráficamente esta propuesta (extraída de [301]). En ella vemos que reutilizando los requisitos R2, R3 y R4, definidos en el catálogo, se reutilizan (se copian y se pegan) los modelos que implementan cada uno de estos requisitos (también definidos previamente), en el modelo de la aplicación. Este modelo de la aplicación podrá ser refinado, o adaptado, utilizando el editor de modelos para el DSL.

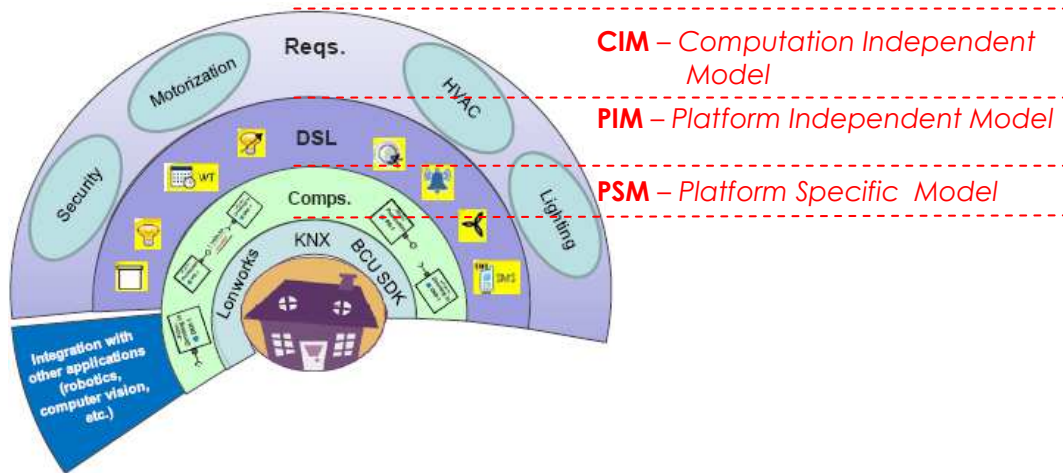


Figura 85 Visión general de la metodología de DSDM para sistemas domóticos [296].

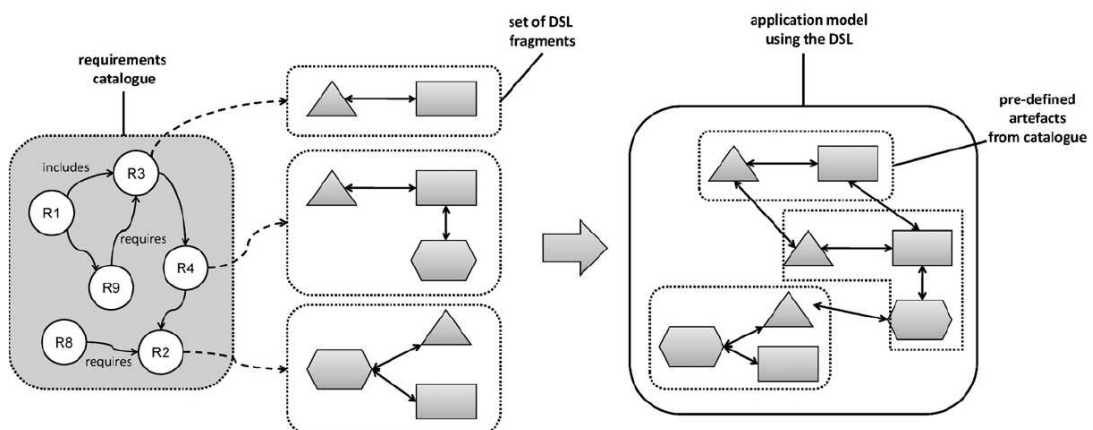


Figura 86 Generación de modelos a partir de los requisitos [301].

8.3.1. Aportaciones de esta tesis doctoral al proceso definido

La colaboración con el grupo DSIE se enmarca a nivel CIM, con el objeto de mejorar la etapa de especificación de requisitos y, sobre todo, generar automáticamente los modelos conformes al DSL a partir de los modelos de requisitos. Nuestras aportaciones a este dominio de los sistemas domóticos, fruto de los trabajos presentados en esta tesis doctoral, son:

- 1) **El metamodelo de requisitos REMM.** REMM mejora el metamodelo de requisitos propuesto por la Dra. Rosique (Figura 87) en el sentido de que permite la definición de requisitos parametrizados, contribuyendo a la genericidad del catálogo de requisitos reutilizables, y también establece un conjunto predefinido de relaciones entre requisitos, tales como requiere, excluye, padre-

hijo (Sección 5.3.3). En la propuesta de Rosique los tipos de relaciones se definen en tiempo de modelado de la especificación de requisitos, con lo que la semántica de las relaciones no está fijada de antemano. Un tipo de relación (TypeOfRelationship en Figura 87) se caracteriza por un *nombre* y una *descripción* que se especifican mediante una cadena de texto (tipo EString). En consecuencia, distintos usuarios podrían utilizar distintas cadenas de texto para representar la misma relación, o al contrario, utilizar la misma cadena cuando se quieren representar relaciones diferentes. La ausencia, en la propuesta de la Dra. Rosique, de un modelo de traza entre requisitos definido a nivel de metamodelo, como si que existe en REMM (Sección 5.3.3), dificulta la reutilización de requisitos y las transformaciones de modelos. En un proceso de reutilización sistemática de requisitos se tiene que tener en cuenta las relaciones entre requisitos de manera que cuando se reutilice un requisito, también se reutilicen aquellos con los que está relacionado y sean requeridos. Por tanto, hay que conocer la semántica de las relaciones a priori. En las transformaciones de modelos, la semántica de las relaciones también puede ser significativa, por ejemplo, los hijos de un requisito pueden representar los parámetros para la configuración de los dispositivos domóticos, lo que permite definir las reglas de transformación.

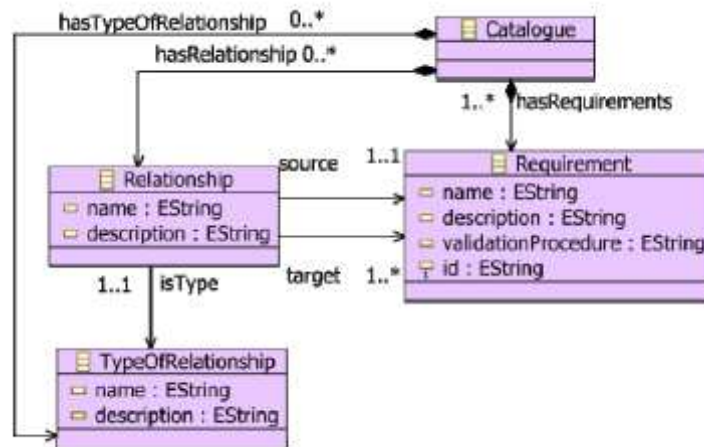


Figura 87 Metamodelo de requisitos propuesto por la Dr. Rosique [296].

- 2) **La herramienta, REMM-Studio***, que soporta la reutilización sistemática de requisitos, a partir de los requisitos del catálogo representados mediante modelos de requisitos conformes a REMM. REMM-Studio proporciona asistentes que facilitan la selección de los requisitos. En la propuesta de Rosique [296] esta selección se hace básicamente copiando y pegando.
- 3) **La definición de transformaciones m2m**, que permiten la generación de los modelos conformes al DSL *automáticamente* a partir del modelo de requisitos de la aplicación domótica. En la propuesta de Rosique [296] al copiar y pegar los requisitos, también se copian los fragmentos de modelos conformes al DSL relacionados con ellos explícitamente. Una vez obtenidos los modelos que

representan una aplicación domótica, se seguiría el proceso de la Figura 85 que llega hasta la generación de código. En consecuencia, en nuestra aportación se hace posible la obtención del código de la aplicación de forma automática a partir de la especificación de requisitos.

8.4. HAREL: un catálogo de requisitos reutilizables para los sistemas domóticos

Como se mencionó en la Sección 1.1.3.4, recientemente, se ha señalado a los lenguajes de especificación de requisitos (RSL, *Requirements Specification Language*) basados en un lenguaje natural restringido, como un punto de partida factible para la definición de transformaciones m2m a partir de requisitos textuales [368]. Un tipo de RSL restringido es el que se conoce como RSL basado en plantillas (*boilerplate-based RSL*) [122] que especifica plantillas de requisitos como una forma de estandarizar el lenguaje que se utiliza para la especificación de requisitos (Sección 1.1.1.1). La especificación de requisitos para una aplicación concreta, utilizando estas plantillas, se convierte en un proceso de selección de las plantillas más adecuadas y completar los datos para las partes de la plantilla que quedan sin especificar. Estas plantillas de requisitos, propuestas por Hull *et al.* [122], se corresponden con los requisitos parametrizados de REMM que se incluyen en los modelos de requisitos *para* reutilización (Sección 5.4). De manera que, el catálogo de requisitos reutilizables para el dominio de la domótica (representado mediante un modelo de requisitos conforme a REMM) se puede considerar como un RSL basado en plantillas que nos permite estandarizar la especificación de requisitos de los sistemas domóticos. Así, restringiendo la forma en la que se especifican los requisitos para este tipo de sistemas, es posible definir transformaciones de modelos que generen el modelo de la aplicación (conforme al DSL de domótica) a partir de los requisitos. A este lenguaje lo hemos denominado HAREL (*Home Automation Requirements Specification Language*), y es el nombre que recibe el catálogo de requisitos reutilizables para el dominio de los sistemas domóticos.

De acuerdo con el modelo de proceso ReToCo (*From Requirements To Code*) presentado en el Capítulo 6, la definición del modelo de requisitos que representa el catálogo HAREL, formaría parte del proceso de Ingeniería del Dominio. En concreto, es el entregable de tarea de *Creación del catálogo de requisitos reutilizables* (Sección 6.2.1.1). En esta tarea los miembros del grupo de investigación del DSIE han desempeñado el rol de *usuarios clave* o expertos del dominio, mientras que esta doctoranda ha jugado el rol de *analista o ingeniera de requisitos* (de acuerdo al modelo de roles de SIREN, Sección 4.2.3) y de *responsable del catálogo*. El catálogo de dominio HAREL, resultado de esta tarea, incluye los requisitos relacionados con el sistema de iluminación, el sistema de motorización (persianas y toldos), el sistema de climatización y el de seguridad (alarmas), que hemos considerado que son los requisitos básicos para la instalación de un sistema domótico. En las siguientes secciones presentamos el fragmento del catálogo de requisitos que representa cada uno de estos aspectos.

8.4.1. Requisitos del sistema de iluminación

La especificación de requisitos del sistema de iluminación (Figura 88) implica la definición de un número de puntos de luz (parámetro X de tipo Number) que pueden ser o no fluorescentes (parámetro TYPE de tipo KindOfLight). Estos puntos de luz pueden ser activados mediante diferentes mecanismos (parámetro MECHANISM de tipo LightMechanism): botón de pulsación (PUSH_BUTTON), detector de presencia (PRESENCE_DETECTOR) o sensor de iluminación (SUNRISE/SUNSET_SENSOR). Los mismos puntos de luz se pueden activar utilizando más de un mecanismo, por eso la cardinalidad del enumerado LightMechanism se ha definido como [1, 3].

Por otro lado, los puntos de luz pueden ser fijos o regulables. Esta configuración se determina por el parámetro ACTION (de tipo LightAction) que establece si las luces se van a encender y apagar o se van a regular. En el caso de que las luces sean regulables, se puede configurar (CHILD::OR) el tiempo para detectar una pulsación larga (que implica que se comiencen a regular las luces, Req_8) y cada cuanto tiempo se tiene que mandar el incremento de regulación de las luces (Req_9). Nótese que la configuración del Req_8 sólo tiene sentido si entre los mecanismos seleccionados para el control de los puntos de luz se encuentra el botón de pulsación.

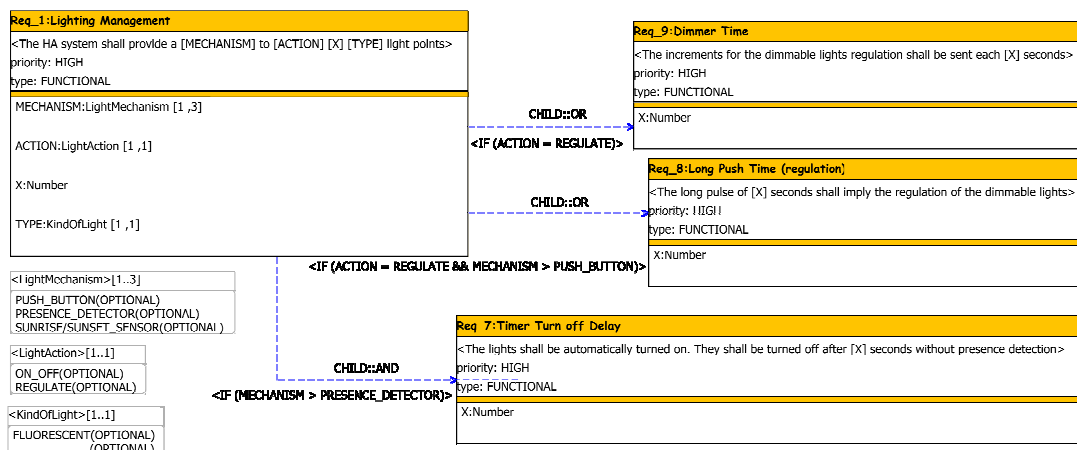


Figura 88 Modelo de requisitos para el sistema de iluminación.

Por último, en el caso de que entre los mecanismos seleccionados se encuentre el detector de presencia, las luces se encenderán automáticamente, pero es necesario (CHILD::AND) establecer el tiempo que debe pasar, desde que no se detecta presencia, hasta que se apagan las luces (Req_7).

8.4.2. Requisitos de motorización: persianas y toldos

Los requisitos de motorización tienen que ver con el control de dispositivos movidos por motores como las *persianas* y los *toldos*. Se controla fundamentalmente el

movimiento de subida y bajada, y se regula la posición de los dispositivos motorizados. También es posible controlar la parada de movimiento de los dispositivos.

En el caso de las persianas (Figura 89), se pueden instalar persianas normales (SHUTTER) o persianas tipo venecianas (VENETIAN_BLINDS). El tipo de persianas se establece mediante el parámetro BLINDS de tipo KindOfBlind. Aunque la mayoría de los dispositivos motorizados cuentan con la detección del final de carrera (subida completa), y calculan y guardan este tiempo la primera vez que se ponen en marcha, es posible que sea necesario configurar el tiempo de actuación sobre el motor para subida o bajada total. La configuración de este valor se recoge en el Req_10, definido como requisitos hijo opcional (CHILD : : OR).

Las persianas se pueden controlar utilizando diferentes mecanismos (parámetro MECHANISM de tipo BlindMechanism) como: botón de pulsación (PUSH_BUTTON), sensor de iluminación (SUNRISE/SUNSET_SENSOR) o un programador semanal (WEEK_TIMER). En el caso de seleccionar entre los mecanismos el botón de pulsación, se debe configurar (CHILD : : AND) el tiempo para detectar una pulsación larga (Req_11). Si el dispositivo detecta una pulsación corta la persiana sube o baja totalmente, mientras que si se detecta una pulsación larga la persiana sube o baja mientras esté pulsado el botón y al soltar para. Por otro lado, en el caso de seleccionar el programador semanal es necesario configurar los días de programación (Req_23) y las horas a las que se van a subir (Req_24) y bajar las persianas (Req_25).

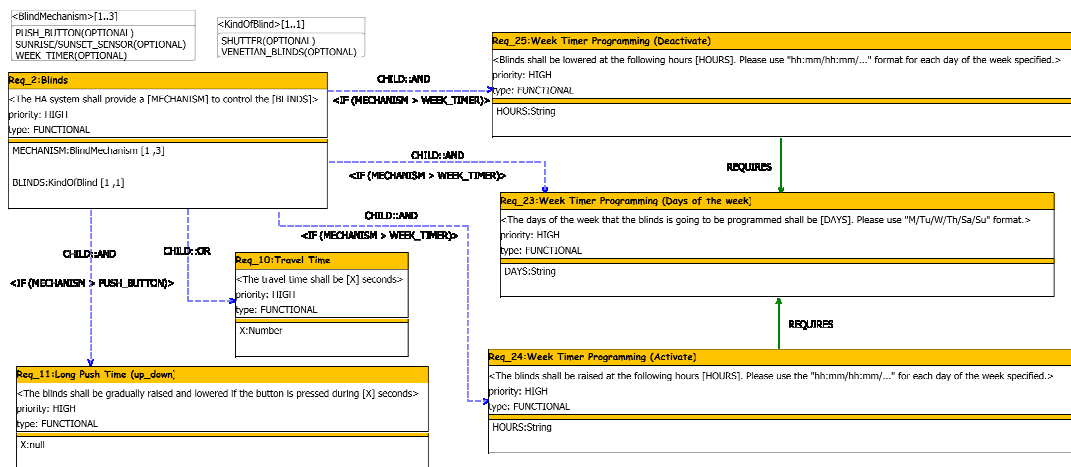


Figura 89 Modelo de requisitos para las persianas.

En el caso de los toldos (Figura 90), los mecanismos que se pueden utilizar para controlarlos son: botones de pulsación, detector de viento (WIND_DETECTOR), sensor de iluminación y el programador semanal. Como ocurría con las persianas, si se elige como mecanismo de control el programador semanal se debe configurar los días y horas para la subida y bajada del toldo.

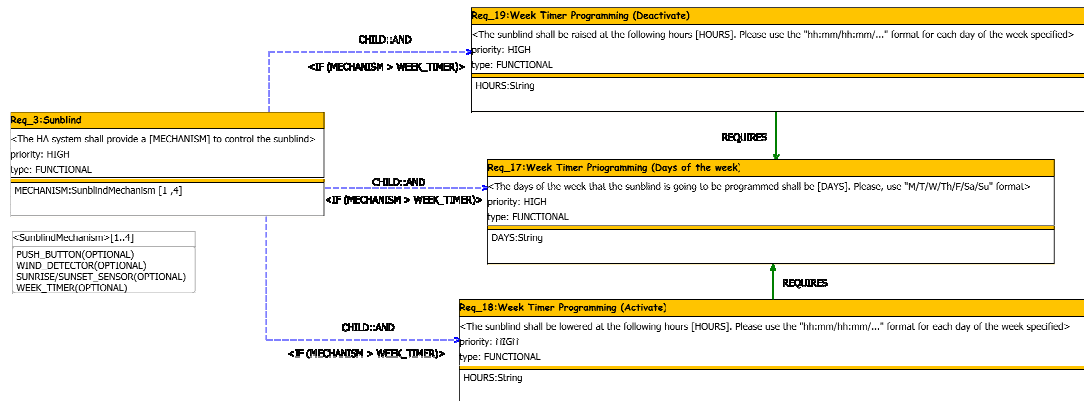


Figura 90 Modelo de requisitos para los toldos.

8.4.3. Requisitos del sistema de climatización

Para la especificación de requisitos del sistema de climatización (Figura 91) hemos considerado la posibilidad de instalar dos tipos de dispositivos (parámetro AC_DEVICE de tipo KindOfAC): SPLIT o FANCOIL. Un *split* es una unidad climatizadora autónoma que suele utilizarse en viviendas, pequeños comercios o despachos. Mientras que un *fancoil* es un sistema de climatización que combina circulación de agua y aire. Resulta ventajoso en edificios donde es preciso economizar el máximo de espacio, ya que suple a los sistemas centralizados que requieren de grandes superficies para instalar sus equipos. En la cubierta del edificio se instala la unidad evaporadora con central térmica, y en cada ambiente a acondicionar un aparato individual o fancoil. Sea cual sea el dispositivo que se instale, la regulación de la temperatura se hará utilizando un termostato en el que hay que configurar la variación de temperatura permitida cuando el dispositivo se encuentra en *standby* (Req_12) y la variación de temperatura nocturna (Req_13).

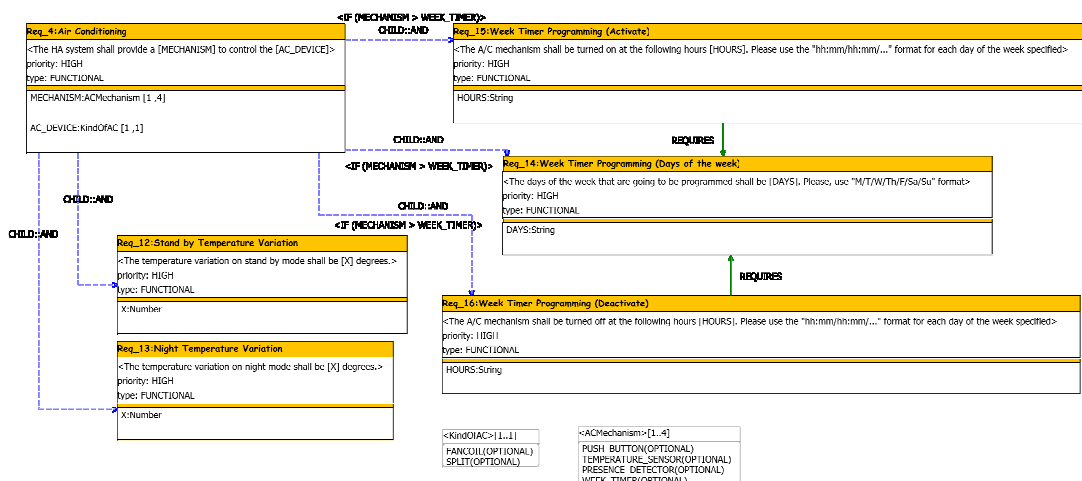


Figura 91 Modelo de requisitos para el sistema de climatización.

Estos dispositivos se pueden controlar utilizando diferentes mecanismos como son: botones de pulsación (para subir y bajar la temperatura), sensores de temperatura (TEMPERATURE_SENSOR), detector de presencia y programador semanal. Al igual que ocurría en el caso de los toldos y las persianas, en el caso de seleccionar el programador semanal, se debe configurar el día y hora para el encendido y apagado del dispositivo de climatización.

8.4.4. Requisitos del sistema de seguridad

La especificación de requisitos para el sistema de seguridad (Figura 92) considera que pueda estar compuesto por diferentes tipos de alarmas (parámetro TYPE de tipo KindOfAlarm): intrusión (INTRUSION), humo (SMOKE), incendio (FIRE) e inundación (WATER). En el caso de la alarma de inundación, se puede solicitar la instalación de una válvula que permita cortar el agua (Req_6). En cualquiera de las alarmas también se puede solicitar que se informe al usuario de que la alarma ha saltado enviando un mensaje al teléfono móvil (Req_20). En ese caso se debe establecer el número de teléfono destino (Req_21) y el mensaje que se va a mandar (Req_22). El requisito Req_5 se tendrá que reutilizar, e instanciar convenientemente, una vez por cada tipo de alarma que se desee instalar.

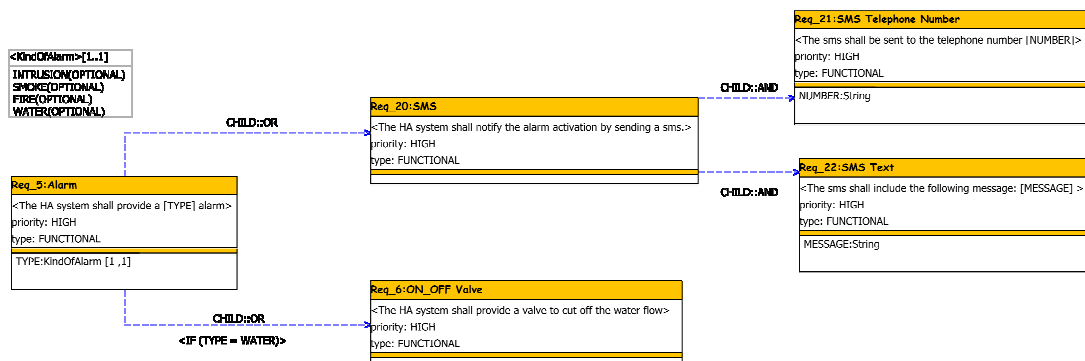


Figura 92 Modelo de requisitos para el sistema de seguridad.

8.5. Especificación de requisitos para una nueva aplicación

Una vez que se ha definido el catálogo de dominio HAREL (que contiene el modelo de requisitos *para* reutilización, conforme a REMM, que representa la especificación de requisitos reutilizables para los sistemas domóticos) se pueden definir los requisitos para una nueva aplicación domótica siguiendo el proceso $SIREN_{MD}$ en su faceta *con* reutilización (Sección 6.2.2). Además, para facilitar la selección e instanciación de los requisitos reutilizados, podemos utilizar las funcionalidades que nos ofrece la herramienta REMM-Studio⁺ (Sección 7.3.3). El resultado del proceso será un nuevo modelo de requisitos, conforme a REMM, que representa la especificación de requisitos de una nueva aplicación.

Así, por ejemplo, en la Figura 93 se muestra un ejemplo de un modelo de requisitos para una nueva aplicación domótica que se ha obtenido reutilizando los requisitos definidos en el catálogo HAREL. Se puede ver en la figura que esta nueva aplicación incluye los requisitos relacionados con los sistemas de iluminación y climatización. Se ha especificado que se requiere la instalación de tres puntos de luz regulables (reutilización e instanciación del requisito Req_1 en Figura 88) que se controlarán utilizando todos los mecanismos posibles (botón, detector de presencia y sensor de iluminación). Además, se requiere que a los cinco segundos de no detectar presencia, las luces se apaguen automáticamente. Por otro lado, se requiere la instalación de un fancoil (reutilización e instanciación del requisito Req_4 en Figura 91) que se va a controlar mediante botones, sensor de temperatura, detector de presencia y programador semanal. La variación de temperatura nocturna y en *standby* debe ser de 2 grados, y se encenderá y apagará automáticamente los fines de semana a las nueve de la mañana y las doce de la noche, respectivamente.

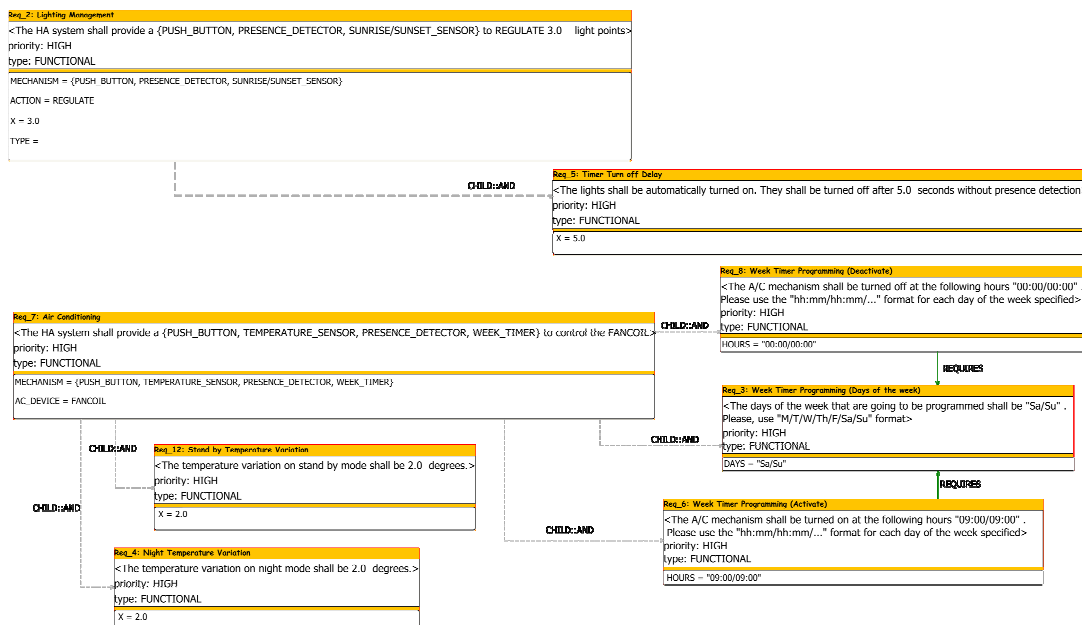


Figura 93 Modelo de requisitos para una nueva aplicación domótica.

8.6. Modelos domóticos generados automáticamente a partir de los requisitos

Siguiendo el modelo de proceso ReToCo, una vez definidos los requisitos de una nueva aplicación, se puede generar automáticamente los modelos conformes al DSL de domótica (Sección 6.2.2.2) ejecutando las transformaciones m2m definidas durante la fase de Ingeniería del Dominio. Merece la pena incidir en que estas reglas de transformación encapsulan el conocimiento de los expertos del dominio que, en base a su experiencia, determinan las unidades funcionales más adecuadas, los controladores

necesarios para su funcionamiento y los servicios que se requieren. Este conocimiento no forma parte de la especificación de requisitos, de manera que hemos conseguido elevar el nivel de abstracción definiendo el sistema de una forma más cercana al usuario no experto. En las siguientes subsecciones detallamos los modelos generados para cada uno de los sistemas especificados en el catálogo HAREL: iluminación (Sección 8.6.1), motorización (Sección 8.6.2), climatización (Sección 8.6.3) y seguridad (Sección 8.6.4). En cada caso destacaremos las decisiones de diseño, encapsuladas en las reglas de transformación, que han dado lugar a los modelos de domótica.

Como decíamos anteriormente, cuando los expertos en el dominio de los sistemas domóticos modelan una nueva aplicación desde cero, utilizando el DSL Habitation, toman decisiones de diseño que no se encuentran especificadas en los requisitos, sino que forman parte de su conocimiento y experiencia. Parte de estas decisiones se han implementado en las reglas de transformación. Las principales decisiones de diseño se pueden categorizar como sigue:

- Selección de los conceptos del DSL. Crear las unidades funcionales, enlaces y servicios necesarios para implementar los requisitos del cliente.
- Adopción del modelo arquitectural. Por ejemplo, arquitectura centralizada vs. distribuida. Las reglas de transformación actuales implementan una arquitectura distribuida en la que cada unidad funcional se conecta a un controlador independiente (SWI, DMI o DMO en Figura 94). En una arquitectura centralizada todas las unidades funcionales estarían conectadas a un único panel de control. El problema que tiene este tipo de arquitectura es que si falla el panel de control, todas las unidades funcionales conectadas a él (luces, aire acondicionado, etc.) dejarían de funcionar. Una versión simplificada de esta idea sería conectar todos los puntos de luz (LDM en Figura 94) al mismo controlador (DMO en Figura 94). Como consecuencia de esta configuración, los tres puntos de luz se encenderían y apagarían al mismo tiempo. Si cada punto de luz se conecta a un controlador distinto, cada punto de luz se puede gestionar de manera independiente.
- Optimización de recursos. Esto es, utilizar el mínimo número de unidades funcionales. Por ejemplo, un mismo detector de presencia puede servir para controlar tanto el sistema de iluminación como el encendido y apagado del aire acondicionado. La optimización de recursos no es soportada actualmente por las reglas de transformación y debe realizarse manualmente utilizando el editor del DSL.
- Configuración del modo de operación de las unidades funcionales. En el DSL Habitation el modo de operación de los dispositivos representados por las unidades funcionales se modela mediante los parámetros. El modo de operación se corresponde, frecuentemente, con decisiones que hay que tomar a nivel de electrónica, como el modo de operación “rising edge” para el controlador SWI. Esta configuración tiene que llevarla a cabo un experto en el dominio. Algunos de los parámetros se pueden configurar durante la especificación de requisitos, mediante requisitos hijos (Req_7, Req_8 y Req_9 en Figura 88, Req_10 y Req_11 en Figura 89, Req_12 y Req_13 en Figura 91 y Req_21 y Req_22 en Figura 92), pero para otros parámetros se establece el valor por

defecto durante el proceso de transformación y deben ser revisados durante el diseño de la aplicación. Por ejemplo, en la Figura 94 se establece el valor ON para el parámetro SWISentOnRisingEdge de la unidad funcional que representa el controlador de entrada SWI.

8.6.1. Modelos domóticos para el sistema de iluminación

La Figura 94 muestra el modelo de domótica generado automáticamente a partir de los requisitos de iluminación de la Figura 93, en el que se especifica la necesidad de instalar tres puntos de luz regulables que se puedan controlar mediante un botón de pulsación, un detector de presencia y un sensor de iluminación. En la Figura 95, se muestra el resultado de la generación del modelo de domótica para los mismos requisitos cambiando la acción de regulación (ACTION = REGULATE en Req_2 de Figura 93) por la de encendido y apagado (ACTION = ON_OFF). A continuación vamos a detallar las decisiones de diseño encapsuladas en las reglas de transformación para la generación de dichos modelos. Éstas son las siguientes:

- D.1. Los *mecanismos* seleccionados establecen las unidades funcionales que se deben crear para el control de los puntos de luz: PB para el botón de pulsación, SS para el sensor de iluminación, PIR para el detector de presencia.
- D.2. La *acción* y el *tipo* seleccionados determinan las unidades funcionales correspondientes a los puntos de luz que serán fijos (LO o FLO, si es fluorescente), si la acción es encender y apagar, o puntos de luz regulables (LDM FLD, si es fluorescente), si la acción es regular.
- D.3. Los valores de los parámetros de las unidades funcionales se especifican en los requisitos hijos. En el caso del modelo de requisitos del sistema de iluminación (Figura 88) el requisito Req_7 recoge el valor del parámetro TMtemp del temporizador (TM), el requisito Req_8 especifica el valor para el parámetro DMOdimTime del controlador del punto de luz regulable (DMO), y el requisito Req_9 establece el valor del parámetro DMI longPushTime del controlador del botón de pulsación (PB). El valor del parámetro TMType del temporizador (TM) es siempre OffDelay, puesto que el requisito especifica que hay un retardo de X segundos para apagar las luces desde que no se ha detectado presencia. El resto de parámetros toma los valores por defecto.
- D.4. Cada mecanismo y punto de luz tiene asociado un *controlador de entrada*, en el caso de los mecanismos (SWI, DMI), o de *salida* (SWO, DMO) en el caso de los puntos de luz. Los controladores de las luces fijas son conmutadores (SWI y SWO) y se conectan mediante enlaces con el servicio switch (Figura 95), puesto que lo que tiene que hacer el dispositivo es cambiar la señal para que las luces pasen de estar encendidas a estar apagadas, y viceversa. Los controladores de las luces regulables son DMI y DMO (Figura 94), y se conectan mediante dos enlaces, uno con la definición de servicio dimmer, para regular la intensidad de la luz, y un enlace con la definición de servicio switch, para encender y apagar. El número de enlaces entre los controladores (SWI - - SWO,

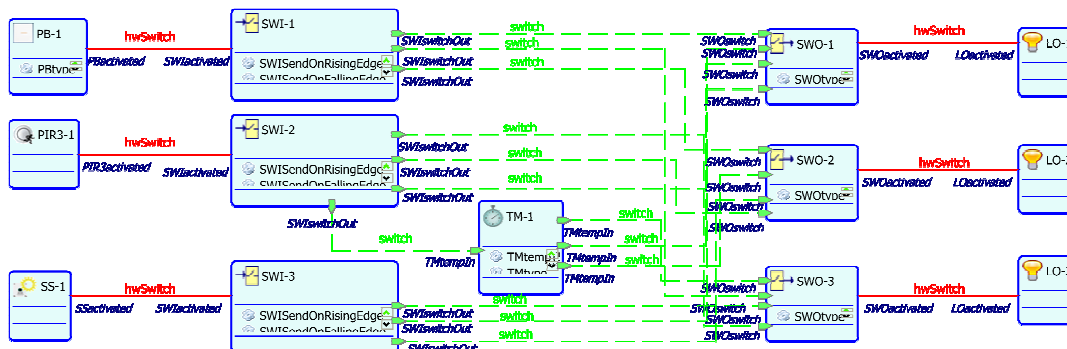


Figura 95 Modelo generado para la instalación de tres puntos de luz fijos.

Los modelos generados para las luces fluorescentes fijas y regulables son idénticos a los de la Figura 95 y Figura 94, con la única diferencia de las unidades funcionales generadas para los puntos de luz (Figura 96). En el caso de puntos de luz fijos (Figura 95), la unidad funcional correspondiente es FLO en lugar de ser LO, y para los puntos de luz regulables (Figura 94), la unidad funcional generada será FLD en lugar de LDM.

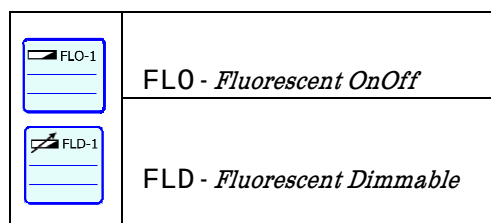


Figura 96 Unidades funcionales para los puntos de luz fluorescentes.

8.6.2. Modelos domóticos para el sistema de motorización: toldos y persianas

Los sistemas de motorización comprenden todos aquellos dispositivos que se mueven utilizando un motor, como las persianas y los toldos. La especificación de requisitos para estos dos elementos motorizados se muestra en la Figura 89 y la Figura 90, respectivamente. Las decisiones de diseño implementadas en las reglas de transformación, permiten generar los modelos de domótica que implementan dichos requisitos (en realidad, una instanciación de los requisitos especificados en el catálogo HAREL). La Figura 97 muestra el modelo generado para el requisito que especifica la instalación de una persiana veneciana que se debe controlar mediante un botón de pulsación, un sensor de iluminación y un programador semanal. La Figura 98 muestra el modelo de domótica que representa el requisito que establece la instalación de un toldo que tiene que ser controlado por un botón de pulsación, un detector de viento, un sensor de iluminación y un programador semanal. Las decisiones de diseño son:

- D.7. Los *mecanismos* seleccionados establecen las unidades funcionales que se deben crear para el control de los elementos motorizados: sensor de iluminación (SS), detector de viento (WiD), programador semanal (WT) y, en el caso de

seleccionar el botón de pulsación, se tienen que generar dos unidades funcionales, una que representa el botón de subida (PB-Up) y otra para el botón de bajada (PB-Down).

- D.8. El *tipo* de las persianas determina la unidad funcional que representa el elemento motorizado: persiana normal (SHM, *Shutter*) o veneciana (VBM, *Venetian Blind*). En el caso del toldo, la unidad funcional que se crea siempre es la misma (SBM, *Sunblind*).
- D.9. Las unidades funcionales que representan los controladores de los motores son MTI (*Motorization In*) y MTO (*Motorization Out*).
- D.10. Los valores de los parámetros, de las unidades funcionales creadas, se especifican en los requisitos hijos. En la Figura 89 (modelo de requisitos de las persianas) el Req_10 especifica el valor del parámetro MTOtravelTime del controlador de salida del motor (MTO) y el Req_11 define el valor del parámetro MTIlongPushTime del controlador de entrada del motor (MTI). El programador semanal (WT) incluye tres parámetros WTweekDays, WTONtime y WToffTime, cuyos valores se especifican en los requisitos Req_23, Req_24 y Req_25.
- D.11. Los controladores del motor se conectan mediante un enlace con la definición de servicio MOVE, que permite el movimiento del elemento motorizado.
- D.12. Los mecanismos seleccionados para el control del elemento motorizado se conectan al controlador de entrada del motor (MTI) mediante dos enlaces tipo canal con la definición de servicio hwSwitch, uno para mandar la señal de subida y el otro para mandar la señal de bajada.
- D.13. El elemento motorizado se conecta al controlador de salida del motor (MTO) al menos mediante dos enlaces con la definición de servicio hwSwitch, para subir y bajar el elemento. En el caso de que el elemento motorizado admita giros, como es el caso de las lamas de las persianas venecianas y del toldo, hay que añadir un tercer enlace con la definición de servicio hwMove. No es necesario añadir otro enlace entre el mecanismo y el motor (MTI) porque el giro tiene lugar cuando se cambia de sentido (de arriba a abajo o de abajo a arriba), por ejemplo, si se está subiendo la persiana utilizando los botones, con la primer pulsación del botón que cambia el sentido del movimiento (bajada en este caso) se produce la rotación de la persiana, si se sigue pulsando al botón de bajada es cuando se inicia el descenso de la persiana. Con el resto de mecanismos el giro es imperceptible, puesto que el movimiento se realiza justo después del giro.

El modelo generado para el caso de las persianas normales es igual que el de la Figura 97, con la diferencia de la unidad funcional que representa el elemento motorizado, que será SHM en lugar de VBM, y que el controlador del motor de salida (MTO) se conecta al SHM únicamente con dos enlaces, para mandar las señales de subida y bajada de la persiana, dado que una persiana no puede girar.

8.6.3. Modelos domóticos para el sistema de climatización

El modelo domótico generado automáticamente a partir del requisito Req_7 en la Figura 93 es el que se muestra en la Figura 99. En la instanciación del requisito Req_4 del catálogo HAREL (Figura 91) se ha seleccionado la instalación de un *fancoil* que va a ser controlado utilizando todos los mecanismos posibles: botones de pulsación, sensor de temperatura, detector de presencia y programador semanal. El modelo generado para un *split*, en lugar de un *fancoil*, sería igual simplemente sustituyendo la unidad funcional que representa el dispositivo de climatización. Las decisiones de diseño implementadas en las reglas de transformación son:

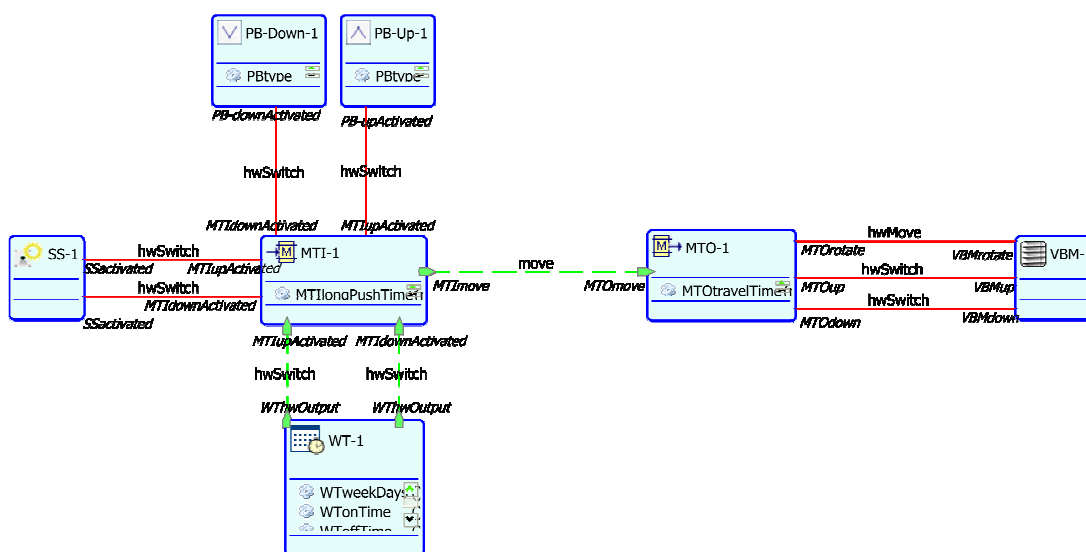


Figura 97 Modelo domótico para la instalación de una persiana veneciana.

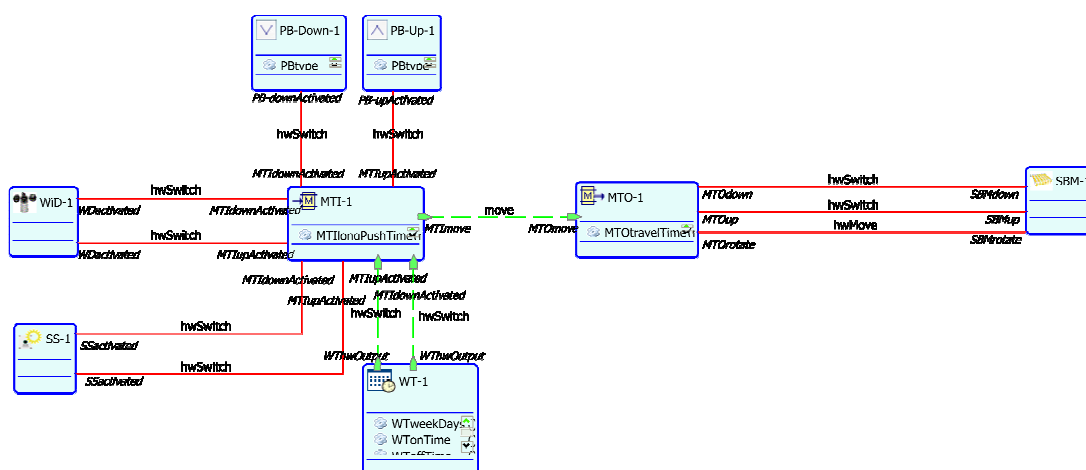


Figura 98 Modelo domótico para la instalación de un toldo.

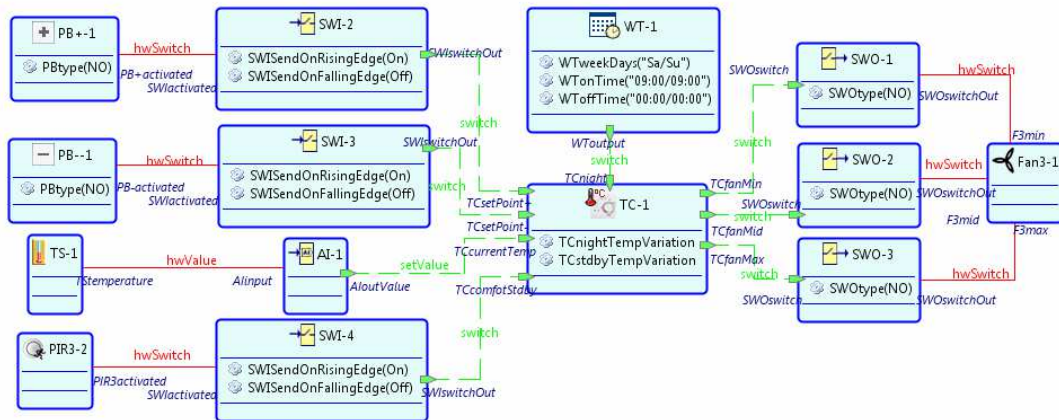


Figura 99 Modelo para la instalación de un *fancoil* como dispositivo de climatización.

- D.14. El *dispositivo* de aire acondicionado seleccionado determina la unidad funcional que se controla en el modelo de climatización: Fan3 para el *fancoil* y SAA para el *split*.
- D.15. Los *mecanismos* seleccionados establecen las unidades funcionales que se van a crear para controlar el dispositivo de climatización: TS para el sensor de temperatura, PIR3 para el detector de presencia, WT para el programador semanal y, en el caso de seleccionar como mecanismo el botón de pulsación, se tienen que crear dos unidades funcionales destinadas a subir (PB+) y bajar la temperatura (PB-).
- D.16. El detector de presencia no requiere de la instalación de un temporizador, como en el caso del sistema de iluminación (Figura 94), puesto que funciona con las temperaturas de confort establecidas.
- D.17. El sistema de climatización requiere la creación de una unidad funcional adicional, no especificada en los requisitos, que representa el *termostato* (TC). La definición de esta unidad funcional incluye dos parámetros `TCstdbyTempVariation` y `TCnightTempVariation` cuyos valores se establecen al instanciar los requisitos Req_12 y Req_14, respectivamente (Figura 91).
- D.18. Los controladores de los botones y del detector de presencia son conmutadores (SWI), y se conectan mediante un enlace tipo canal con la definición de servicio `hwSwitch`. El sensor de temperatura necesita entrada analógica (IA) que será la encargada de enviar al termostato (TC) el valor de la lectura del sensor. La conexión entre el sensor de temperatura y la entrada analógica se establece mediante un enlace, también tipo canal, con la definición de servicio `hwValue`.
- D.19. Los controladores de los mecanismos seleccionados se conectan al termostato mediante enlaces con el servicio `switch`. El programador semanal no requiere controlador y se conecta directamente al termostato mediante un enlace con el servicio `switch`.

- D.20.** El termostato requiere tres conmutadores de salida (SWO) para gestionar las velocidades del dispositivo de aire acondicionado (mínima, media y máxima). El termostato se conecta a cada uno de los conmutadores mediante enlaces con el servicio Switch. Y los conmutadores, a su vez, se conectan con el dispositivo de aire acondicionado (*fancoil* o *split*) mediante enlaces tipo canal con el servicio hwSwitch.

8.6.4. Modelos domóticos para el sistema de seguridad

La Figura 100 muestra todos los modelos domóticos que representan la instalación de los distintos tipos de alarmas contemplados en la especificación de requisitos: humo, incendio, inundación e intrusión. Cada modelo se corresponde con una de las posibles instanciaciones del requisito Req_5 de la Figura 92, en el caso de la alarma de inundación se requiere la instalación de una válvula para cortar el agua, y en el caso del detector de presencia se solicita el envío de un mensaje al teléfono móvil avisando de la intrusión. Las decisiones de diseño implementadas en las reglas de transformación son:

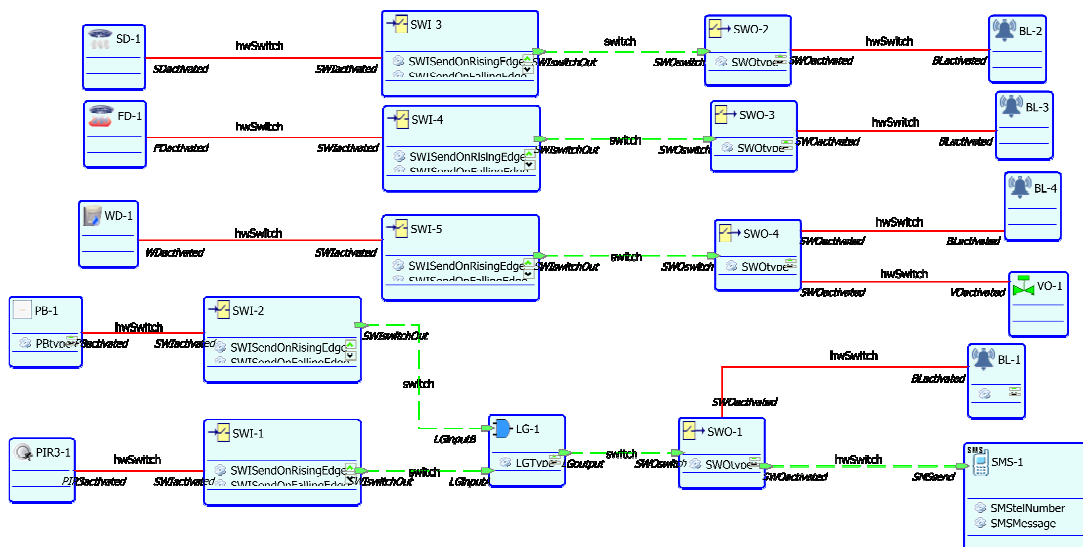


Figura 100 Modelos para la instalación de las alarmas: humo, incendio, inundación e intrusión.

- D.21.** Cualquier instanciación del requisito genera la unidad funcional de la alarma que se puede activar (BL).
- D.22.** El *tipo* de alarma (humo, incendio, inundación o intrusión) determina la unidad funcional que representa el dispositivo que puede activar la alarma: SD para el detector de humo, FD para el detector de fuego, WD para el detector de agua y PIR3 para el detector de presencia.
- D.23.** Todos los dispositivos de activación de la alarma utilizan un conmutador (SWI) al que se conectan mediante un enlace tipo canal con el servicio hwSwitch.

- D.24. La alarma (BL) se conecta a un conmutador (SWO) mediante un enlace tipo canal con el servicio hwSwitch.
- D.25. La instalación de una alarma de intrusión requiere la instalación de un botón de pulsación (PB) para activar la alarma. Los conmutadores (SWI) del botón y del detector de presencia se conectan a una puerta lógica (LG) tipo AND, de manera que sonará la alarma si se ha activado y se ha detectado presencia.
- D.26. La creación, opcional, de una unidad funcional que representa una válvula (VO) para cortar el agua en caso de inundación se recoge en el requisito hijo Req_6 (Figura 92).
- D.27. La creación, opcional, de la unidad funcional que representa el envío de un mensaje a un teléfono móvil (SMS) se especifica en el requisito hijo Req_20 (Figura 92). Los valores de los parámetros de la unidad funcional SMS, SMS telNumber y SMSMessage se especifican al instanciar los requisitos Req_21 y Req_22, respectivamente.

8.7. Generación de código a partir de los modelos de domótica

Una vez definido el modelo de la aplicación conforme al DSL de domótica, de acuerdo con el enfoque de modelado específico de dominio, se puede generar el código de la aplicación a partir de dicho modelo. Este proceso de transformación, de modelos a código, forma parte de la tesis doctoral de la Dra. Rosique [296], que en esta sección resumimos para dar una visión integral de la propuesta de desarrollo de sistemas domóticos que hacemos en esta tesis doctoral, desde los requisitos hasta el código de la aplicación (Figura 101).

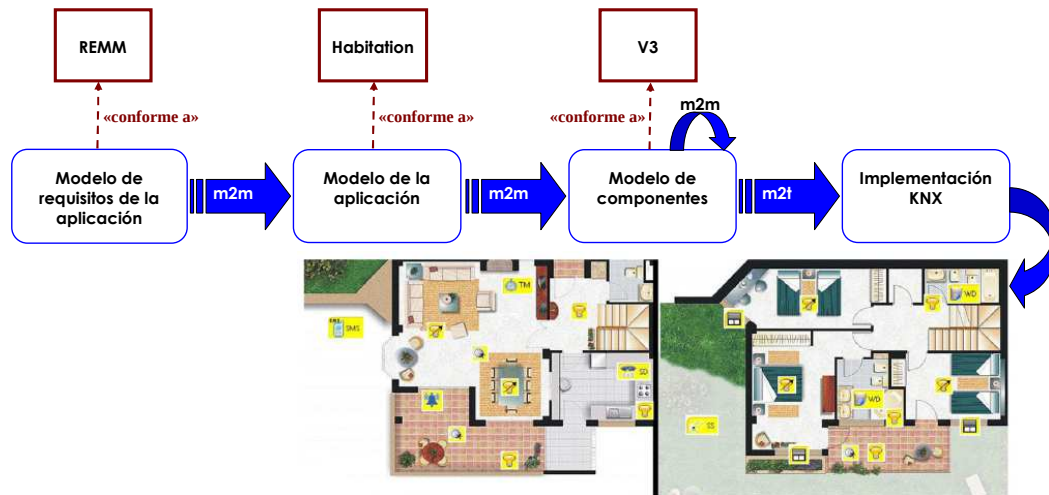


Figura 101 Esquema de transformaciones de modelos desde los requisitos hasta el código.

En la propuesta de Sánchez *et al.* [301], el código no se genera directamente a partir del modelo de la aplicación (conforme al DSL Habitation), sino que se obtiene un modelo

intermedio, concretamente, un modelo de componentes basado en UML (conforme al metamodelo V3 [21]), mediante transformaciones m2m. Según los autores, el uso de un modelo intermedio permite la integración con otros tipos de aplicaciones, como robótica o visión por computador. Este modelo de componentes inicial es un modelo de componentes independiente de la plataforma y es necesario refinarlo, también mediante transformaciones m2m, para completar los componentes generados con información específica de la tecnología seleccionada, en este caso KNX. Finalmente, a partir del modelo de componentes específico de la plataforma, se genera el código de la aplicación, mediante transformaciones m2t (Figura 101).

La definición de las transformaciones m2m que tiene como entrada el modelo de la aplicación y obtiene como resultado el modelo de componentes, se ha implementado utilizando un enfoque basado en gramáticas de grafos [297] (en particular, el *plug-in EMT, EMF Model Transformation Framework* [34], para el entorno de desarrollo Eclipse). Cada una de las reglas de transformación incluidas en la definición está compuesta por (Figura 102): (1) *lado izquierdo* (LHS), se especifican las precondiciones que deben cumplirse para que se aplique la regla de transformación, (2) *lado derecho* (RHS), muestra el grafo resultado de la transformación, y, opcionalmente, (3) *condición de no aplicación* (NAC), establece qué no deben cumplirse para que se pueda aplicar la regla de transformación. Tanto el LHS y RHS son grafos, que suelen contener elementos de los modelos de entrada y salida de la transformación. En nuestro caso, el modelo de entrada es el DSL de domótica, y conceptos como unidad funcional o servicio forman parte de Habitation. El modelo de salida es el modelo de componentes, y conceptos como componente simple o puerto forman parte de V3. La inclusión de elementos del modelo de salida en el LHS asegura que las transformaciones se ejecuten en la secuencia correcta. La regla de la Figura 102 establece que cuando se encuentre un servicio en el modelo de la aplicación (LHS) se debe transformar en puertos, interfaces y servicios del modelo de componentes (RHS). Esta regla no debe aplicarse si la transformación ya se ha ejecutado (NAC).

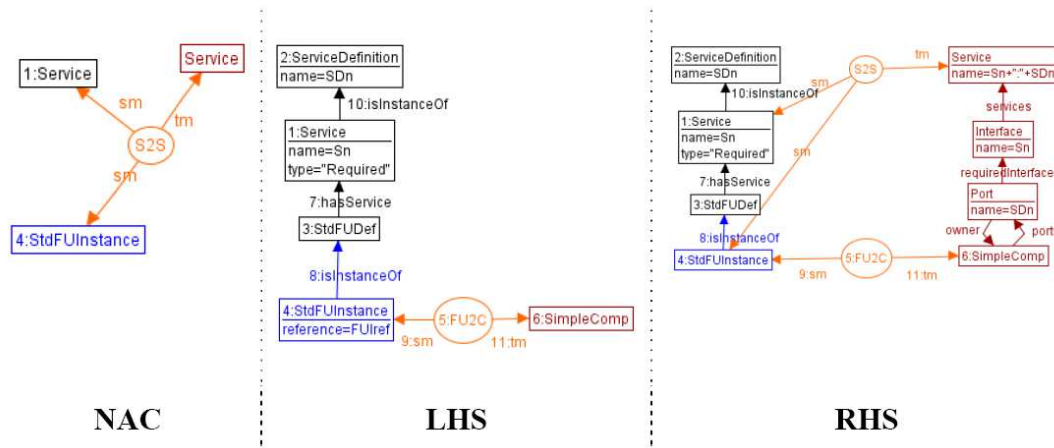


Figura 102 Ejemplo de regla de transformación del modelo del DSL al modelo de componentes.

Finalmente, a partir del modelo de componentes específico de la plataforma, se genera el código de la aplicación, mediante transformaciones m2t implementadas en

JET [5]. Para la generación del código había dos alternativas posibles: 1) generar directamente el código para la programación de los dispositivos, o 2) reutilizar las herramientas disponibles en el mercado para la tecnología seleccionada. Sánchez *et al.* [301] optaron por la segunda de las alternativas, de manera que, a partir de los modelos de componentes, se generan macros en VBScript que luego se pueden editar y ejecutar en la herramienta comercial ETS (*Engineering Tool Software*). Por tanto, la herramienta ETS es el soporte para la ejecución del código generado (*macros*), obteniendo así la configuración final que se implementa en cada uno de los dispositivos domóticos implicados en la aplicación.

8.8. Encuesta de validación

Las encuestas se utilizan con frecuencia por los investigadores en Ingeniería del Software para llegar a comprender mejor algunos aspectos complejos y dar soporte a una toma de decisiones efectiva [162]. En nuestro caso, hemos planificado una encuesta con el fin de caracterizar las actitudes u opiniones de los expertos en el desarrollo del software acerca de la aproximación que estamos proponiendo en esta tesis doctoral. Hemos llevado a cabo una encuesta a través de Internet, utilizando LimeSurvey [8], una aplicación gratuita de código abierto. Para el diseño de la encuesta hemos seguido los consejos de Kitchenham y Pfleeger en cuanto a la elección del método de muestreo y la elaboración y evaluación de los cuestionarios [165-167, 267]. El objetivo de la encuesta era contestar a la siguiente pregunta de investigación:

¿Es adecuada nuestra propuesta de ingeniería de requisitos dirigida por modelos para incrementar la calidad de los modelos de sistemas domóticos y mejorar tanto el proceso de desarrollo, en general, como el de IR, en particular?

Existen diferentes formas de medir actitudes, pero las más populares son las *escalas tipo Likert* [191]. Se trata de un tipo de escala en la que las sentencias sobre algún tema se siguen de una serie de números, entre los que los encuestados tienen que elegir, para mostrar hasta qué punto están de acuerdo con dichas sentencias. Hemos utilizado una escala de Likert de 5 puntos en el que los valores de las respuestas varían entre “1. Muy en desacuerdo (*strongly disagree*)” y “5. Muy de acuerdo (*strongly agree*)”. La ventaja de utilizar este tipo de respuestas estandarizadas es que permite reducir el tiempo de respuesta [166], lo cual constituye un importante factor a tener en cuenta para animar a los encuestados a participar.

Una vez establecidos los objetivos de la encuesta, no sólo es importante elegir el tamaño de la muestra, de manera que sea realmente representativa de la población global, sino que también es necesario asegurarse que un número suficiente de los encuestados respondan a la encuesta, para conseguir resultados significativos [165]. El perfil ideal de nuestros encuestados son los desarrolladores de sistemas domóticos con experiencia en, o al menos conocimiento de, la propuesta de DSDM. Dado que no existe un censo en el que se haya registrado a los individuos pertenecientes a esta población objetivo, no conocemos la población completa a partir de la cual se debe seleccionar la muestra, y, por tanto, hemos utilizado el *método de muestreo de la bola de nieve* [289]

para crear la muestra. En primer lugar, seleccionamos un grupo con experiencia muy relacionada con nuestra propuesta, incluyendo tanto desarrolladores de sistemas domóticos como investigadores tanto del área de los sistemas domóticos como del DSDM. A continuación, enviamos un correo a los candidatos que habíamos seleccionado para explicarles el motivo de la encuesta y solicitarles que nos proporcionaran otras personas, relacionadas con el tema, que podrían estar interesadas en participar. Como resultado de este proceso de *mailing*, 40 personas accedieron a participar en la encuesta. Era necesario utilizar un método de muestreo no probabilístico, como es el de la bola de nieve, porque de esta forma podíamos estar seguros que los candidatos entenderían las preguntas y pensamos que podrían estar motivados porque se les invitaba a conocer, de primera mano, una nueva propuesta relacionada con su trabajo. Es necesario destacar que, puesto que no conocemos la población total de antemano, no pretendemos generalizar los resultados del estudio a la población de la cual se ha tomado la muestra, sino simplemente describir los datos obtenidos a partir de la encuesta.

El motivo por el que se han seleccionado tanto expertos pertenecientes al dominio de los sistemas domóticos como al DSDM es que cada grupo puede proporcionarnos interesante retroalimentación (*feedback*) desde diferentes puntos de vista: 1) los expertos en domótica como usuarios potenciales de la propuesta y 2) los expertos en DSDM como tecnólogos. Por un lado, los expertos en domótica utilizan herramientas para el desarrollo de los sistemas domóticos. La mayoría de estas herramientas están muy próximas a la tecnología utilizada para el desarrollo de los sistemas [148], como ya comentábamos al principio de este capítulo. En este caso, el *feedback* obtenido por este grupo de expertos nos permitirá evaluar la viabilidad de nuestra propuesta en comparación con sus procesos de desarrollo actuales en el dominio de los sistemas domóticos. Por otro lado, dada la amplia experiencia de algunos grupos de investigación en el desarrollo de diferentes entornos utilizando la tecnología de DSDM, el *feedback* obtenido por estos expertos nos permitirá reforzar las decisiones tomadas desde el punto de vista tecnológico.

Para favorecer que la encuesta fuese contestada y devuelta, tomamos las siguientes decisiones conducentes a mejorar el ratio de respuesta:

- La encuesta era anónima, no sólo para favorecer la participación, sino también para reducir la “presión de la amistad” si el encuestado conocía a alguno de los autores de la propuesta.
- Enviamos una invitación a los candidatos vía e-mail para motivarles a participar en la encuesta. En este e-mail explicábamos el objetivo de la encuesta, el número y tipo de preguntas que incluía y también el tiempo estimado para contestarla.
- Enviamos un recordatorio después de que hubiera pasado una semana desde la invitación a participar, y también tres días antes de que cerrásemos la encuesta.
- Validamos el cuestionario, siguiendo las indicaciones de Kitchenham y Pfleeger [167], para asegurarnos de que las preguntas eran simples, no ambiguas, comprensibles, neutrales (en el sentido de que la forma en la que estaban escritas no influían en lo que el encuestado pensaba sobre el tema), y que el orden de las preguntas no influía en las respuestas.

Finalmente, el 60% de las personas que inicialmente estaban interesadas en participar completaron la encuesta. Las personas encuestadas proceden de alguno de los

grupos de investigación que forman parte de la red de investigación oficial sobre DSDM y también de la industria de domótica. Dado que el origen era tanto español como francés, el cuestionario se desarrolló en inglés, para facilitar el acceso a todos los implicados. Al comienzo de la encuesta se solicitaba el perfil del participante como: A1- Desarrollador de sistemas domóticos sin conocimientos de DSDM (17% de los participantes), A2- Desarrollador de sistemas domóticos con experiencia en DSDM (25%). A3- Desarrollador de sistemas domóticos con conocimiento de DSDM (8%). A4- Desarrollador de software dirigido por modelos sin conocimientos acerca de los sistemas domóticos (50%). El perfil de los participantes se muestra en la Tabla 46. Para poder comparar los resultados, hemos dividido la muestra en dos grupos: personas con experiencia en sistemas domóticos (A1, A2 y A3) y personas con experiencia sólo en DSDM (A4). La muestra incluye un 50% de encuestados de cada categoría. La hipótesis inicial antes de comparar ambos perfiles es que la propuesta podría resultar más útil a los desarrolladores de sistemas domóticos dado que los expertos en DSDM podrían encontrar la propuesta simplemente como una aplicación de las técnicas del DSDM para un dominio específico.

Dado que los participantes estaban geográficamente dispersos, era difícil preparar un seminario presencial para explicar la propuesta y las herramientas de soporte, y en el que los participantes pudieran usar las herramientas y resolver las dudas que les surgieran. Para solucionar este problema, preparamos una *demo*, o presentación multimedia (<http://dis.um.es/~bmoros/results/demo/HAdemo.htm>), en el que se mostraba un caso de estudio con el que se ilustraba la puesta en práctica de la propuesta. Más concretamente, la demo comienza con una breve introducción en la que se explica los objetivos y la motivación de la propuesta, se resume el proceso de desarrollo (ReToCo) y se explican los beneficios esperados. A continuación se muestra el modelo de requisitos que representa el catálogo de requisitos reutilizables para los sistemas domóticos. Después, se ilustra cómo definir los requisitos de una nueva aplicación reutilizando los requisitos del catálogo. Seguidamente, se generan los modelos de domótica a partir de los requisitos de la aplicación y, por último, se genera el código de la aplicación domótica a partir de estos modelos. De esta manera, la demo contiene un ejemplo completo de aplicación del proceso, desde los requisitos hasta el código. En el correo de invitación se instaba a los participantes a ver la demo, y contestar, una vez vista, el cuestionario compuesto por diez preguntas fuertemente relacionadas con el contenido de la demo (ver Tabla 45).

Las respuestas de los participantes se muestran en la Tabla 46. Estudiando cada pregunta del cuestionario de manera individual, la Tabla 47 contiene el estudio estadístico descriptivo (moda, primer cuartil, mínimo, mediana, máximo, tercer cuartil, rango y rango intercuartil) de las respuestas (Tabla 46) tomando todas las respuestas de manera conjunta. La Figura 103 muestra la misma información gráficamente utilizando un diagrama de cajas (*boxplot*). Este diagrama es una presentación visual de los valores extremos (mínimo y máximo) de una variable y los cuartiles (primer cuartil, segundo cuartil o mediana y tercer cuartil), lo que permite mostrar la dispersión de la variable. La frecuencia de las respuestas se encuentra de forma tabulada en la Tabla 48 y gráficamente en la Figura 104. Se ha utilizado Microsoft Excel© tanto para calcular los estadísticos como para crear los diagramas.

- Q1.** It is easy to define the requirements of a new project by reusing the requirements from the HA requirements catalogue.
- Q2.** The HA requirements catalogue is sufficiently complete to specify a new HA application.
- Q3.** The requirements relationships which have been specified in the HA requirements catalogue (such as parent-child relationships) are helpful in assisting me not to forget essential requirements of the HA application.
- Q4.** The automatically generated HA models are intuitive and allow designers to understand the system behaviour.
- Q5.** If necessary, it would be easy to refine the HA models generated by using the model editor.
- Q6.** It is faster to refine the HA model generated (if necessary) than to build it from scratch.
- Q7.** It seems to me that an HA system developer would gain time by using this new approach and the development process would be improved.
- Q8.** In general terms, I think that the automatic transformation of requirements models to HA models improves the quality of the HA models.
- Q9.** The requirements engineering phase is improved by reusing the HA requirements catalogue.
- Q10.** The traceability matrix generated, which links requirements to functional units, is helpful to discover which devices are related to a specific requirement.

Tabla 45 Cuestionario.

Perfil		id	Q ₁	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Domótica	A1	61	3	3	4	4	4	4	3	4	3	3
	A1	212	2	2	3	3	4	5	5	5	4	3
	A1	31	4	4	5	5	3	5	3	5	4	5
	A1	71	4	5	5	4	4	5	5	4	4	5
	A2	152	4	2	4	2	2	3	2	5	4	4
	A2	232	4	3		4	3	3	4	5	5	5
	A2	101	4	4	3	5	4	3	4	5	4	5
	A2	202	3	4	4	4	3	3	5	5	5	5
	A2	131	5	4	5	5	4	5	3	5	4	5
	A2	51	5	4	4	4	5	5	5	5	5	4
	A3	41	5	3	4	4	3	4	4	4	3	4
	A3	252	4	3	5	5	4	4	5	4	4	5
DSDM	A4	192	3	3	3	3	4	2	2	3	3	3
	A4	222	4	3	3	4	4	4	4	3	3	4
	A4	301	4	4	2	4	4	4	2	4	4	5
	A4	91	4	3	4	3	5	3	4	4	5	4
	A4	172	5	3	4	3	4	4	5	4	4	5
	A4	281	4	3	4	4	3	4	5	5	4	5
	A4	291	5	3	5	3	4	5	4	4	4	4
	A4	182	5	4	4	5	3	5	3	3	5	5
	A4	162	4	3	5	4	4	5	5	5	5	5
	A4	121	5	5	5	4	4	5	5	3	5	5
	A4	262	5	4	4	4	4	5	5	5	5	5
	A4	132	4	3	5	4	3	4	4	5	5	5

Tabla 46 Respuestas al cuestionario.

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10
Moda	4	3	4	4	4	5	5	5	4	5
1 ^{er} Cuartil	4	3	4	3.75	3	3.75	3	4	4	4
Mínimo	2	2	2	2	2	2	2	3	3	3
Mediana	4	3	4	4	4	4	4	4.5	4	5
Máximo	5	5	5	5	5	5	5	5	5	5
3 ^{er} Cuartil	5	4	5	4	4	5	5	5	5	5
Rango	3	3	3	3	3	3	3	2	2	2
Rango inter-cuartil	1	1	1	0.25	1	1.25	2	1	1	1

Tabla 47 Estadística descriptiva para los datos de la Tabla 46.

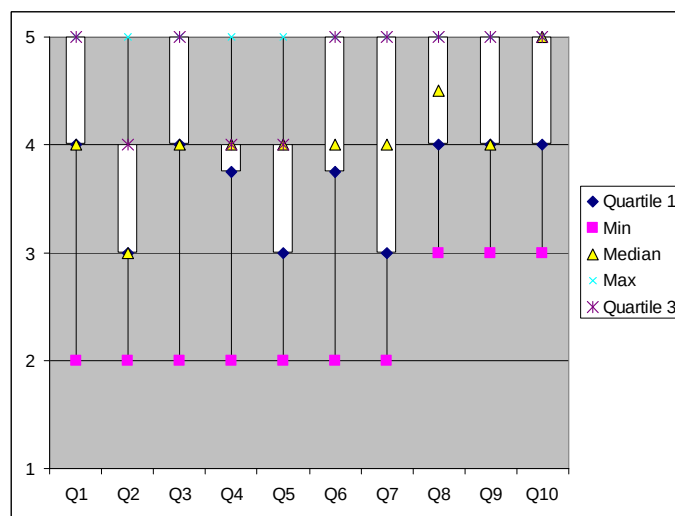


Figura 103 Diagrama de cajas para cada pregunta.

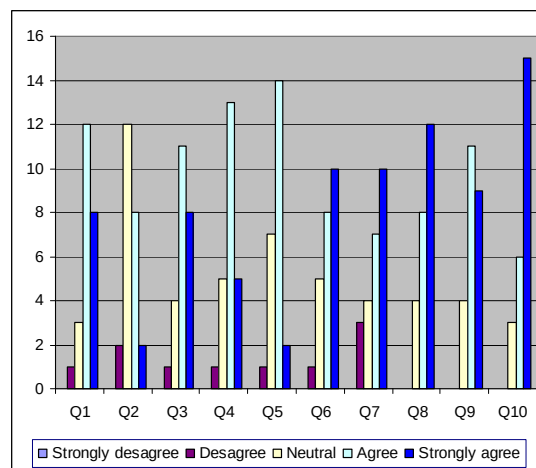


Figura 104 Frecuencia de respuestas.

Las principales conclusiones que se pueden extraer del análisis estadístico de la Tabla 47 son:

- Las preguntas con la mediana más alta son Q8 y Q10, de manera que son las preguntas con las que la mayoría de los participantes están de acuerdo. El 50% de los encuestados (ver Tabla 48 para los porcentajes) están totalmente de acuerdo (*strongly agree*) en que la transformación automática de los modelos de requisitos en modelos domóticos mejora la calidad de los modelos domóticos, y el 62,5% de los participantes están totalmente de acuerdo con la utilidad de la matriz de trazabilidad que liga los requisitos con las unidades funcionales de los sistemas domóticos.

		Muy en desacuerdo	Desacuerdo	Neutro	Acuerdo	Muy de acuerdo
Q1	#	0	1	3	12	8
	%	0%	4.2%	12.5%	50%	33.3%
Q2	#	0	2	12	8	2
	%	0%	8.3%	50%	33.3%	8.3%
Q3	#	0	1	4	11	8
	%	0%	4.2%	16.7%	45.8%	33.3%
Q4	#	0	1	5	13	5
	%	0%	4.2%	20.8%	54.2%	20.8%
Q5	#	0	1	7	14	2
	%	0%	4.2%	29.2%	58.3%	8.3%
Q6	#	0	1	5	8	10
	%	0%	4.2%	20.8%	33.3%	41.7%
Q7	#	0	3	4	7	10
	%	0%	12.5%	16.7%	29.2%	41.7%
Q8	#	0	0	4	8	12
	%	0%	0%	16.7%	33.3%	50%
Q9	#	0	0	4	11	9
	%	0%	0%	16.7%	45.8%	37.5%
Q10	#	0	0	3	6	15
	%	0%	0%	12.5%	25%	62.5%

Tabla 48 Frecuencia de respuesta de cada pregunta.

- No existe ningún desacuerdo con las preguntas Q8, Q9 y Q10, dado que el valor mínimo en todos los casos es tres. En relación a la pregunta Q9, el 45,8% de los participantes están de acuerdo en que la fase de IR se mejora utilizando el catálogo HAREL. El mínimo del resto de preguntas es 2, pero merece la pena destacar que el desacuerdo procede sólo de 4 de los 24 participantes (ids 212, 152, 192 y 301 en Tabla 46). El participante 152 (a partir de ahora id152) ha sido el más crítico con la propuesta. Afortunadamente este participante rellenó el campo de respuesta libre destinado a otros comentarios, lo que nos ha permitido entender sus respuestas. Por un lado, id152 no está de acuerdo con la pregunta Q2. A pesar de que reconoce que le gusta el catálogo de requisitos, le gustaría saber cómo se pueden crear nuevos

requisitos. Esta es una reflexión interesante que teníamos identificada como trabajo futuro. Además, id152 tampoco está de acuerdo con las preguntas Q4 y Q5 en relación con los modelos domóticos. Piensa que los modelos domóticos generados están muy relacionados con la herramienta ETS que se utiliza para el desarrollo de estas aplicaciones y que, por tanto, son difíciles de entender para aquellas personas que no están familiarizadas con dicha herramienta. Una explicación más detallada de los modelos en la demo, quizás podría haber favorecido que id152 cambiara de opinión. En un experimento anterior en relación a la *usabilidad* del editor de modelos que soporta el DSL de los sistemas domóticos [148], el grupo que participaba en el experimento valoró muy positivamente el editor de modelos, incluyendo la facilidad de aprendizaje y la comprensión de los modelos. En este caso, se desarrollaron tres sesiones de entrenamiento previas al experimento, en lugar de una demo de 30 minutos que precede a nuestra encuesta.

- Q7 es la pregunta que ha generado más controversia puesto que es la que presenta el mayor grado de variación (el rango inter-cuartil es 2, el más alto). Es la pregunta con la que más gente está en desacuerdo (12,5% de los participantes), aunque el 41,7% de los encuestados está totalmente de acuerdo con que el proceso de desarrollo se mejora con la aplicación de nuestra propuesta. Uno de los participantes destacó que, sin ninguna duda, las primeras etapas del proyecto se beneficiaban fuertemente debido a nuestro enfoque de transformación. Sin embargo, las conclusiones derivadas del análisis de esta pregunta no pueden ser en ningún caso concluyentes puesto que la redacción de esta pregunta se ha demostrado que no era adecuada (ver Sección 8.5.1).
- La pregunta Q2 se puede considerar que es la más neutra, en el sentido de que el 50% de los encuestados la han valorado con un 3 (3 es también tanto la moda como la mediana). Somos conscientes de que el catálogo de requisitos es incompleto para poder especificar una aplicación domótica real, pero sí que es cierto que incluye los servicios más habituales en este tipo de aplicaciones, de manera que se podría definir una nueva aplicación domótica básica a partir del catálogo HAREL. Sin embargo, la falta de completitud ha sido puesta de manifiesto por los participantes, y el 8,3% de ellos (que pertenecen al perfil de domótica) han mostrado su desacuerdo. Es razonable que no exista desacuerdo entre los participantes expertos en DSDM, dado que su desconocimiento en el dominio hace que no puedan evaluar la completitud del catálogo de requisitos. De ahí que, el 66,7% de los participantes en DSDM valoraran esta pregunta con un 3. Pensamos que el catálogo HAREL es un buen punto de partida para la definición de un catálogo de requisitos de los sistemas domóticos completo.
- Aunque tanto el rango de las preguntas Q2 y Q5 varía entre 3 y 4, a diferencia de Q2, la moda y la mediana de Q5 es 4. Esto significa que el 58,8% de los encuestados están de acuerdo con que los modelos domóticos generados podrían refinarse fácilmente utilizando el editor para este tipo de modelos.

Finalmente, la Figura 105 y la Figura 106 recogen el análisis estadístico de cada pregunta dividiendo la muestra en los dos grupos definidos previamente: expertos en domótica y expertos en DSDM. Las principales diferencias entre estos dos grupos se pueden resumir como sigue:

- En relación a la pregunta Q1, ambos grupos coinciden en la sencillez del proceso de reutilización de requisitos (4 es la mediana de cada grupo, lo que significa que el 50% de los participantes de cada grupo están de acuerdo con la pregunta Q1). Sin embargo, uno de los expertos en sistemas domóticos (id212 en Tabla 46) no está de acuerdo con esta pregunta, mientras que no existe ningún desacuerdo entre los expertos en DSDM, es más, el 41,7% de los encuestados que pertenecen a este grupo están totalmente de acuerdo con la pregunta Q1.
- En relación con la pregunta Q3, ambos grupos coinciden en la utilidad de definir relaciones de traza entre los requisitos (como las relaciones padre-hijo). El grado de variación es ligeramente mayor en el grupo de expertos en DSDM (el rango inter-cuartil es de 1,25 en lugar de 1), dado que existe un participante de este grupo que está en desacuerdo con esta relación de traza. No existe desacuerdo entre los participantes del grupo de expertos en domótica.
- Como era lógico pensar, los modelos domóticos parecen ser más intuitivos para los desarrolladores de este tipo de sistemas (Q4). Aunque ambos grupos muestran el mismo grado de consenso (el rango inter-cuartil es 1 en los dos), el rango de los expertos en domótica oscila entre 4 (50%) y 5 (33,3%), mientras que el rango de los expertos en DSDM oscila entre 3 (33,3%) y 4 (58,3%).
- También parece un resultado razonable el que los expertos en DSDM estén de acuerdo en que es fácil refinar los modelos domóticos generados, utilizando el editor de modelos (Q5), concretamente, el 66,7% de los participantes que pertenecen a este grupo coinciden en esta apreciación, con el grado de dispersión más bajo (0,25). En cuanto a los expertos en domótica, aunque el 50% de ellos están de acuerdo con lo que se dice en la pregunta Q5, existe un grado de dispersión mayor. Incluso el encuestado id152 está en desacuerdo con esta pregunta, como explicábamos anteriormente. Uno de los encuestados comentaba que un ingeniero del software habitualmente no se siente cómodo si tiene que cambiar algo que ha sido generado automáticamente.
- Aunque no existe ningún desacuerdo entre los participantes pertenecientes al grupo de domótica en relación a la pregunta Q6, y la moda de esta pregunta en este grupo es 5, el grado de dispersión es mayor entre los encuestados de este perfil (el rango inter-cuartil es 2). El 33,3% de los expertos en domótica encuestados, no tienen claro si es más rápido refinar los modelos domóticos generados o construirlos desde el principio. Habría sido necesario que utilizaran la herramienta para el desarrollo del caso de estudio para que lo comprobaran. Parece que hay un mayor grado de consenso entre los encuestados del grupo del DSDM, quienes están más acostumbrados a la representación de los modelos. En ambos grupos, el 41,7% de los participantes están totalmente de acuerdo con la pregunta Q6.
- En relación a la mejora de la etapa de IR (Q9), parece que los participantes del grupo del DSDM son más conscientes de los beneficios de la reutilización de requisitos dado que el 50% de ellos están totalmente de acuerdo con la pregunta Q9. Entre los participantes del grupo de domótica, el grado de variación es bajo (el rango inter-cuartil es 0,25), lo que significa que, básicamente, están de acuerdo con esta pregunta (58,3% de los encuestados).

105.a Diagrama de cajas para las respuestas del grupo de los expertos en domótica

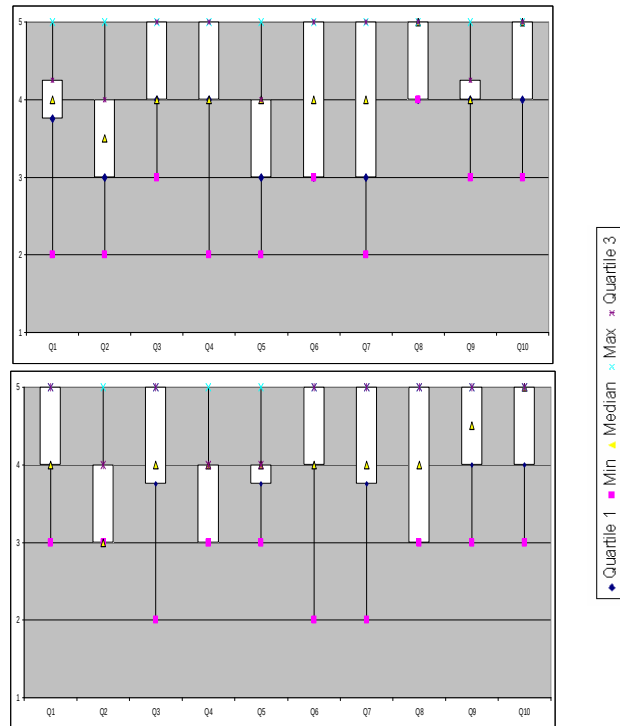


Figura 105 Estadística descriptiva por grupos.

106.a Frecuencia de las respuestas para el grupo de expertos en domótica

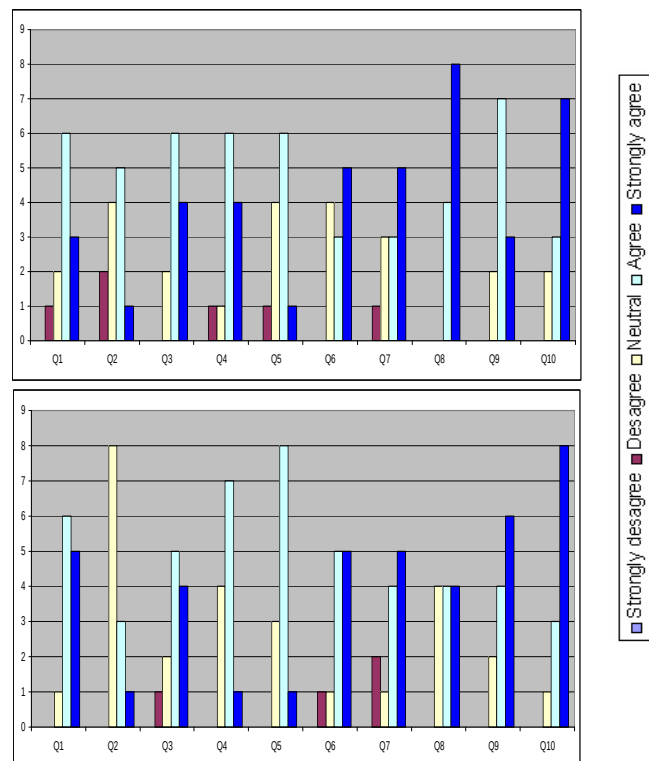


Figura 106 Frecuencia de las respuestas por grupos.

- Una de las conclusiones más relevantes que podemos extraer del estudio es que el 66,7% de los participantes del grupo de domótica están *totalmente de acuerdo* con la utilidad de las transformaciones, de los modelos de requisitos en modelos que representan un sistema domótico, para mejorar la calidad de los modelos domóticos generados (Q8). El resto de los participantes del grupo de domótica están de acuerdo con esta apreciación, y no existe ninguna respuesta que muestre desacuerdo o neutralidad en este hecho. En el grupo de los expertos en DSDM, las respuestas están distribuidas uniformemente entre 3 y 5 (33,3% cada una). Este resultado confirma la hipótesis inicial.

8.8.1. Limitaciones del estudio

Aunque los resultados obtenidos son alentadores, somos conscientes de que existen dos limitaciones fundamentales en nuestro proceso de evaluación: falta de experimentación personal de los participantes y el tamaño de la muestra. Por un lado, hemos utilizado una demostración multimedia (demo) para resolver el problema de reunir a todos los participantes, pero esto ha conllevado que los participantes no hayan podido utilizar la herramienta (editores de requisitos y de domótica) y que no hayan tenido la oportunidad de aplicar el proceso de desarrollo por ellos mismos. La encuesta, por tanto, sólo nos ha permitido obtener la percepción de los expertos hacia nuestra propuesta en base a su propia experiencia previa. Sin embargo, pensamos que su experiencia es lo suficientemente fundada para poder proporcionarnos valiosos comentarios acerca de la viabilidad del nuestro enfoque. Además, dado que era importante ajustar la duración de la demo para evitar que los participantes abandonaran la encuesta por falta de tiempo, es posible que no todos los aspectos de la aproximación se hayan explicado con el detalle suficiente para cada participante y les hayan podido surgir dudas durante la visualización de la demos. Puesto que los participantes no podían obtener la respuesta a sus dudas de manera inmediata, algunas cuestiones de la encuesta se han podido ver afectadas de manera negativa sólo por falta de entendimiento, como pensamos que ha ocurrido. Por tanto, somos conscientes de estas limitaciones, pero a pesar de ellas, hemos abordado la evaluación de este modo tras un estudio concienzudo de las ventajas e inconvenientes. Las razones que nos llevaron finalmente a decidirnos a continuar en esta línea fueron:

- Obtener una rápida retroalimentación de los expertos. Pensamos que la experiencia de los expertos les permite conseguir una impresión general de la propuesta sólo viendo la demo, y esta impresión es la retroalimentación útil que necesitamos para seguir con nuestra investigación en esta línea.
- Resolver los problemas de distribución geográfica de los expertos. El número de expertos conocidos relacionados con el tema de la propuesta es limitado y están geográficamente dispersos. En consecuencia, era difícil reunirlos en una misma localidad para realizar un experimento controlado.
- Animar a la participación a los expertos proporcionándoles material que no les llevara mucho tiempo responder. Podríamos haber dejado la herramienta disponible para que los expertos se la descargasen y “jugaran” con ella durante unos días. Pero

pensamos que el número de respuestas habría sido menor que la que hemos obtenido por restricciones de tiempo de los encuestados.

Por otro lado, el tamaño final de la muestra no es lo suficientemente grande como para poder generalizar nuestras conclusiones a toda la población a partir de la que se ha tomado la muestra. La mayoría de los métodos estadísticos necesitan que el tamaño de la muestra sea de al menos cincuenta individuos. Además, la selección del método estadístico debería haberse planeado antes de abordar la encuesta en base a la población objetivo. En nuestro caso, no conocemos la población. Por tanto, nuestra intención no era generalizar las conclusiones del estudio a la población completa sino simplemente describir los datos obtenidos a partir de la muestra. Dada la experiencia de los expertos seleccionados, confiamos en sus apreciaciones, y los resultados nos animan a continuar con esta línea de investigación. Además, nos han proporcionado una útil retroalimentación que nos servirá para mejorar tanto la herramienta como el proceso de desarrollo. Merece la pena destacar, que no es una tarea sencilla encontrar un grupo de expertos adecuado para evaluar una propuesta como la que recoge esta tesis doctoral, puesto que el conocimiento requerido para evaluar un nuevo método de desarrollo de software no es trivial, y la disponibilidad de los usuarios potenciales no es común. Sin embargo, hemos sido capaces de encontrar un grupo con una formación y experiencia cercana a nuestra aproximación, que incluye tanto el desarrollo de sistemas domóticos como las técnicas relacionadas con el DSDM, que proporciona un valor añadido a la encuesta.

Por último, queda resaltar que, aunque comprobamos la redacción del cuestionario antes de llevar a cabo la encuesta, a posteriori nos dimos cuenta que la pregunta Q7 (Tabla 45) se había formulado de manera que no es posible extraer ninguna conclusión de las respuestas obtenidas. En su formulación se incluyen dos preguntas diferentes, de manera que, en caso de desacuerdo, no queda claro si el encuestado no está de acuerdo con la primera parte de la pregunta (si los desarrolladores van a ganar tiempo), con la segunda (si se mejorará el proceso de desarrollo) o con las dos. Desde una perspectiva lógica, el encuestado sólo estará de acuerdo con la pregunta si está de acuerdo con ambas partes. De hecho, la pregunta Q7 se escribió en el sentido de que la segunda parte de la pregunta (la mejora del proceso de desarrollo) era una consecuencia de la primera parte (reducción del tiempo) pero no estamos totalmente seguros de que esta haya sido la interpretación que le han dado los encuestados.

8.9. Conclusiones y aportaciones al estado del arte

Los trabajos desarrollados por el grupo DSIE de la UPCT han aplicado con éxito las técnicas del DSDM para el desarrollo de sistemas domóticos. La definición de un DSL para domótica permite elevar el nivel de abstracción de los modelos y desarrollar diseños de las aplicaciones independientes de la tecnología de implementación utilizada. Así, el DSL proporciona una notación única para la definición de las aplicaciones domóticas. Después, los modelos conformes al DSL, se pueden ir refinando, mediante transformaciones de modelos, hasta generar el código de la aplicación para una plataforma concreta. Aunque actualmente sólo están definidas las transformaciones

para generar el código para la tecnología KNX, este esquema de desarrollo permitiría que se generara el código para otras plataformas, como Lonworks, a partir del mismo modelo de diseño.

Por otro lado, los procesos de desarrollo tradicionales para los sistemas domóticos no tienen en cuenta la gestión de requisitos. Además, no prestan atención a la reutilización, sino que se rigen, más bien, por la cultura del proyecto actual. La incorporación de REMM y del proceso de IR SIREN_{MD}, contribuye a la mejora de estos procesos de desarrollo en general, y del proceso propuesto con el DSIE, en particular, proporcionando la capa de gestión y reutilización de requisitos previa al diseño de los modelos de domótica. El proceso final, fruto de la colaboración de los grupos de investigación de la UPCT y la UMU, representa una contribución, creemos que importante, en el campo de la domótica pues proporciona, a los desarrolladores de este tipo de sistemas, herramientas que les permite mejorar la calidad de los modelos y sus procesos de desarrollo, incorporando principios propios de la Ingeniería del Software.

Esta contribución ha sido refrendada por la encuesta realizada a los expertos en el dominio quienes han destacado, fundamentalmente, la utilidad de las transformaciones desde los modelos de requisitos para mejorar la calidad de los modelos de domótica. También coinciden en que la etapa de IR se mejora teniendo en cuenta la reutilización de requisitos del catálogo HAREL y que el proceso de reutilización propuesto es sencillo.

Para concluir sólo resta destacar que en esta tesis doctoral se propone un proceso general, ReToCo, que ha sido particularizado al dominio de los sistemas domóticos. Sin embargo, pensamos que esta misma estrategia podría aplicarse a otros dominios, como, por ejemplo, el de las redes de sensores inalámbricos [347] en el que también se han obtenido buenos resultados en la aplicación del enfoque DSDM y para el que se ha definido un DSL. Para aplicar el proceso ReToCo a otros dominios sería necesario abordar el proceso de ingeniería de dominio para definir el DSL, si no se ha hecho ya, y elicitar los requisitos genéricos del dominio que serían representados en un modelo de requisitos reutilizables. El conocimiento del dominio también se utilizaría para implementar las reglas de transformación que encapsularían la forma en la que los expertos del dominio modelarían los requisitos especificados utilizando el DSL.

Capítulo 9

Análisis de resultados, conclusiones y vías futuras

9.1. Introducción

Una vez desarrollada y presentada la propuesta que constituye esta tesis doctoral, debemos volver la vista atrás, recapitular y analizar si se han alcanzado los objetivos que se habían establecido al principio de la investigación. El resultado de este análisis, que se presenta en la Sección 9.2, nos permitirá refrendar, en su caso, la hipótesis de partida. Además, en la Sección 9.3, recapitulamos las principales conclusiones y aportaciones de esta tesis doctoral.

Por otro lado, en el transcurso de la investigación, los resultados parciales se han ido presentando en diferentes foros, tanto nacionales como internacionales, con el objeto de darlos a conocer a la comunidad científica y obtener la retroalimentación necesaria que nos permitiera avanzar y mejorar nuestro trabajo. En la Sección 9.4, detallamos las publicaciones presentadas en el marco de esta tesis doctoral y destacamos los trabajos que las referencian.

Por último, la línea de investigación desarrollada en el campo de la Ingeniería de Requisitos (IR) y el Desarrollo de Software Dirigido por Modelos (DSDM) no debe terminar con la presentación de esta tesis doctoral, sino que debe abrir camino a otros trabajos futuros. Nuestras reflexiones en este sentido se presentan en la sección 9.5.

9.2. Análisis de la consecución de objetivos

En esta sección resumimos las principales aportaciones de la investigación llevada a cabo en el marco de esta tesis doctoral, para mostrar cómo tales aportaciones desarrollan los objetivos planteados en la Sección 1.2.

Objetivo 1. *Definir un método práctico de IR basado en reutilización, en el que integrar las técnicas del DSDM.*

En el Capítulo 4 se ha presentado SIREN, un método de IR basado en reutilización y en los principales estándares de Ingeniería del Software. La

definición de SIREN incluye: 1) un modelo de proceso en espiral, que incorpora explícitamente las tareas relacionadas explícitamente con la reutilización de requisitos; 2) unas guías para la definición de requisitos y relaciones de trazas entre ellos y 3) una herramienta de soporte, denominada SirenTool, implementada como un *add-in* de la herramienta comercial RequisitePro. De esta forma, SirenTool integra las capacidades de gestión de requisitos de RequisitePro, que complementa con las funcionalidades necesarias para dar soporte a la reutilización de requisitos. El método SIREN ha sido publicado, entre otros sitios, en la principal revista de IR, *Requirement Engineering Journal* [343] y en el principal congreso de esta disciplina *International Conference on Requirements Engineering* (ICRE) [344].

Objetivo 2. *Estudiar las propuestas de integración de las técnicas de DSDM en IR, en concreto analizaremos los metamodelos de requisitos existentes en la literatura y las transformaciones desde modelos de requisitos hacia otros modelos del desarrollo.*

Para abordar el estudio del estado del arte en relación tanto a los metamodelos de requisitos como a las transformaciones desde los modelos de requisitos, se han llevado a cabo sendas revisiones sistemáticas de la literatura (RSL) que se detallan en el Capítulo 3. El desarrollo de las RSL permite analizar los trabajos existentes en relación al tema que se quiere investigar y, en consecuencia, detectar posibles “lagunas” que puedan dar lugar a nuevas investigaciones. En este sentido, de las RSLs abordadas se puede concluir que: 1) es necesario llegar a un consenso en la definición de un metamodelo de requisitos que recoja los conceptos y relaciones (tanto pre-RS como post-RS) que representen una especificación de requisitos y 2) no existen propuestas que soporten la generación automática de modelos a partir de los modelos de requisitos basados en lenguaje natural, de manera que se pueda definir un proceso completo desde los requisitos hasta el código. Estos resultados justifican el interés de los trabajos que se presentan en esta tesis doctoral y sustentan la hipótesis de partida. Esta hipótesis fue presentada en la *Segunda Conferencia Internacional en las Tecnologías del Software y Datos* (ICSOFT 2007) [233].

Objetivo 3. *Definir un metamodelo de requisitos que incluya los conceptos fundamentales para dar soporte a las actividades de IR y que dé soporte a la reutilización sistemática de requisitos.*

En el Capítulo 5 se ha presentado REMM (*Requirements Engineering MetaModel*), un metamodelo de requisitos que permite definir modelos de requisitos que representan una especificación de requisitos textual. Los conceptos y relaciones especificadas en REMM son fruto del estudio y análisis de la literatura especializada, nuestra experiencia en la definición y aplicación del método SIREN y en las buenas prácticas en IR. REMM soporta las dos facetas implicadas en la reutilización de requisitos, el modelado de requisitos *para* reutilización y el modelado de requisitos *con* reutilización. La versión inicial de REMM fue presentada en la

conferencia europea *TOOLS Europe 2007* [349], y posteriormente la extensión de REMM para dar soporte a la variabilidad y reutilización de requisitos se presentó en el *EMMSAD 2008* [234].

Objetivo 4. *Definir transformaciones modelo a texto, para generar la documentación de requisitos, y transformaciones modelo a modelo, para integrar los modelos de requisitos en un proceso de DSDM completo (desde los requisitos al código).*

El proceso completo de DSDM, desde los requisitos hasta el código, se detalla en el Capítulo 6. Este proceso diferencia las etapas de Ingeniería del Dominio e Ingeniería de la Aplicación, de forma análoga al desarrollo de líneas de producto software. Durante la etapa de Ingeniería del Dominio se tiene que definir: 1) un DSL (*Domain Specification Language*), que permita especificar los sistemas utilizando conceptos cercanos al dominio del problema; 2) el catálogo reutilizable de requisitos del dominio (modelo de requisitos *para* reutilización); y 3) las transformaciones desde el modelo de requisitos a los modelos del dominio conformes con el DSL. Siguiendo un enfoque de modelado específico del dominio, la definición del DSL irá acompañada de un editor para los modelos y las transformaciones modelo a texto que permitan generar el código de la aplicación. De esta forma, una vez definidos todos estos elementos, será posible, durante la etapa de Ingeniería de la Aplicación: 1) especificar los requisitos de nuevas aplicaciones del dominio (modelo de requisitos *con* reutilización), reutilizando los requisitos del catálogo; 2) generar el modelo de diseño de la aplicación (modelo conforme al DSL) a partir de la especificación de requisitos y 3) generar el código de la aplicación a partir del modelo del dominio. Por tanto, hemos definido un proceso completo, apoyándonos en las técnicas que nos ofrece el DSDM (metamodelado y transformaciones de modelos), que nos permite generar el código de la aplicación a partir de los modelos de requisitos. Además, a partir de los modelos de requisitos también se pueden generar automáticamente los documentos textuales de especificación de requisitos, mediante transformaciones modelo a texto.

Objetivo 5. *Diseñar e implementar el prototipo de una herramienta de soporte a los modelos definidos en el Objetivo 3 y las transformaciones definidas en el Objetivo 4.*

La herramienta de soporte al proceso se denomina REMM-Studio⁺ y se presenta en el Capítulo 7. REMM-Studio⁺ proporciona diferentes editores para la creación de modelos de requisitos para reutilización y modelos de requisitos con reutilización. En este segundo editor se incorporan las capacidades de búsqueda y selección de requisitos, definidos en los catálogos, que facilitan la reutilización sistemática de requisitos. Por otro lado, a partir de los modelos de requisitos creados se pueden lanzar las transformaciones y visualizar los modelos generados. La herramienta ha sido presentada en la sección de demostración de herramientas en diferentes foros nacionales [235, 350] e internacionales [236].

Objetivo 6. *Validar los resultados anteriores en el marco de un caso de estudio real.*

El proceso definido como resultado del Objetivo 4 se ha particularizado al caso de estudio de los sistemas domóticos. Esta instanciación del proceso se ha llevado a cabo en colaboración con el grupo de investigación DSIE, de la Universidad Politécnica de Cartagena, que cuenta con experiencia real en el desarrollo de estos sistemas. La colaboración se ha materializado con la definición de un catálogo de requisitos reutilizables para los sistemas domóticos (denominado HAREL) y las transformaciones de los modelos de requisitos a los modelos conformes con el DSL de domótica Habitation. La aplicación del proceso al campo de la domótica ha sido validada mediante la realización de una encuesta a través de Internet. Los encuestados han sido expertos tanto en el desarrollo de sistemas domóticos como en la aplicación de las técnicas de DSDM, en algunos casos los participantes tenían experiencia en ambos campos. El resultado principal del estudio es que los desarrolladores de sistemas domóticos consideran de utilidad la definición de las transformaciones a partir de los requisitos como medio para mejorar la calidad de los modelos generados. La instanciación del proceso a los sistemas domóticos, y las conclusiones del el proceso de evaluación mediante la encuesta, se han enviado para su evaluación a la revista *Information and Software Technology* [232]. Actualmente se encuentra en proceso de segunda revisión.

La hipótesis de partida de esta tesis doctoral decía que la IR se podía beneficiar al aplicar las técnicas del DSDM, puesto que la formalización en un metamodelo de los conceptos y relaciones involucradas en esta fase favorecerían la reutilización sistemática de requisitos y la trazabilidad, y por su parte el enfoque DSDM se beneficiaría de la definición del metamodelo de requisitos, puesto que permitiría integrar los requisitos en el proceso de desarrollo de forma compatible con su filosofía basada en modelos (Sección 1.2). Creemos que el desarrollo de los objetivos anteriores permite probar dicha hipótesis de partida y satisface el objetivo global de esta tesis doctoral: *Aplicar las técnicas del DSDM (metamodelado y transformaciones de modelos) para dar soporte a las actividades de IR, favoreciendo la reutilización y trazabilidad de requisitos, de manera que los requisitos se integren en un proceso completo de DSDM (desde los requisitos hasta el código).*

9.3. Conclusiones finales

Aunque al final de cada uno de los capítulos, que componen esta tesis doctoral, se presentan las conclusiones y aportaciones al estado del arte, en esta sección vamos a recoger los aspectos más relevantes a modo de resumen.

El estudio sistemático del estado del arte acerca de los metamodelos de requisitos propuestos en la literatura, y las transformaciones de modelos a partir de los requisitos, justifica el interés de los trabajos realizados en esta tesis doctoral. Los resultados de las revisiones sistemáticas de la literatura abordadas revelan que: 1) es necesario definir un metamodelo de requisitos que cubra tanto las trazas pre-RS como post-RS, teniendo en cuenta las buenas prácticas de la IR; 2) los trabajos estudiados no abordan las

transformaciones de modelos que permitan generar modelos de análisis, o diseño, a partir de modelos de requisitos, que representan especificaciones textuales de requisitos; 3) el uso de un lenguaje natural restringido puede ayudar a definir las transformaciones a partir de los modelos de requisitos y 4) si existe un patrón predefinido que los expertos siguen para realizar los modelos de diseño, este conocimiento algorítmico se puede implementar mediante reglas de transformación.

Todas estas consideraciones han sido tenidas en cuenta en el desarrollo de nuestra investigación, fruto de la cual se ha definido: 1) el metamodelo de requisitos REMM, que incluye un modelo de trazabilidad que abarca todas las trazas pre-RS y post-RS que hemos considerado como fundamentales; 2) un proceso, basado en transformaciones de modelos, que propone generar los modelos conformes a un DSL a partir de las especificaciones de requisito. Este proceso se ha particularizado al caso de estudio de los sistemas domóticos para el que se ha definido un lenguaje de especificación de requisitos restringido, a partir del cual se pueden generar los modelos de la aplicación domótica (conformes a un DSL de domótica), utilizando reglas de transformación, que encapsulan el conocimiento de los expertos en el dominio. Siguiendo el enfoque de modelado específico de dominio, a partir de los modelos de la aplicación se puede generar el código del sistema domótico para la plataforma elegida.

Si bien REMM es la pieza clave para representar la especificación de requisitos, dicha especificación debe obtenerse siguiendo un proceso de IR. El proceso que hemos propuesto es una adaptación del método SIREN, basado en la reutilización de requisitos, al DSDM, y que hemos denominado SIREN_{MD}. El método SIREN, en cuya definición ha participado esta doctoranda, es un método práctico, sencillo y flexible para mejorar el estado de la práctica de IR, que ha sido probado en entornos industriales y que pensamos puede servir como un primer paso para acercar dicho estado de la práctica al estado de la investigación en IR.

9.4. Contraste de resultados de la investigación

A lo largo del desarrollo de esta tesis doctoral se han publicado diversas contribuciones en diferentes foros científicos. En esta sección se recogen estos resultados de la investigación. Para ello, en primer lugar, en la Tabla 49 se muestra una clasificación de las publicaciones y trabajos realizados.

Tipo de contribución	Total
Artículos en revistas internacionales indexadas en JCR	2+1
Artículos en revistas internacionales no indexadas	1
Capítulos de libro	2
Congresos y talleres internacionales	5
Congresos, talleres y reuniones científicas nacionales	9
Proyectos fin de carrera	2
Informes técnicos	1
Registros de la propiedad intelectual	3
Total de contribuciones	26

Tabla 49 Número de publicaciones y trabajos realizados por tipos.

En las siguientes secciones se desglosan las contribuciones anteriores, por categorías, y al final se destacan las citas a los trabajos publicados.

9.4.1. Artículos en revistas internacionales indexadas en JCR

Referencia	Publicación
(Toval <i>et al.</i> 02a) [343]	Ambrosio Toval, Joaquín Nicolás, <u>Begoña Moros</u> , Fernando García (2002). <i>Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach</i> . Requirements Engineering Journal (REJ) , vol. 6 (4): 205-219. Factor de impacto JCR (2011) = 0,971 (Q2)
(Toval <i>et al.</i> 08) [339]	Ambrosio Toval, <u>Begoña Moros</u> , Joaquín Nicolás, Joaquín Lasheras (2008). <i>Eight Key Issues for an Effective Reuse-based Requirements Process</i> . International Journal of Computer Systems Science and Engineering (IJCSSE) , vol. 23 (6): 373-385. Factor de impacto JCR (2011) = 0,371 (Q4)
(Moros <i>et al.</i> 12) [232]	<u>Begoña Moros</u> , Ambrosio Toval, Francisca Rosique, Pedro Sánchez. <i>Transforming and Tracing Reused Requirements Models to Home Automation Models</i> . Information and Software Technology . (<i>en proceso de revisión</i>) Factor de impacto JCR (2011) = 1,250 (Q2, 1 ^{er} tercio en la categoría COMPUTER SCIENCE, SOFTWARE ENGINEERING)

Tabla 50 Artículos en revistas indexadas en JCR.

9.4.2. Artículos en revistas no indexadas

Referencia	Publicación
(Vicente-Chicote <i>et al.</i> 07a) [348]	Cristina Vicente-Chicote, <u>Begoña Moros</u> , Ambrosio Toval (2007). <i>REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting</i> . Journal of Object Technology , vol. 6 (9): 1660-1769.

Tabla 51 Artículos en revistas no indexadas.

9.4.3. Capítulos de libro

Referencia	Publicación
(Toval <i>et al.</i> 01a) [340]	Ambrosio Toval, Joaquín Nicolás, <u>Begoña Moros</u> (2001). <i>Requisitos Reutilizables de Seguridad en Sistemas de Información y Bases de Datos en Seguridad en Bases de Datos</i> . Editorial Dintel, Madrid: pp. 269-299. ISBN: 84-931933-9-9.
(Toval <i>et al.</i> 02b) [342]	Ambrosio Toval, Joaquín Nicolás, <u>Begoña Moros</u> (2002). <i>SIREN: Un proceso de Ingeniería de Requisitos Basado en Reutilización en Applying Requirements Engineering</i> . Editorial Catedral Publicaciones, Sevilla: pp. 57-72. ISBN: 84-96086-06-2.

Tabla 52 Capítulos de libro.

9.4.4. Congresos y talleres internacionales

Referencia	Publicación
Congresos calificados como ERA (Excellence in Research for Australia) A:	
(Moros <i>et al.</i> 08a) [236]	<u>Begoña Moros</u> , Cristina Vicente-Chicote, Ambrosio Toval (2008). <i>REMM-Studio+: Modeling Variability to Enable Requirements Reuse (tool demo)</i> . 27th International Conference on Conceptual Modeling (ER 2008) . Barcelona, Spain. Lecture Notes in Computer Science 5231, Springer-Verlag Berlin Heidelberg.
Congresos calificados como ERA B:	
(Vicente-Chicote <i>et al.</i> 07b) [349]	Cristina Vicente-Chicote, <u>Begoña Moros</u> , Ambrosio Toval. <i>REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting (paper)</i> . TOOLS Europe 2007: Object, Models, Components and Patterns . Zurich, Suiza.
Otros congresos:	
(Gutiérrez <i>et al.</i> 05a) [116]	Carlos Gutiérrez, <u>Begoña Moros</u> , Ambrosio Toval, Eduardo Fdez-Medina, Mario Piattini (2005). <i>Security Requirements for Web Services based on SIREN (short paper)</i> . Symposium on Requirements Engineering for Information Security (SREIS 2005) en paralelo con RE'05. Paris, Francia.
(Moros <i>et al.</i> 07) [233]	<u>Begoña Moros</u> , Cristina Vicente-Chicote, Ambrosio Toval (2007). <i>A Model-Driven Engineering Approach To Requirements Engineering: How these disciplines may benefit each other (short paper)</i> . 2nd International Conference on Software and Data Technology (ICSOT'07) . Barcelona, Spain.
(Moros <i>et al.</i> 08b) [234]	<u>Begoña Moros</u> , Cristina Vicente-Chicote, Ambrosio Toval (2008). <i>Metamodeling Variability to Enable Requirements Reuse (paper)</i> . Exploring Modeling Methods for Systems Analysis and Design (EMMSAD'08) en paralelo con CAISE'08. Montpellier, France. (calificado como CORE A en el año de publicación)

Tabla 53 Congresos y talleres internacionales.

9.4.5. Congresos, talleres y reuniones científicas nacionales

Referencia	Publicación
(Toval <i>et al.</i> 01b) [341]	Ambrosio Toval, Joaquín Nicolás, <u>Begoña Moros</u> (2001). <i>SIREN: un Proceso de Ingeniería de Requisitos Basado en Reutilización</i> (ponencia). Jornadas de Ingeniería de Requisitos Aplicada (JIRA) . Sevilla.
(Lasheras <i>et al.</i> 04a) [177]	Joaquín Lasheras, Ambrosio Toval, Joaquín Nicolás, <u>Begoña Moros</u> (2004). <i>Soporte automatizado a la reutilización de requisitos</i> (ponencia). VIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2003) . Alicante.
(Lasheras <i>et al.</i> 04a) [181]	Joaquín Lasheras, Ambrosio Toval, Joaquín Nicolás, <u>Begoña Moros</u> (2004). <i>Definición de Requisitos de Seguridad con Fines de Reutilización</i> (ponencia). I Taller de Seguridad en Ingeniería del Software y Bases de Datos (SISBD'2004) . Málaga.

Tabla 54 Congresos, talleres y reuniones científicas nacionales. (1/2)

Referencia	Publicación
(Lasheras <i>et al.</i> 04b) [179]	Joaquín Lasheras, Joaquín Nicolás, Ambrosio Toval, <u>Begoña Moros</u> (2004). <i>Hacia un Modelo del Dominio de los Sistemas Teleoperados a través de una extensión de SIREN</i> (ponencia). II Jornadas de trabajo DYNAMICA . Málaga.
(Gutiérrez <i>et al.</i> 05b) [117]	Carlos Gutiérrez, Begoña Moros, Eduardo Fdez-Medina, Ambrosio Toval, Mario Piattini (2005). <i>Requisitos de Seguridad de Servicios Web en el marco de SIREN</i> (ponencia). II Taller de Seguridad en Ingeniería del Software y Bases de Datos (SISBD'05) . Deusto, Bilbao.
(Nicolás <i>et al.</i> 05) [246]	Joaquín Nicolás, Joaquín Lasheras, Ambrosio Toval, <u>Begoña Moros</u> (2005). <i>Ingeniería de Requisitos basada en reutilización: una propuesta de aplicación a los sistemas teleoperados para la limpieza de cascos de buque</i> (ponencia). III Jornadas de trabajo DYNAMICA . Almagro, Ciudad Real.
(Martínez <i>et al.</i> 05) [207]	Miguel Ángel Martínez, Joaquín Lasheras, Joaquín Nicolás, Ambrosio Toval, <u>Begoña Moros</u> (2005). <i>Análisis de la LOPD y LSSICE, su Impacto en los Sistemas de Información</i> (ponencia). COMMON Europa/España: IBM Forum . Madrid.
(Vicente-Chicote 07c) [350]	Cristina Vicente-Chicote, <u>Begoña Moros</u> , Ambrosio Toval (2007). <i>REMM-Studio: Un entorno integrado para dar soporte a un enfoque de Ingeniería de Requisitos Dirigido por Modelos</i> (tool demo). XII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'07) . Zaragoza.
(Moros <i>et al.</i> 08c) [235]	<u>Begoña Moros</u> , Cristina Vicente-Chicote, Ambrosio Toval (2008). <i>REMM-Studio+: Extensiones para Modelar Variabilidad y Permitir la Reutilización de Requisitos</i> (tool demo). XIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2008) . Gijón, Asturias.

Tabla 55 Congresos, talleres y reuniones científicas nacionales. (2/2)

9.4.6. Proyectos fin de carrera

Referencia	Publicación
(Lasheras 03) [177]	Joaquín Lasheras Velasco (2003). <i>Análisis y prototipazo de una herramienta CARE de soporte al método SIREN: SirenTool</i> . Proyecto Informático. Directores: Ambrosio Toval, Joaquín Nicolás y Begoña Moros. Calificación: Sobresaliente.
(Martínez 05) [205]	Miguel Ángel Martínez Aguilar (2005). <i>Revisión de perfiles de SIREN de Protección de Datos Personales y de Seguridad y aplicación a un caso de estudio real</i> . Proyecto Informático. Directores: Ambrosio Toval, Joaquín Nicolás y Begoña Moros. Calificación: Sobresaliente.

Tabla 56 Proyectos Final de Carrera.

9.4.7. Informes técnicos

Referencia	Publicación
(Nicolás <i>et al.</i> 09) [249]	Joaquín Nicolás, Begoña Moros, Joaquín Lasheras, Ambrosio Toval (2009). <i>SIREN: Un método práctico de Ingeniería de Requisitos basado en reutilización, versión 2.0. TR DIS 1-2009</i> . Departamento de Informática y Sistemas. Universidad de Murcia.

Tabla 57 Informes técnicos.

9.4.8. Registros de la propiedad intelectual

Título	<i>Catálogo de requisitos de Seguridad (Perfil SIREN SRS y SyRS de seguridad (Catálogos reutilizables))</i>	
Inventores	José Ambrosio Toval Álvarez, Joaquín Lasheras Velasco, Joaquín Nicolás Ros, Begoña Moros Valle	
Nº de solicitud	Fecha de presentación	Nº asiento registral
MU-818-2011	29/09/11	08/2011/830
Título	<i>SIREN: Un método práctico de Ingeniería de Requisitos basado en Reutilización (Método de Ingeniería de Requisitos)</i>	
Inventores	José Ambrosio Toval Álvarez, Joaquín Nicolás Ros, Begoña Moros Valle	
Nº de solicitud	Fecha de presentación	Nº asiento registral
MU-819-2011	29/09/11	08/2011/831
Título	<i>SIRENTOOL V2.0 (Software, Manual de usuario e instalación)</i>	
Inventores	Joaquín Lasheras Velasco, José Ambrosio Toval Álvarez, Joaquín Nicolás Ros, Begoña Moros Valle	
Nº de solicitud	Fecha de presentación	Nº asiento registral
MU-820-2011	29/09/11	08/2011/832

Tabla 58 Trabajos incluidos en el Registro de la Propiedad Intelectual.

9.4.9. Citas a los trabajos relacionados con la tesis doctoral

En esta sección presentamos los trabajos que referencian a las contribuciones fruto de esta tesis doctoral (Sección 9.3). Estas referencias las hemos obtenido de los resultados proporcionados por GoogleScholar y la base de datos SCOPUS. De los resultados reportados por ambas fuentes hemos eliminado las auto-citas, esto es, los trabajos en los que coincide alguno de los autores con el que se está referenciando. Si

bien es cierto, que en ocasiones, aunque coincide algún autor, los trabajos se corresponden con líneas de investigaciones diferentes, aunque relacionadas, como es el caso de los trabajos de Molina *et al.* [218-221, 300] o en los trabajos en los que ha participado Vicente-Chicote [196, 347].

9.4.9.1. Citas a los trabajos relacionados con la reutilización de requisitos

(Toval <i>et al.</i> 02a) [343]	<i>Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach.</i>
Citas: 1 al 11 / 47	Nancy R. Mead (2003). <i>Requirements Engineering for Survivable Systems. Technical Report CMU/SEI-2003-TN-013.</i> Carnegie Mellon University.
	Guttorm Sindre , Donald G. Firesmith , Andreas L. Opdahl (2003). <i>A Reuse-Based Approach to Determining Security Requirements. 9th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'03).</i> Klagenfurt/Velden, Austria.
	Ruth Breu, Klaus Burger, Mag. Michael Hafner, Gerhard Popp (2004). <i>Towards a Systematic Development of Secure Systems. Information Systems Security.</i> Vol. 13 (3).
	Ruth Breu, Klaus Burger, Mag. Michael Hafner, Gerhard Popp (2004). <i>Towards a Systematic Development of Secure Systems. 2nd International Workshop on Security in Information Systems (WOSIS 2004).</i> Porto, Portugal.
	Carlos Gutiérrez, Eduardo Fdez-Medina, Mario Piattini (2005). <i>Web services enterprise security architecture: a case study. ACM Workshop on Secure web services (SWS '05).</i> George W. Johnson Center at George Mason University, Fairfax, VA, USA.
	Nicolas Mayer, André Rifaut, Eric Dubois (2005). <i>Towards a Risk-Based Security Requirements Engineering Framework. Eleventh International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'05).</i> Porto, Portugal.
	Bilge Karabacak, Ibrahim Sogukpinar (2005). <i>ISRAM: information security risk analysis method. Computers & Security,</i> Vol. 24 (2): 147-159.
	Eduardo Fernández-Medina, Juan Trujillo, Rodolfo Villarroel, Mario Piattini (2006). <i>Access control and audit model for the multidimensional modeling of data warehouses. Decision Support Systems.</i> Vol. 42 (3): 1270-1289.
	Eduardo Fernández-Medina, Carlos Gutiérrez, Mario Piattini (2006). <i>Developing web services security systems: a case study. International Journal of Web Engineering and Technology.</i> Vol. 2 (4/2006): 292-306.
	Carlos Gutiérrez, Eduardo Fernández-Medina, Mario Piattini (2006). <i>PWSSec: Process for Web Services Security. International Conference on Web Services (ICWS '06).</i> Chicago, Illinois.
	Carlos Gutiérrez, Eduardo Fernández-Medina, Mario Piattini (2006). <i>PWSSec: Secure Web Services-based Systems Development Process. IEEE América Latina (IDEAS 06).</i> Vol. 4 (2): 115-122.

Tabla 59 Trabajos que referencian a SIREN (1/5).

(Toval <i>et al.</i> 02a) [343]	<i>Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach.</i>
Citas: 12 al 22 / 47	Carlos Gutiérrez, Eduardo Fernández-Medina, Mario Piattini (2006). <i>Towards a Process for Web Services Security. Journal of Research and Practice in Information Technology</i> , Vol. 38 (1): 57- 67.
	Ralf Knackstedt, Christian Brelage, Noogie C. Kaufmann (2006). <i>Development of Web-Applications in Consideration of Legal Requirements – Juridical Framework, State-of-the-Art and Conceptual Modelling. Wirtschaftsinformatik / Angewandte Informatik – WI</i> . Vol. 48 (1): 27-35.
	Daniel Mellado, Eduardo Fernández-Medina, Mario Piattini (2006). <i>A comparison of the common criteria with proposals of information systems security requirements. First International Conference on Availability, Reliability and Security (ARES'06). Viena, Austria.</i>
	Daniel Mellado, Eduardo Fernández-Medina, Mario Piattini (2006). <i>A compartive study of proposals for establising security requirements for the development of secure information systems. Lecture Notes in Computer Science</i> 3982, pp. 1044-1053.
	Daniel Mellado, Eduardo Fernández-Medina, Mario Piattini (2006). <i>Applying a Security Requirements Engineering Process. Computer Security - ESORICS 2006. Lecture Notes in Computer Science Volume 4189/2006</i> : 192-206.
	David G. Rosado, Eduardo Fernandez-Medina, Mario Piattini, Carlos Gutiérrez (2006). <i>A Study of Security Architectural Patterns. First International Conference on Availability, Reliability and Security (ARES'06). Viena, Austria.</i>
	Eduardo Fernández-Medina, Juan Trujillo, Mario Piattini (2007). <i>Model-driven multidimensional modeling of secure data warehouses. European Journal of Information Systems</i> . Vol. 16: 374-389.
	Carlos Gutiérrez, Eduardo Fernández-Medina, Mario Piattini (2007). <i>Web Services-based Security Requirements Elicitation. IEICE TRANSACTIONS on Information and Systems</i> . Vol. E90-D (9): 1374-1387.
	Hiroya Itoga, Atsushi Ohnishi (2007). <i>Security Requirements Elicitation via Weaving Scenarios Based on Security Evaluation Criteria. Seventh International Conference on Quality Software (QSIC'07). Portland, Oregon, USA.</i>
	Daniel Mellado, Moisés Rodríguez, Eduardo Fernández-Medina, Mario Piattini (2007). <i>Gestión automatizada de requisitos de seguridad en proyectos de desarrollo de Sistemas de Información. Jornadas sobre Tecnologías de la Información para la Modernización de las Administraciones Públicas (TECNIMAP). Gijón, Asturias.</i>
	Daniel Mellado, Eduardo Fernández-Medina, Mario Piattini (2007). <i>SREPPLine: Towards a security requirements engineering process for software product lines. 5th International Workshop on Security in Information Systems (WOSIS 2007). Funchal, Madeira.</i>

Tabla 60 Trabajos que referencian a SIREN (2/5).

(Toval <i>et al.</i> 02a) [343]	<i>Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach.</i>
Citas: 23 al 34 / 47	Daniel Mellado, Eduardo Fernández-Medina, Mario Piattini (2007). <i>A common criteria based security requirements engineering process for the development of secure information systems.</i> Computer Standards & Interfaces. Vol. 29 (2): 244-253.
	Daniel Mellado, Moisés Rodríguez, Eduardo Fernández-Medina, Mario Piattini (2007). <i>Soporte Automatizado a la Ingeniería de Requisitos de Seguridad.</i> Conferencia Iberoamericana de Software Engineering (CibSE 2007). Isla de Marguerita, Venezuela.
	Daniel Mellado, Eduardo Fernández-Medina, Mario Piattini (2008). <i>Towards security requirements management for software product lines: A security domain requirements engineering process.</i> Computer Standards & Interfaces. Vol. 30 (6): 361-371.
	Daniel Mellado, Eduardo Fernández-Medina, Mario Piattini (2008). <i>Applying a Security Domain Requirements Engineering Process for Software Product Lines.</i> Revista IEEE América Latina (IDEAS'09). Vol. 6 (3): 298 - 305.
	Atsushi Ohnishi, Hiroya Itoga (2008). <i>Verification of scenarios using the common criteria.</i> 3rd International Conference on Software and Data Technologies (ICSOFT 2008). Porto.
	I-Hsin Chou, Chin-Feng Fan (2009). <i>An object-oriented security knowledge framework for the nuclear safety system project.</i> International Journal of Software Engineering and Knowledge Engineering. Vol. 20 (3): 347-365.
	Meenakshi Deshmukh (2009). <i>Security Requirements Engineering Process.</i> Seminar in Information System, Security Engineering. Summer Semester 2009. University of Duisburg-Essen.
	Jens Dibbern, Michael Geisser, Tobias Hildenbrand, Armin Heinzl (2009). <i>Design, Implementation, and Evaluation of an ICT-Supported Collaboration Methodology for Distributed Requirements Determination.</i> Working Paper Series in Business Administration and Information Systems. University of Mannheim, Department of Business Administration and Information Systems.
	Jing Du, Ye Yang, Qing Wang (2009). <i>An Analysis for Understanding Software Security Requirement Methodologies.</i> Third IEEE International Conference on Secure Software Integration and Reliability Improvement (SSIRI '09). Shanghai. China.
	Carlos Gutiérrez, David G. Rosado, Eduardo Fernández-Medina (2009). <i>The practical application of a process for eliciting and designing security in web service systems.</i> Information and Software Technology. Vol. 51 (12): 1712-1738.
	Martin Ivarsson, Tony Gorshek (2009). <i>Technology transfer decision support in requirements engineering research: a systematic review of REj.</i> Requirements Engineering. Vol. 14 (3): 155-175.
	Jostein Jensen, Inger Anne Tøndel, Martin Gilje Jaatun, Per Håkon Meland, Herbjørn Andresen (2009). <i>Reusable Security Requirements for Healthcare Applications.</i> International Conference on Availability, Reliability and Security (ARES'09). Fukuoka, Japan.

Tabla 61 Trabajos que referencian a SIREN (3/5).

(Toval <i>et al.</i> 02a) [343]	<i>Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach.</i>
Citas: 35 al 47 / 47	Daniel Mellado, Jesús Rodríguez, Eduardo Fernández-Medina, Mario Piattini (2009). <i>Automated Support for Security Requirements Engineering in Software Product Line Domain Engineering. International Conference on Availability, Reliability and Security (ARES'09).</i> Fukuoka, Japan.
	Sharon Nachtigal (2009). <i>E-business Information Systems Security Design Paradigm and Model. Technical Report RHUL-MA-2009-16.</i> Department of Mathematics, Royal Holloway, University of London.
	Atsushi Ohnishi (2009). <i>Common Criteria Based Security Scenario Verification. Software and Data Technologies. Communications in Computer and Information Science.</i> Volume 47, Part 3, 37-47,
	Yu-Ping Ou Yang, How-Ming Shieh, Jun-Der Leu, Gwo-Hshiung Tzeng (2009). <i>A Vikor-Based Multiple Criteria Decision Method for Improving Information Security Risk. International Journal of Information Technology & Decision Making.</i> Vol. 8 (2): 267- 287.
	Jesús Rodríguez, Daniel Mellado, Eduardo Fernández-Medina, Mario Piattini (2009). <i>Gestión Automatizada de Requisitos de Seguridad para Proyectos de Desarrollo de Líneas de Producto Software. V Congreso Iberoamericano de Seguridad Informática (CIBSI 2009).</i> Montevideo, Uruguay.
	I-Hsin Chou, Chin-Feng Fan (2010). <i>Regulatory-based development processes for software security in nuclear safety systems. Progress in Nuclear Energy.</i> Vol. 52 (4): 395-402.
	Siv Hilde Houmb, Shareeful Islam, Eric Knauss, Jan Jürjens and Kurt Schneider (2010). <i>Eliciting security requirements and tracing them to design: an integration of Common Criteria, heuristics, and UMLsec. Requirements Engineering.</i> Vol. 15 (1): 63-93. Special Sigues - Security Requirements Engineering.
	Daniel Mellado, Carlos Blanco, Luis E. Sánchez, Eduardo Fernández-Medina (2010). <i>A systematic review of security requirements engineering. Computer Standards & Interfaces.</i> Vol. 32 (4): 153-165.
	Daniel Mellado, Eduardo Fernández-Medina, Mario Piattini (2010). <i>Security requirements engineering framework for software product lines. Information and Software Technology.</i> Vol. 52 (10): 1094-1117.
	David G. Rosado, Eduardo Fernández-Medina, Javier López, Mario Piattini (2010). <i>Analysis of Secure Mobile Grid Systems: A systematic approach. Information and Software Technology.</i> Vol. 52 (5): 517-536.
	Aleksander Dikanski, Sebastian Abeck (2011). <i>A View-based Approach for Service-Oriented Security Architecture Specification. Sixth International Conference on Internet and Web Applications and Services (ICIW 2011).</i> St. Maarten, The Netherlands Antilles.
	Dorcas Adebola Babatunde, Mohamad Hisyam Selamat (2012). <i>Investigating Information Security Management and Its Influencing Factors in the Nigerian Banking Industry: a Conceptual Model. International Journal on Social Science, Economics and Art.</i> Vol. 2 (2): 55-59.
	Indu Singh, Manoj Kumar (2012). <i>Evaluation of approaches for designing secure data warehouse. International Conference on Advances in Computing, Communications and Informatics (ICACCT'12).</i> Chennai, India.

Tabla 62 Trabajos que referencian a SIREN (4/5).

(Toval <i>et al.</i> 08) [339]	<i>Eight Key Issues for an Effective Reuse-based Requirements Process.</i>
Citas: 1 / 1	Pedro Sánchez, Diego Alonso, Francisca Rosique, Bárbara Álvarez, Juan Ángel Pastor (2011). <i>Introducing safety requirements traceability support in model-driven development of robotic applications.</i> IEEE Transactions on Computers. Vol. 60 (8): 1059-1071.
(Toval <i>et al.</i> 01b) [341]	<i>SIREN: un Proceso de Ingeniería de Requisitos Basado en Reutilización</i>
Citas: 2 / 2	Maria Cristina Carreón Suárez del Real (2011). <i>Construcción de un catálogo de patrones de requisitos funcionales para ERP.</i> Master's Thesis. Universidad Politécnica de Cataluña. Directores: Xavier Franch y María Carme Quer. Carlos M. Zapata J., J. Fernando Arango Isaza (2004). <i>Alineación entre metas organizacionales y elicitación de requisitos del software.</i> Dyna. Vol. 71 (143): 101-110. Universidad Nacional de Colombia. Medellín, Colombia.
(Lasheras <i>et al.</i> 04a) [177]	<i>Soporte automatizado a la reutilización de requisitos</i>
Citas: 1 / 1	Daniel Mellado, Moisés Rodríguez, Eduardo Fernández-Medina, Mario Piattini (2007). <i>Soporte Automatizado a la Ingeniería de Requisitos de Seguridad.</i> Conferencia Iberoamericana de Software Engineering (CIBSE 2007). Isla de Marguerita, Venezuela.

Tabla 63 Trabajos que referencian a SIREN (5/5).

9.4.9.2 Citas a los trabajos relacionados con el enfoque de IR dirigido por modelos

(Vicente-Chicote <i>et al.</i> 07a) [348]	<i>REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting</i>
Citas: 1 al 5 / 16	Arda Goknil, Ivan Kurtev, Klaas van den Berg (2008). <i>A metamodeling approach for reasoning about requirements.</i> 4th European Conference Model Driven Architecture - Foundations and Applications (ECMDA-FA 2008). Berlin, Germany. LNCS 5095. Arda Goknil, Ivan Kurtev, Klaas van den Berg (2008). <i>Change Impact Analysis based on Formalization of Trace Relations for Requirements.</i> ECMDA Traceability Workshop (ECMDA-TW). Berlin, Germany. David Ameller (2009). <i>Considering Non-Functional Requirements in Model-Driven Engineering.</i> Master's Thesis. Departamento de Lenguajes y Sistemas Informáticos. Universidad Politécnica de Cataluña. Director: Xavier Franch. R.S.A. van Domburg (2009). <i>Empirical evaluation of change impact predictions using a requirements management tool with formal relation types. A quasi-experiment.</i> Master's Thesis. University of Twente. Jan Willem Veldhuis (2009). <i>Tool support for a metamodeling approach for reasoning about requirements.</i> Master's Thesis. University of Twente.

Tabla 64 Trabajos que referencian a REMM (1/3).

(Vicente-Chicote <i>et al.</i> 07a) [348]	<i>REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting</i>
Citas: 6 al 16 / 16	José Bocanegra, Joaquín Peña, Antonio Ruiz-Cortés (2010). <i>Una Aproximación MDD para Agilizar el Modelado de Negocio y su Especificación en Entornos Interorganizacionales. XIII Congreso Iberoamericano en Software Engineering (CIbSE'10)</i> . Cuenca, Ecuador.
	Arda Goknil, Ivan Kurtev, Klaas van den Berg and Jan-Willem Veldhuis (2010). <i>Semantics of trace relations in requirements models for consistency checking and inferencing. Software and Systems Modeling</i> . Vol. 10 (1): 31-54.
	Chetna Gupta, Yogesh Singh, Durg Singh Chauhan (2010). <i>Dependency based Process Model for Impact Analysis: A Requirement Engineering Perspectiva. International Journal of Computer Applications</i> . Vol. 6 (6): 28-30.
	Nicolas Sannier, Benoît Baudry (2011). <i>Défis pour la variabilité et la traçabilité des exigences en ingénierie système. INFORSID 2011</i> . Lille.
	Pedro Sánchez, Diego Alonso, Francisca Rosique, Bárbara Álvarez, Juan Ángel Pastor (2011). <i>Introducing safety requirements traceability support in model-driven development of robotic applications. IEEE Transactions on Computers</i> . Vol. 60 (8): 1059-1071.
	Pedro Sánchez, Manuel Jiménez, Francisca Rosique, Bárbara Álvarez, Andrés Iborra (2011). <i>A framework for developing home automation systems: From requirements to code. Journal of Systems and Software</i> . Vol. 84 (6): 1008-1021.
	Soheila Bashardoust Tajali, Vojislav D. Radonjic, Jean-Pierre Corriveau (2011). <i>Challenges of Variability in Model-Driven and Transformational Approaches: A Systematic Survey. Ninth Working IEEE/IFIP Conference on Software Architecture (WICSA'11)</i> . Boulder, Colorado, USA.
	Martin Monperrus, Benoit Baudry, Joël Champeau, Brigitte Hoeltzner, Jean-Marc Jézéquel (2011). <i>Automated measurement of models of requirements. Software Quality Journal</i> .
	Sebastian Adam (2012). <i>Providing Software Product Line Knowledge to Requirements Engineers - A Template for Elicitation Instructions. 18th Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2012)</i> . Essen, Germany. LNCS 7195, 147-164.
	Zoya Durdik, Ralf Reussner (2012). <i>Position paper: approach for architectural design and modelling with documented design decisions (ADMD3). 8th International ACM SIGSOFT Conference on Quality of Software Architectures (QoSA'12)</i> . Bertinoro, Italy.
	Nicolas Sannier, Benoit Baudry (2012). <i>Toward Multilevel Textual Requirements Traceability Using Model-Driven Engineering and Information Retrieval. Second International Model-Driven Requirements Engineering Workshop (MoDRE) junto con RE'12</i> . Chicago, Illinois, USA.

Tabla 65 Trabajos que referencian a REMM (2/3).

(Moros <i>et al.</i> 08b) [234]	Metamodeling Variability to Enable Requirements Reuse
Citas: 5 / 5	Soheila Bashardoust Tajali, Vojislav D. Radonjic, Jean-Pierre Corriveau (2011). <i>Challenges of Variability in Model-Driven and Transformational Approaches: A Systematic Survey</i> . Ninth Working IEEE/IFIP Conference on Software Architecture (WICSA'11) . Boulder, Colorado, USA.
	Jose Andre Dorigan, Rodolfo Miranda de Barros (2012). <i>A Model of Requirements Engineering Process for Standardization and Quality Increase</i> . Applied Computing 2012. Madrid.
	Xavier Franch, Cristina Palomares, Carme Quer, Samuel Renault and François De Lazzer (2010). <i>A Metamodel for Software Requirement Patterns</i> . 16th International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ 2010) . Essen, Germany. LNCS 6182.
	Samuel Renault, Óscar Méndez Bonilla, Javier Franch Gutiérrez, Maria Carme Quer Bosor (2011). <i>A pattern-based method for building requirements documents in call-for-tender processes</i> . International Journal of Computer Science & Applications . Vol. 6 (5): 175-202.
	Samuel Renault, Óscar Méndez Bonilla, Javier Franch Gutiérrez, Maria Carme Quer Bosor (2009). <i>PABRE: Pattern-based Requirements Elicitation</i> . Third International Conference on Research Challenges in Information Science (RCIS 2009) . Fès, Morocco.

Tabla 66 Trabajos que referencian a REMM (3/3).

9.5. Líneas de trabajo futuro

A lo largo del desarrollo de esta tesis doctoral hemos ido poniendo de relieve aspectos de la propuesta que necesitaban ser abordados con posterioridad y que debemos retomar como trabajo futuro más inmediato. Sobre todo, los aspectos destacados por los encuestados en el proceso de validación (Sección 8.7). Las líneas de trabajo futuras incluyen:

- *Completar el catálogo HAREL*. Actualmente el catálogo de requisitos reutilizables para los sistemas domóticos incluye los requisitos que hemos considerado típicos de este tipo de aplicaciones: luz, persianas, toldos, alarmas, climatización. Somos conscientes de que este catálogo debe ser ampliado para considerar otros aspectos de instalación, como por ejemplo, la definición de escenas. Una escena permite configurar la ejecución de varios servicios de unidades funcionales de forma secuencial con una única acción. Por ejemplo, se puede crear una escena "Presentación" que se activa desde un pulsador y ejecuta una serie de servicios preestablecidos con antelación, tales como, bajar persiana, bajar intensidad de las luces y bajar pantalla de proyección.
- *Definición de nuevos requisitos no incluidos en el catálogo*. El catálogo de requisitos HAREL funciona como un lenguaje restringido para la especificación de los requisitos de los sistemas domóticos, de manera que cualquier nueva aplicación se tiene que definir, por completo, reutilizando los requisitos del

catálogo. Dada la incompletitud del catálogo actual, algunos de los expertos en el dominio, encuestados en la fase de validación, apuntaban la posibilidad de incluir en la especificación de requisitos nuevos requisitos no contemplados en el catálogo. Actualmente, aunque el editor de modelos de requisitos *con* reutilización de REMM-Studio^{*} (Sección 7.3.3) permite la definición de nuevos requisitos, el problema es que estos nuevos requisitos no tendrían definida la regla de transformación que permitiera generar el modelo de la aplicación (modelo conforme al DSL Habitation). En consecuencia, la inclusión de un nuevo requisito tendría que hacerse añadiendo previamente este requisito al catálogo HAREL y definiendo la regla de transformación. Deberíamos estudiar una forma ágil de llevar a cabo este proceso, teniendo en cuenta que requiere, no sólo del ingeniero de requisitos, sino del experto en el dominio que tenga claro cuál es el diseño, utilizando los conceptos del dominio (DSL), asociado al requisito nuevo que se está elicitando.

- *Gestión de cambios en los requisitos.* No parece que exista ningún problema si el proceso ReToCo se aplica de forma secuencial y, de forma resumida, del modelo de requisitos se genera el modelo de la aplicación y a partir de éste el código. No obstante, son de sobra conocidas las ventajas de abordar el desarrollo de un sistema de forma iterativa y, es más que probable, que haya que volver sobre la especificación de requisitos una vez que se han generado ya algunos modelos de la aplicación. Los modelos iniciales de la aplicación puede incluso que hayan sido refinados utilizando el editor del DSL. Estos cambios realizados en los modelos de la aplicación deben mantenerse en futuras ejecuciones de las transformaciones desde los modelos de requisitos. En consecuencia, se debe estudiar el modo de combinar los modelos de la aplicación generados tras el cambio en los requisitos con los modelos de la aplicación obtenidos previamente.
- *Optimización de los modelos generados.* Las reglas de transformación actualmente generan un modelo de la aplicación por cada requisito. Estos modelos podrían combinarse para optimizar el número de unidades funcionales que se necesitan, esto es, que una misma unidad funcional, por ejemplo, el detector de presencia, controle diferentes dispositivos, como la luz y las persianas. En este caso, en la implementación actual se genera un detector de presencia por cada dispositivo que hay que controlar.
- *Aplicar el proceso ReToCo a otros dominios.* Aunque el proceso de ha instanciado con éxito para el caso de los sistemas domóticos, sería necesario comprobar sus bondades al aplicarlo en otros dominios diferentes. Sería conveniente seleccionar un dominio en el que se haya aplicado el enfoque DSM puesto que, de esta forma, dispondríamos del DSL, el editor de modelos de la aplicación y el generador de código. De esta forma, tendríamos que estudiar cómo definir los requisitos de manera que se puedan transformar fácilmente en los modelos de diseño de la aplicación.

Por otro lado, el grupo de investigación GIIS de la que esta doctoranda forma parte, en el marco del proyecto PANGAEA (Sección 1.3.2.6), está trabajando en la definición de un proceso de IR para el Desarrollo Global del Software (DGS). El DGS se está consolidando como uno de los aspectos más relevantes en la investigación en Ingeniería del Software en la década de 2010. Surge hace más de 20 años con las primeras

prácticas de *outsourcing* pero se oficializa como tal en 2006 con la celebración de la primera conferencia internacional sobre desarrollo global ICGSE (*IEEE International Conference on Global Software Engineering*). Desde 2008 se han publicado varias revisiones sistemáticas de la literatura que resumen las principales cuestiones del DGS y que demuestran que se trata de un campo de plena actualidad que presenta nuevos problemas y desafíos constantemente. En este contexto, y dada la relevancia de la IR en la Ingeniería del Software, el GIIS ha analizado los problemas, desafíos y riesgos que amenazan la ingeniería de requisitos cuando se realiza en un ambiente global y distribuido de desarrollo, y las soluciones propuestas en la literatura para gestionarlos. Las conclusiones del estudio han servido para crear un repositorio de buenas prácticas en el que se clasifican tales amenazas y se proponen guías o salvaguardas que ayudan a manejarlas, y para definir PANGEA (*Process for globAl requiremeNts enGinEering and quALity*), una extensión del método SIREN de reutilización de requisitos adaptada al DGS que materializará la implantación de las salvaguardas propuestas en el repositorio de buenas prácticas.

En esta línea, sería interesante aplicar las técnicas del DSDM para definir un modelo de referencia (metamodelo) de requisitos unificado para el DGS, con un triple propósito:

- (1) Establecer formalmente los elementos que forman parte de la especificación de requisitos y el modelo de trazabilidad tanto entre requisitos como entre requisitos y el resto de artefactos que participan en el desarrollo software. Habría que extender REMM, si hiciera falta, para incluir los aspectos propios de la IR en el DGS
- (2) Servir como base para intercambiar las especificaciones de requisitos desarrolladas en cada una de las ubicaciones participantes en el proceso de desarrollo.
- (3) Permitir establecer los modelos de requisitos a utilizar en cada ubicación como vistas de este modelo de referencia teniendo la información recogida en un único modelo subyacente (conforme al modelo de referencia).

Bibliografía y referencias

- [1] ATL Transformation Language. <<http://www.eclipse.org/m2m/atl/>> (accessed Septiembre 2012).
- [2] Attempto Project. Attempto Controlled English (ACE). <<http://attempto.ifi.uzh.ch/site/>> (accessed octubre 2012).
- [3] Attributed Graph Grammar System: A Development Environment for Attributed Graph Transformation Systems. <<http://user.cs.tu-berlin.de/~gragra/agg/>> (accessed septiembre 2012).
- [4] Controlled Natural Languages. <<http://sites.google.com/site/controllednaturallanguage/>> (accessed octubre 2012).
- [5] JET Home Page. <<http://www.eclipse.org/modeling/m2t/?project=jet#jet>> (accessed septiembre 2012).
- [6] Modelica® - A Unified Object-Oriented Language for Systems Modeling. <<https://www.modelica.org/>> (accessed octubre 2012).
- [7] MOFScript Home Page. <<http://www.eclipse.org/gmt/mofscript/>> (accessed Septiembre 2012).
- [8] Página oficial de LimeSurvey. <<http://www.limesurvey.org/es>> (accessed 2012).
- [9] ReDSeeDS project. Requirements-Driven Software Development System <<http://www.redseeds.eu/>> (accessed octubre 2012).
- [10] Requirements Management Tools, Ludwig Consulting Services. <http://www.jiludwig.com/Requirements_Management_Tools.html> (accessed 2012).
- [11] Simulink® Simulación y diseño basado en modelos. <<http://www.mathworks.es/products/simulink/>> (accessed).
- [12] University of Toronto, Canada. GRL - Goal Oriented Requirements Language. <<http://www.cs.toronto.edu/km/GRL/>> (accessed octubre 2012).
- [13] Análisis y Gestión de Riesgos con MAGERIT en la Dirección Regional de Sistemas de Información y Telecomunicaciones. Contrato CARMMA 3142-UMU. CARM (Comunidad Autónoma de la Región de Murcia) y GIS (Grupo de Investigación de Ingeniería del Software, Universidad de Murcia). (1999).
- [14] Technological transfer of a patient management system for the intensive care unit, contrato entre la Universidad de Murcia y la Fundación para la Investigación Biomédica en el Hospital de Getafe (Madrid), (2006-2007).

A

- [15] J.e.A. Aguilar, I. Garrigós, J.-N. Mazón, J. Trujillo, An MDA Approach for Goal-oriented Requirement Analysis in Web Engineering, Journal of Universal Computer Science, 16 (2010) 2475-2494.

- [16] A. Albinet, S. Begoc, J.-L. Boulanger, O. Casse, I. Dal, H. Dubois, F. Lakhal, D. Louar, M.-A. Peraldi-Frati, Y. Sorel, Q.-D. Van, The MeMVaTE_x methodology: from requirements to models in automotive application design, ERTS'08, Toulouse, France, 2008.
- [17] I. Alexander, Requirements Engineering Tool Vendors and Freeware Suppliers. <<http://easyweb.easynet.co.uk/~iany/other/vendors.htm>> (accessed 2012).
- [18] I. Alexander, F. Kiedaisch, Towards Recyclable System Requirements, Ninth Annual IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'02), Lund, Sweden, 2002.
- [19] M. Alférez, A. Moreira, V. Amaral, J. Araújo, Model-Driven Requirements Specification for Software Product Lines, in: Model-Driven Domain Analysis and Software Development. Architectures and Functions, Information Science Reference (an imprint of IGI Global), 2011.
- [20] D.d. Almeida Ferreira, A. Rodriguez da Silva, A Controlled Natural Language Approach for Integrating Requirements and Model-Driven Engineering, Fourth International Conference on Software Engineering Advances, IEEE Computer Society Conference Publishing Services, Porto, Portugal, 2009.
- [21] D. Alonso. Desarrollo de software para robots de servicio : un enfoque dirigido por modelos y orientado a componentes. PhD. Tecnologías de la Información y las Comunicaciones. Universidad Politécnica de Cartagena. 2008.
- [22] B. Álvarez, P. Sánchez, J.A. Pastor, F. Ortiz, An Architectural Framework for Modeling Teleoperated Service Robots, ROBOTICA - Intl. Journal of Information, Education and Research in Robotics and Artificial Intelligence, 24 (2006) 411-418.
- [23] A. Antón, Publication Guide. <<http://www4.ncsu.edu/~aiananton/publications.html>> (accessed 2012).
- [24] G. Antoniol, B. Berenbach, A. Eyged, S. Ferguson, J. Maletic, A. Zisman, Problem Statements and Grand Challenges. COET-GCT-06-01-0.9, Center of Excellence for Traceability, 2006.
- [25] C. Atkinson, T. Kühne, Model-Driven Development: A Metamodeling Foundation, IEEE Software, 20 (2003) 36-41.
- [26] F. Azam, Z. Li, R. Ahmad, Introducing VIP Business Modeling Framework for Innovative, Information-rich and Process-aware E-commerce Web Applications, International Conference on Emerging Technologies (ICET 2006), Peshawar, Pakistan, 2006.

B

- [27] R. Baskerville, Information Systems Design Methods: Implications for Information Systems Development, ACM Computing Surveys, 25 (1993) 375-414.
- [28] L. Bass, P. Clements, P. Donohoe, J. McGregor, L. Northrop, Fourth Product Line Practice Workshop Report. CMU/SEI-2000-TR-002
ESC-TR-2000-002, CMU/SEI, 2000.
- [29] L. Bass, P. Clements, R. Kazman, Software Architecture in Practice (2nd Edition), Addison-Wesley, 2003.
- [30] B. Baudry, C. Nebut, Y.L. Traon, Model-driven Engineering for Requirements Analysis, 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), IEEE Computer Society Press, Maryland, MD, USA, 2007.
- [31] S. Bechhofer, F.v. Harmelen, J. Hendler, I. Horrocks, D.L. McGuinness, P.F. Patel-Schneider, L.A. Stein, OWL Web Ontology Language Reference. W3C Recommendation. <<http://www.w3.org/TR/owl-ref/>> (accessed octubre 2012).
- [32] D. Beuche, A. Birk, H. Dreier, A. Fleischmann, H. Galle, G. Heller, D. Janzen, I. John, R.T. Kolagari, T.v.d. Maßen, A. Wolfram, Using Requirements Management Tools in Software Product Line Engineering: The State of the Practice, 11th International Software Product Line Conference (SPLC'07), 2007.

- [33] J. Bézivin, In Search of a Basic Principle for Model Driven Engineering, UPGRADE (European Journal for the Informatics Professional), V (2004) 21-24.
- [34] E. Biermann, K. Ehrig, C. Ermel, C. Köhler, G. Taentzer, The EMF Model Transformation Framework, in: Applications of Graph Transformations with Industrial Relevance Springer-Verlag Berlin, Heidelberg, 2008, pp. 566 - 567
- [35] G. Booch, J. Rumbaugh, I. Jacobson, The Unified Modeling Language User Guide. 2nd ed., Addison-Wesley, 2005.
- [36] Borland, CALIBER®. <<http://www.borland.com/us/products/caliber/index.html>> (accessed 2012).
- [37] A. Bragança, R.J. Machado, Adopting Computational Independent Models for Derivation of Architectural Requirements of Software Product Lines, Fourth International Workshop on Model-Based Methodologies for Pervasive and Embedded Software (MOMPES'07), IEEE Computer Society Press, Braga, Portugal, 2007.
- [38] T. Breaux, A.I. Antón, Analyzing Regulatory Rules for Privacy and Security Requirements, IEEE Transactions on Software Engineering, 34 (2008) 5-20.
- [39] E. Brottier, B. Baudry, Y.L. Traon, D. Touzet, B. Nicolas, Producing a Global Requirements Model from Multiple Requirement Specifications, 11th IEEE International Enterprise Distributed Object Computing Conference (EDOC 2007), IEEE Computer Society, Annapolis, Maryland, USA, 2007.
- [40] E. Brottier, Y.L. Traon, B. Nicolas, Composing models at two modeling levels to capture heterogeneous concerns in requirements, 9th International Conference on Software Composition co-located with TOOLS federated conferences 2010, Springer-Verlag, Málaga, Spain, 2010.
- [41] M. Broy, Challenges in automotive software engineering, in: 28th International Conference on Software Engineering, ACM, Shanghai, China, 2006, pp. 33-42.
- [42] B.R. Bryant, J. Gray, M. Mernik, Domain-specific software engineering, in: Proceedings of the FSE/SDP workshop on Future of software engineering research, ACM, Santa Fe, New Mexico, USA, 2010.
- [43] F. Budinski, D. Steinberg, E. Merks, R. Ellersick, T. Grose, Eclipse Modeling Framework, Addison-Wesley Professional, 2003.
- [44] S. Bühne, K. Lauenroth, K. Pohl, Why is it not Sufficient to Model Requirements Variability with Feature Models?, International Workshop on Automotive Requirements Engineering (AuRE 2004) co-located at REO4, Naza University, Nagoya, Japan, 2004.
- [45] S. Bühne, K. Lauenroth, K. Pohl, Modelling Requirements Variability across Product Lines, 13th IEEE International Requirements Engineering Conference, Paris, France, 2005.
- [46] F. Buschmann, R. Meunier, H. Rohnert, P. Sommerlad, M. Stal, Pattern-Oriented Software Architecture. Volume 1: A System of Patterns, England: John Wiley and Sons Ltd, 1996.

C

- [47] R. Capilla, Variability Description in Requirements for Product Family Support, INTERNATIONAL WORKSHOP ON REQUIREMENTS REUSE IN SYSTEM FAMILY ENGINEERING. Co-located with International Conference on Software Reuse, Madrid, Spain, 2004.
- [48] P. Carlshamre, K. Sandahl, M. Lindvall, B. Regnell, J.N.o. Dagc, An Industrial Survey of Requirements Interdependencies in Software Product Release Planning, Fifth International Symposium on Requirements Engineering (RE'01), Toronto, Canada, 2001.
- [49] S. Castano, M.G. Fugini, G. Martella, P. Samarati, Database Security, Addison-Wesley, 1994.
- [50] J. Castro, M. Lucena, C. Silva, F. Alencar, E. Santos, J. Pimentel, Changing attitudes towards the generation of architectural models, Journal of Systems and Software, 85 (2012) 463-479.

- [51] V.D. Castro, E. Marcos, J.M. Vara, Applying CIM-to-PIM model transformations for the service-oriented development of information systems, *Information and Software Technology*, 53 (2011) 87-105.
- [52] R. Cerón, J.C. Dueñas, E. Serrano, R. Capilla, A Meta-model for Requirements Engineering in System Family Context for Software Process Improvement using CMMI, 6th International Conference on Product Focused Software Process Improvement. LNCS 3547 Oulu, Finland, 2005, pp. 173-188.
- [53] P. Clements, L. Northrop, *Software Product Lines. Practice and Patterns*, Addison-Wesley, 2002.
- [54] P. Colombo, F. Khendek, L. Lavazza, Bridging the gap between requirements and design: An approach based on Problem Frames and SysML, *The Journal of Systems and Software*, 85 (2012) 717-745.
- [55] J.L. Cybulsky, K. Reed, *Requirements Classification and Reuse: Crossing Domains Boundaries*, ICSR'2000, Springer-Verlag, Vienna, Austria, 2000, pp. 190-210.
- [56] K. Czarnecki, M. Antkiewicz, Mapping Features to Models: A Template Approach Based on Superimposed Variants. LNCS 3676/2005 (2005) 422-437.
- [57] K. Czarnecki, S. Helsen, Classification of Model Transformation Approaches, OOPSLA'03 Workshop on Generative Techniques in the Context of Model-Driven Architecture Anaheim, California, 2003.
- [58] J. Champeau, E. Rochefort, *Model Engineering and Traceability*, UML 2003. SIVOEES-MDA Workshop, San Francisco. California, 2003.
- [59] X. Chen, Google Scholar's Dramatic Coverage Improvement Five Years after Debut, *Serials Review*, (2010).
- [60] B.H.C. Cheng, J.M. Atlee, Research Directions in Requirements Engineering, International Conference on Software Engineering (ICSE'07), Minneapolis, MN 2007.
- [61] Y. Chernak, Building Foundation for Structured Requirements. Aspect-oriented Requirements Engineering Explained- Part 1, in: Requirements Networking Group (RQNG), 2009.
- [62] Y. Chernak, Building Foundation for Structured Requirements. Aspect-oriented Requirements Engineering Explained- Part 2, in: Requirements Networking Group (RQNG), 2009.
- [63] Y. Chernak, Requirements Reuse: the State of the Practice. <http://www.requirementsnetwork.com/system/files/Chernak_Requirements%20Reuse_0.pdf> (accessed February 2012).
- [64] L. Chung, Dealing with security requirements during the development of information systems, 5th Intl. Conf. on Advanced Information Systems Engineering (CAISE'93), LNCS 685. Springer, Berlin, Germany, 1993.
- [65] L. Chung, B.A. Nixon, E. Yu, J. Mylopoulos, *Non-Functional Requirements in Software Engineering*, Kluwer Academic Publishers, 2000.

D

- [66] A.G. Dahlstedt, A. Persson, Requirements Interdependencies- Moulding the State of Research into a Research Agenda, Ninth International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ '03) In connection with: CAiSE'03, Klagenfurt/Velden, Austria, 2003.
- [67] A.G. Dahlstedt, A. Persson, Requirements Interdependencies: State of the Art and Future, in: A. Aurum, C. Wohlin (Eds.) *Engineering and Managing Software Requirements*, Springer, 2005.

- [68] D. Damian, J. Chisan, An Empirical Study of the Complex Relationships between Requirements Engineering Processes and Other Processes that Lead to Payoffs in Productivity, Quality, and Risk Management, *IEEE Transaction on Software Engineering*, 32 (2006).
- [69] H.L. Dang, H. Dubois, S. Gérard, Towards a traceability model in a MARTE-based methodology for real-time embedded systems, *Innovations in Systems and Software Engineering*. Springer London, 4 (2008) 189-193.
- [70] A. Dardenne, A.v. Lamsweerde, S. Fickas, Goal-Directed Requirements Adquisition, *Science of Computer Programming*, 20 (1993).
- [71] A. Davis, *JERM: Just Enough Requirements Management*, Dorset House Publishing, 2005.
- [72] A.M. Davis, *201 Principles of Software Development*, McGraw-Hill, 1995.
- [73] N. Debnath, M.C. Leonardi, M.V. Mauco, G. Montejano, D. Riesco, Improving Model Driven Architecture with Requirements Models, *Fifth International Conference on Information Technology: New Generations*, IEEE, Las Vegas, USA, 2008.
- [74] C. Denger, D.M. Berry, E. Kamsties, Higher Quality Requirements Specifications through Natural Language Patterns, *IEEE International Conference on Software-Science, Technology & Engineering (SWSTE '03)*, Herzelia, Israel, 2003.
- [75] J. Dick, Requirements Guide For All (REGAL): An INCOSE Initiative, I.C. Society (Ed.) 14th IEEE International Requirements Engineering Conference (RE'06), Minneapolis/St. Paul, Minnesota, USA, 2006.
- [76] R. Dömgies, K. Pohl, Adapting traceability environments to project-specific needs, *Communications of the ACM*, 41 (1998) 54 - 62.
- [77] H. Dubois, M.-A. Peraldi-Frati, F. Lakhali, A Model for Requirements Traceability in a Heterogeneous Model-Based Design Process: Application to Automotive Embedded Systems, *5th IEEE International Conference on Engineering of Complex Computer Systems*, Oxford, United Kingdom, 2010.
- [78] A. Duque, J. Lasheras, A. Toval, ECAPRIS: Metodología ágil de medición de calidad y productividad en PyMEs, *XIV Jornadas de Ingeniería del Software y Bases de Datos (JISBD2009)* San Sebastián, Spain, 2009.

E

- [79] Eclipse, Generative Modeling Technologies (GMT) project. <<http://www.eclipse.org/gmt/>> (accessed 2012).
- [80] Eclipse, Graphical Modeling Project (GMP). <<http://www.eclipse.org/modeling/gmp/>> (accessed 2012).
- [81] M.J. Escalona, G. Aragón, NDT. A Model-Driven Approach for Web Requirements, *IEEE Transactions on Software Engineering*, 34 (2008) 377-390.
- [82] M.J. Escalona, N. Koch, Metamodeling Requirements of Web Systems, *International Conference on Web Information System and Technologies (WEBIST 2006)*, Setúbal, Portugal, 2006.
- [83] S. España, A. González, Ó. Pastor, Communication Analysis: a requirements engineering method for information systems, *CAiSE'09*, Springer. LNCS vol. 5565, Amsterdam, The Netherlands, 2009, pp. 530-545.

F

- [84] J.-M. Favre, Towards a Basic Theory to Model Model Driven Engineering, *Workshop on Software Model Engineering (WISME 2004) joint event with UML2004*, Lisbon, Portugal, 2004.

- [85] P.H. Feiler, D.P. Gluch, J.J. Hudak, The Architecture Analysis & Design Language (AADL): An Introduction. CMU/SEI-2006-TN-011, SEI, Carnegie Mellon University, 2006.
- [86] N.E. Fenton, M. Neil, A Strategy for Improving Safety Related Software Engineering Standards, *IEEE Transactions on Software Engineering*, 24 (1998) 1002-1014.
- [87] E. Fernández-Medina, J. Jurjens, J. Trujillo, S. Jajodia, Model-Driven Development for secure information systems. Editorial, *Information and Software Technology*, 51 (2009).
- [88] A. Fernández, E. Insfrán, S. Abrahão, Usability evaluation methods for the web: A systematic mapping study, *Information and Software Technology*, 53 (2011) 789-817.
- [89] C. Fernández, Andrés Iborra, B. Álvarez, J.A. Pastor, P. Sánchez, J.M. Fernández-Meroño, N. Ortega, Co-operative Robots in the Ship Repair Industry, *IEEE Robotics and Automation Magazine*, 12 (2005) 65-77.
- [90] D.F. Ferraiolo, R. Sandhu, S. Gavrila, D.R. Kuhn, R. Chandramouli, Proposed NIST standard for role-based access control, *ACM Transactions on Information and System Security* 4(2001) 224-274.
- [91] D. Firesmith, Modern Requirements Specification, *Journal of Object Technology*, 2 (2003) 53-64.
- [92] D.G. Firesmith, Specifying Reusable Security Requirements, *Journal of Object Technology*, 3 (2004) 61-75.
- [93] J. Fons, V. Pelechano, M. Albert, O. Pastor, Development of Web Applications from Web Enhanced Conceptual Schemas, 22nd International Conference on Conceptual Modeling, Chicago, Illinois, 2003.
- [94] L. Fuentes, A. Vallecillo, An Introduction to UML Profiles, *UPGRADE (The European Journal for the Informatics Professional)*, 5 (2004) 5-13.

G

- [95] J. García, M.J. Ortín, B. Moros, J. Nicolás, Modelado de Casos de Uso y Conceptual a partir del Modelado del Negocio V Workshop Menhir, Granada, 2000.
- [96] J. García, M.J. Ortín, B. Moros, J. Nicolás, El Modelo de Negocios como base del Modelo de Requisitos, *Jornadas de Ingeniería de Requisitos Aplicada*, Sevilla, 2001.
- [97] J. García, M.J. Ortín, B. Moros, J. Nicolás, Transforming de OOram Three Model Architecture into a UML-based Process, *Journal of Object Technology (JOT)*, 1 (2002) 119-136.
- [98] J. García, M.J. Ortín, B. Moros, J. Nicolás, A. Toval, De los procesos de negocio a los casos de uso, *V Jornadas de Ingeniería del Software y Bases de Datos (JISBD'00)*, Valladolid, 2000.
- [99] J. García, M.J. Ortín, B. Moros, J. Nicolás, A. Toval, Towards Use Case and Conceptual Models through Business Modeling, L.S. Laender AHF, Storey VC (Ed.) 19th International Conference on Conceptual Modeling, ER'2000., Springer, Salt Lake City (Utah, USA), 2000, pp. 281-294.
- [100] J. García, M.J. Ortín, B. Moros, J. Nicolás, A. Toval, Transforming the OOram Three Model Architecture into a UML-based Process, *Thirty Fourth Hawaii International Conference on System Science (HICSS-34)*, Maui (Hawaii, USA), 2001.
- [101] D. Garlan, R. Monroe, D. Wile, Acme: an architecture description interchange language, *Conference of the Centre for Advanced Studies on Collaborative research (CASCON'97)*, Toronto, Ontario, Canada, 1997.
- [102] E. Geisberger, J. Grünbauer, B. Schätz, A Model-Based Approach To Requirements Analysis, in: E. Brinksma, D. Harel, A. Mader, P. Stevens, R. Wieringa (Eds.) *Methods for Modelling Software Systems (MMOSS)*, Internationales Begegnungs- und Forschungszentrum für Informatik (IBFI), Schloss Dagstuhl, Germany, Dagstuhl, Germany, 2007.
- [103] R.L. Glass, *Software Runaways: Monumental Disasters*, Prentice Hall, 1998.
- [104] R.L. Glass, *Software Engineering: Facts and Fallacies*, Addison-Wesley, 2002.

- [105] J. Goguen, *Introducing OBJ3*. Technical report. SRI-CSL-88-9, SRI International, Computer Science Laboratory, 1988.
- [106] A. Goknil, I. Kurtev, K.v.d. Berg, *Change Impact Analysis based on Formalization of Trace Relations for Requirements*, 4th ECMDA Traceability Workshop, Berlin, Germany, 2008.
- [107] A. Goknil, I. Kurtev, K.v.d. Berg, *A Metamodeling Approach for Reasoning about Requirements*, 4th European Conference on Model Driven Architecture - Foundations and Applications. LNCS 5095, Berlin, Germany, 2008, pp. 310 - 325.
- [108] H. Gomaa, *Designing Software Product Lines with UML: From Use Cases to Pattern-Based Software Architectures*, Addison Wesley Professional, 2004.
- [109] A. González, S. España, M. Ruiz, Ó. Pastor, *Systematic Derivation of Class Diagrams from Communication-Oriented Business Process Models*, Fifth IEEE International Conference on Research Challenges in Information Science, Gosier, Guadeloupe, France, 2011.
- [110] J. Gordijn, H. Akkermans, *Value based requirements engineering: Exploring innovative e-commerce idea*, *Requirements Engineering Journal*, 8 (2003) 114-134.
- [111] O. Gotel, A. Finkelstein, *An Analysis of the Requirements Traceability Problem*, 1st International Conference on Requirements Engineering (ICRE'94), Colorado Springs, 1994, pp. 94-101.
- [112] O. Gotel, A. Finkelstein, *Extended Requirements Traceability: Results of an Industrial Case Study*, 3rd International Symposium on Requirements Engineering (RE '97), IEEE Computer Society Press Annapolis, Maryland, USA, 1997.
- [113] J. Greenfield, K. Short, S. Cook, S. Kent, *Software Factories: Assembling Applications with Patterns, Models, Frameworks, and Tools*, Wiley, 2004.
- [114] C. Gutiérrez. *PWSSec: Proceso para la Seguridad en Sistemas basados en Servicios Web* Universidad de Castilla-La Mancha. 2006.
- [115] C. Gutiérrez, E. Fernández-Medina, M. Piattini, *Web Services Security: is the problem solved?*, *Information Systems Security*, 13 (2004) 22-31.
- [116] C. Gutiérrez, E. Fernandez-Medina, M. Piattini, B. Moros, A. Toval, *Security Requirements for Web Services Based on SIREN*, *Symposium on Requirements Engineering for Information Security*, Paris, France, 2005.
- [117] C. Gutiérrez, B. Moros, E. Fernandez-Medina, A. Toval, M. Piattini, *Requisitos de Seguridad de Servicios Web en el marco de SIREN*, II Taller de Seguridad en Ingeniería del Software y Bases de Datos, Deusto, Bilbao, 2005.

H

- [118] B. Hailpern, P. Tarr, *Model-driven development: The good, the bad, and the ugly*, *IBM SYSTEMS JOURNAL*, 45 (2006) 451-461.
- [119] A. Harbouche, M. Erradi, A. Mokhtari, *A transformation approach for collaboration based requirement models*, *International Journal of Software Engineering & Applications*, 3 (2012) 47-60.
- [120] E. Herzog, A. Pandikow, *SysML - an Assessment*, 15th INCOSE International Symposium, 2005.
- [121] C. Hofmeister, *Applied Software Architecture*, Addison Wesley, 2000.
- [122] E. Hull, K. Jackson, J. Dick, *Requirements Engineering*. 2nd edition, Springer, 2005.

I

- [123] IBM, IBM® Rational® DOORS®. <<http://www-01.ibm.com/software/awdtools/doors/>> (accessed 2012).
- [124] IBM, IBM® Rational® RequisitePro® <<http://www-01.ibm.com/software/awdtools/reqpro/>> (accessed 2012).
- [125] IBM, Business Process Execution Language for Web Services version 1.1. <<http://www.ibm.com/developerworks/library/specification/ws-bpel/>> (accessed septiembre 2012).
- [126] A. Iborra, D. Alonso, F. Ortiz, J.A. Pastor, P. Sánchez, B. Alvarez, Design of Service Robots, IEEE Robotics & Automation, 16 (2009) 24-33.
- [127] IEEE, IEEE Std 610.12-1990. IEEE Standard Glossary of Software Engineering Terminology. 1990.
- [128] IEEE, IEEE Std 1233, 1998 Edition, IEEE Guide for Developing System Requirements Specifications, in: IEEE Standards Software Engineering. Volumen One: Customer and Terminology Standards, The Institute of Electrical and Electronics Engineers, Inc., 1999.
- [129] IEEE, IEEE/EIA, Std 12207.0-1996. Industry Implementation of International Standard. ISO/IEC 12207: 1995. Standard for Information Technology-Software Life Cycle Processes, in: Volume 1: Customer and Terminology Standards, The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Std. Collection, 1999.
- [130] IEEE, IEEE/EIA, Std 12207.1-1997. Guide for Information Technology - Software life cycle processes - Life cycle data, in: Volume 1: Customer and Terminology Standards, The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Std. Collection, 1999.
- [131] IEEE, Std 830-1998 Guide to Software Requirements Specifications (ANSI), in: Volume 4: Resource and Technique Standards, The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Standards Collection, 1999.
- [132] IEEE, Std 1233-1998 Guide for Developing System Requirements Specifications, in: Volume 1: Customer and Terminology Standards, The Institute of Electrical and Electronics Engineers, Inc. IEEE Software Engineering Standards Collection, 1999.
- [133] IEEE, Guide to the Software Engineering Body of Knowledge. <<http://www.computer.org/portal/web/swebok/htmlformat>> (accessed 2012).
- [134] INCOSE, Requirements Working Group Web site. <<http://www.incose.org/practice/techactivities/wg/rqmts/>> (accessed octubre 2012).
- [135] INCOSE, Systems Engineering Handbook. A "how to" guide for all Engineers. Version 2.0, 2000.
- [136] INCOSE, Requirements Management Tools Survey. <<http://www.incose.org/ProductsPubs/products/toolsdatabase.aspx>> (accessed 2012).
- [137] ISO, ISO 8807:1989. Information processing systems – Open Systems Interconnection – LOTOS – A formal description technique based on the temporal ordering of observational behaviour., (1989).
- [138] ISO/IEC, ISO/IEC 9126: Information technology - Software Product Evaluation - Quality characteristics and guidelines for their use - 1991, in, International Organization for Standardisation, 1991.
- [139] ISO/IEC, ISO/IEC 15408. Evaluation Criteria for Information Technology Security, (1999).
- [140] ISO/IEC, ISO/IEC 27001 Information Technology - Security Techniques - Information Security Management Systems - Requirements, (2005).
- [141] ISO/IEC, ISO/IEC 25010:2011. Systems and software engineering – Systems and software Quality Requirements and Evaluation (SQuARE) – System and software quality models. 2011.
- [142] M. Jackson, Problem Frames: Analysing and Structuring Software Development Problems, Pearson Education, 2000.

J

- [143] I. Jacobson, P.-W. Ng, *Aspect-Oriented Software Development with Use Cases*, Addison-Wesley, 2004.
- [144] E. Jendrock, J. Ball, D. Carson, I. Evans, S. Fordin, K. Haase, The Java EE 5 Tutorial. For Sun Java System Application Server 9.1. <<http://docs.oracle.com/javaee/5/tutorial/doc/>> (accessed Septiembre 2012).
- [145] H. Ji, O. Lenord, D. Schramm, A Model Driven Approach for Requirements Engineering of Industrial Automation Systems, 4th International Workshop on Equation-Based Object-Oriented Modeling Languages and Tools, Zurich, Switzerland, 2011.
- [146] M. Jiang, Z. Ding, From Message Sequence Charts to Service Component Model, in: *Advances in Information Technology and Industry Applications*, Springer Berlin Heidelberg, 2012, pp. 687-695.
- [147] M. Jiménez. Desarrollo de sistemas domóticos utilizando un enfoque dirigido por modelos. Ph.D. Tecnología Electrónica. Universidad Politécnica de Cartagena. 2009.
- [148] M. Jiménez, F. Rosique, P. Sánchez, B. Álvarez, A. Iborra, Habitation: A Domain-Specific Language for Home Automation, *IEEE Software*, 26 (2009) 30-38.
- [149] T. Jingbai, H. Keqing, W. Chong, L. Wei, A Context Awareness Non-functional Requirements Metamodel based on Domain Ontology, *IEEE International Workshop on Semantic Computing and Systems (WSCS '08)*, Huangshan, China, 2008.
- [150] P. Johnson, *Reading in Empirical Evaluation for Budding Software Engineering Researchers*. Collaborative Software Development Laboratory. University of Hawaii, 2005.

K

- [151] H. Kaindl, S. Brinkkemper, J.A. Bubenko, B. Farbey, S.J. Greenspan, C.L. Hetimeyer, J.C.S.d.P. Leite, N.R. Mead, J. Mylopoulos, J. Siddiqi, Requirements Engineering and Technology Transfer: Obstacles, Incentives and Improvement Agenda, *Requirements Engineering*, 7 (2002) 113-123.
- [152] H. Kaindl, J. Falb, Can We Transform Requirements into Architecture? , *The Third International Conference on Software Engineering Advances*, IEEE Computer Society, Sliema, Malta 2008.
- [153] H. Kaindl, M. Smialek, D. Svetinovic, A. Ambroziewicz, J. Bojarski, W. Nowakowski, T. Straszak, H. Schwarz, D. Bildhauer, J. Brogan, K. Mukasa, K. Wolter, T. Krebs, Requirements Specification Language Definition - Defining the ReDSeeDS Languages. EU, ReDSeeDS, Projektnr. IST-2006-033596. D2.4.1, Institute of Computer Technology, Vienna University of Technology, 2007.
- [154] H. Kaindl, P. Wagner, A Unification of the Essence of Goal-Oriented Requirements Engineering Fourth International Conference on Software Engineering Advances (ICSEA '09), IEEE, Porto, Portugal, 2009.
- [155] A. Kalnins, J. Barzdins, E. Celms, Model transformation language MOLA, *Model-Driven Architecture: Foundations and Applications (MDAFA 2004)*, Linköping, Sweden, 2004.
- [156] A. Kalnins, E. Kalnina, E. Celms, A. Sostaks, From Requirements to Code in a Model Driven Way, in: *Advances in Databases and Information Systems*, Springer Berlin / Heidelberg, 2010, pp. 161-168.
- [157] A. Kalnins, E. Kalnina, E. Celms, A. Sostaks, A Model-Driven Path from Requirements to Code, *Latvia University Raksti Nr. 756. Computer Science and Information Technology*, (2010) 33-57.

- [158] K. Kang, S. Cohen, J. Hess, W. Novak, A. Peterson, Feature-Oriented Domain Analysis (FODA) Feasibility Study. CMU/SEI-90-TR-21, Software Engineering Institute, Carnegie Mellon University, 1990.
- [159] K. Kang, S. Kim, J. Lee, E. Shin, M. Huh, FORM: A Feature-Oriented Reuse Method with Domain-Specific Reference Architectures., *Annals of Software Engineering* 5, 5, (1998) 143-168.
- [160] R. Karban, L. Andolfato, M. Zamparelli, Towards Model Re-usability for the development of telescope control systems, 12th International Conference on Accelerator and Large Experimental Control Systems (ICALEPCS 2009), Kobe, Japan, 2009.
- [161] M. Kassab, O. Ormandjieva, M. Daneva, A Metamodel for Tracing Non-functional Requirements, 2009 WRI World Congress on Computer Science and Information Engineering (CSIE'09), Los Angeles, California USA, 2009.
- [162] M. Kasunic, Designing an Effective Survey. CMU/SEI-2005-HB-004, SEI, Carnegie Mellon University, 2005.
- [163] S. Kelly, J.-P. Tolvanen, Domain-Specific Modeling: Enabling Full Code Generation, Wiley-IEEE Computer Society Press, 2008.
- [164] B. Kitchenham, Guidelines for performing Systematic Literature Reviews in Software Enginnering EBSE-2007-01, Software Engineering Group, School of Computer Science and Mathematics, Keele University., 2007.
- [165] B. Kitchenham, S.L. Pfleeger, Principles of Survey Research. Part 2: Designing a Survey, *ACM SIGSOFT Software Engineering Notes* 27 (2002) 18-20.
- [166] B. Kitchenham, S.L. Pfleeger, Principles of Survey Research. Part 3: Constructing a Survey Instrument, *ACM SIGSOFT Software Engineering Notes* 27 (2002) 20-24.
- [167] B. Kitchenham, S.L. Pfleeger, Principles of Survey Research. Part 4: Questionnaire Evaluation, *ACM SIGSOFT Software Engineering Notes* 27 (2002) 20-23.
- [168] B. Kitchenham, S.L. Pfleeger, L. Pickard, P.W. Jones, D.C. Hoaglin, K. El Eman, J. Rosenberg, Preliminary Guidelines for Empirical Research in Software Engineering, *IEEE Transaction on Software Engineering*, 28 (2002) 721-734.
- [169] B. Kitchenham, L. Pickard, S.L. Pfleeger, Case Studies for Method and Tool Evaluation, *IEEE Software*, 12 (1995) 52-62.
- [170] A. Kleppe, J. Warmer, W. Bast, MDA Explained: The Model Driven Architecture™: Practice and Promise, Addison Wesley, 2003.
- [171] A.v. Knethen, B. Paech, F. Kiedaisch, F. Houdek, Systematic Requirements Recycling through Abstraction and Traceability, RE'02, Essen, Germany, 2002, pp. 273-281.
- [172] N. Koch, A. Kraus, The expressive Power of UML-based Web Engineering., 2nd International Workshop on Web-oriented Software Technology (IWWOST 2002), Málaga, Spain, 2002.
- [173] G. Kotonya, I. Sommerville, Requirements Engineering. Processes and Techniques, John Wiley & Sons, 1998.

L

- [174] W. Lam, J.A. MacDermid, A.J. Vickers, Ten Steps Towards Systematic Requirements Reuse, 3rd IEEE International Symposium on Requirements Engineering (RE'97), Annapolis, MD, 1997.
- [175] A.v. Lamsweerde, Goal-Oriented Requirements Engineering: A Guided Tour, 5th IEEE International Symposium on Requirements Engineering, Toronto, 2001, pp. 249-263.
- [176] A.v. Lamsweerde, E. Letier, From Object Orientation to Goal Orientation: A Paradigm Shift for Requirements Engineering, 9th International Workshop on Radical Innovations of Software and Systems Engineering in the Future (RISSEF 2002), Lecture Notes in Computer Science 2941, Venice, Italy, 2002.

- [177] J. Lasheras. Análisis y Prototipado de una herramienta CARE de Soporte al Método SIREN:SirenTool. Facultad de Informática. Universidad de Murcia. 2003.
- [178] J. Lasheras. Marco de trabajo de representación y reuso de requisitos de seguridad. PhD. Dpto de Informática y Sistemas. Universidad de Murcia. 2011.
- [179] J. Lasheras, J. Nicolás, A. Toval, B. Moros, Hacia un Modelo de Dominio de los Sistemas Teleoperados a través de una extensión de SIREN, II Jornadas de Trabajo DYNAMICA, Málaga, 2004.
- [180] J. Lasheras, A. Toval, J. Nicolás, B. Moros, Soporte automatizado a la reutilización de requisitos, VIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'03), Alicante, 2003.
- [181] J. Lasheras, A. Toval, J. Nicolás, B. Moros, Definición de requisitos de seguridad con fines de reutilización, I Taller de Seguridad en Ingeniería del Software y Bases de Datos, Málaga, 2004.
- [182] M. Lázaro, E. Marcos, Research in Software Engineering: Paradigms and Methods, Workshop CAISE'05, 2005.
- [183] I.M. Lazarte, E. Tello-Leal, J. Roa, O. Chiotti, P.D. Villarreal, Model-Driven Development Methodology for B2B Collaborations, 4th IEEE International Enterprise Distributed Object Computing Conference Workshops, Vitória, ES, Brazil, 2010.
- [184] D. Leffingwell, Calculating your return on investment from more effective requirements management. <<http://www-128.ibm.com/developerworks/rational/library/347.html>> (accessed febrero 2012).
- [185] D. Leffingwell, D. Widrig, The Role of Requirements Traceability in System Development, RationalEdge, (2002).
- [186] D. Leffingwell, D. Widrig, Managing Software Requirements: A Use Case Approach, 2nd Edition, Addison Wesley, 2003.
- [187] J.C.S.d.P. Leite, G.D.S. Hadad, J.H. Doorn, G.N. Kaplan, A Scenario Construction Process, Requirements Engineering, 5 (2000) 38-61.
- [188] M.C. Leonardi, N.V. Mauco, Integrating Natural Language Oriented Requirements Models into MDA, VII Workshop on Requirements Engineering (WER 2004), Argentina, 2004.
- [189] P. Letelier, A Framework for Requirements Traceability in UML-based projects, 1st International Workshop on Traceability in Emerging Forms of Software Engineering (TEFSE'02), Edinburgh, U.K., 2002.
- [190] S. Liaskos, L. Jiang, A. Lapouchnian, Y. Wang, Y. Yu, J.C.S.d.P. Leite, J. Mylopoulos, Exploring the Dimensions of Variability: a Requirements Engineering Perspective, K. Pohl, P. Heymans, K.-C. Kang, A. Metzger (Eds.) First International Workshop on Variability Modelling of Software-intensive Systems (VAMOS), Limerick, Ireland, 2007.
- [191] R. Likert, A Technique for the Measurement of Attitudes, Archives of Psychology, 22 (1932) 1-55.
- [192] G. Loniewski, A. Armesto, E. Insfran, An Agile Method for Model-Driven Requirements Engineering, The Sixth International Conference on Software Engineering Advances (ICSEA 2011), Barcelona, Spain, 2011.
- [193] G. Loniewski, A. Armesto, E. Insfran, Incorporating Model-Driven Techniques into Requirements Engineering for the Service-Oriented Development Process, 4th IFIP WG 8.1 Working Conference on Method Engineering (ME 2011), Paris, France, 2011.
- [194] G. Loniewski, E. Insfran, S. Abrahão, A Systematic Review of the Use of Requirements Engineering Techniques in Model-Driven Development, 13th ACM/IEEE International Conference on Model Driven Engineering and Systems (MODELS 2010). LNCS 6395, Oslo, Norway, 2010.
- [195] LOPD, Ley Orgánica 15/1999, de 13 de diciembre, de Protección de Datos de Carácter Personal en España, BOE, 298 (1999).
- [196] F. Losilla, C. Vicente-Chicote, B. Álvarez, A. Iborra, P. Sánchez, Wireless Sensor Network Application Development: An Architecture-Centric MDE Approach, First European Conference on Software Architecture (ECSA 2007), Aranjuez, Spain, 2007.

- [197] K. Lukka, The constructive research approach, in: L.H. Ojala, O-P. (Ed.) Case study research in logistics, Publications of the Turku School of Economics and Business Administration. Series B 1, 2003, pp. 83-101.
- [198] R.R. Lutz, Extending the Product Family Approach to Support Safe Reuse, The Journal of Systems and Software, 53 (2000) 207-217.

M

- [199] M. Mannion, H. Kaindl, Using parameters and discriminants for product line requirements, Systems Engineering, 11 (2007) 61 - 80.
- [200] J. Mansell, A. Bediaga, R. Vogel, K. Mantell, A process framework for the successful adoption of model driven development, Second European conference on Model Driven Architecture: foundations and Applications (ECMDA-FA'06), Bilbao Spain, 2006.
- [201] MAP, Métrica v.3. Metodología de Planificación, Desarrollo y Mantenimiento de Sistemas de Información.
<http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=P60085901274201580632&langPae=es> (accessed 2012).
- [202] MAP, MAGERIT - versión 2. Metodología de Análisis y Gestión de Riesgos de los Sistemas de Información.
<http://administracionelectronica.gob.es/?_nfpb=true&_pageLabel=PAE_PG_CTT_Area_Descargas&langPae=es&iniciativa=184> (accessed 2012).
- [203] V. Marinelli, P.A. Laplante, Requirements Engineering: The State of the Practice Revisited. <<http://www.projectsmart.co.uk/docs/requirements-engineering-the-state-of-the-practice-revisited.pdf>> (accessed febrero 2012).
- [204] F. Marschall, M. Schoenmakers, Towards Model-Based Requirements Engineering for Web-Enabled B2B Applications 10th IEEE International Conference and Workshop on the Engineering of Computer-Based Systems (ECBS'03) Huntsville, AL, USA, 2003.
- [205] M.A. Martínez-Aguilar. Revisión de los Perfiles SIREN de Protección de Datos Personales y de Seguridad y Aplicación a un Caso de Estudio Real. Proyecto Informático. Facultad de Informática. 2005.
- [206] M.A. Martínez-Aguilar. AUDIT-PDP: Un Método de Auditoría de Datos Personales basado en Ingeniería de Requisitos. PhD. Dpto de Informática y Sistemas. Universidad de Murcia. 2012.
- [207] M.A. Martínez, J. Lasheras, J. Nicolás, A. Toval, B. Moros, Análisis de la LOPD y LSSICE, su Impacto en los Sistemas de Información, COMMON Europa/España: IBM Forum, Madrid, 2005.
- [208] R. Matulevicius. Process Support for Requirements Engineering. A Requirements Engineering Tool Evaluation Approach. Department of Computer and Information Science. Faculty of Information Technology, Mathematics and Electrical Engineering. Norwegian University of Science and Technology. 2005.
- [209] J.-N. Mazón, J. Pardillo, J. Trujillo, A Model-Driven Goal-Oriented Requirement Engineering Approach for Data Warehouses, ER 2007, LNCS 4802, Springer-Verlag Berlin Heidelberg, 2007, pp. 255-264.
- [210] C. McClure, Software Reuse Techniques. Adding Reuse to the System Development Process, Prentice Hall, 1997.
- [211] D. Mellado, E. Fernandez-Medina, M. Piattini, Security Requirements Variability for Software Product Lines, 3rd Intl. Conf. on Availability, Reliability and Security (ARES 2008), Barcelona, Spain, 2008.
- [212] N. Mellegård, M. Staron, A Domain Specific Modelling Language for Specifying and Visualizing Requirements, The First International Workshop on Domain Engineering (DE@CAISE09), Amsterdam, The Netherlands, 2009.

- [213] T. Mens, P.V. Gorp, A Taxonomy of Model Transformation, *Electronic Notes in Theoretical Computer Science*, 152 (2006) 125-142.
- [214] D. Merschen, A. Polzer, G. Botterweck, S. Kowalewski, Experiences of applying model-based analysis to support the development of automotive software product lines, 5th Workshop on Variability Modeling of Software-Intensive Systems (VaMoS'11), Namur, Belgium, 2011.
- [215] H. Mili, F. Mili, A. Mili, Reusing Software: Issues and Research Directions, *IEEE Transactions on Software Engineering*, 21 (1995) 528-562.
- [216] A. Mitschke, N. Loughran, B. Josko, S. Häusler, H. Dierks, F. Poppen, C. Alonso, G. Pai, K. Villela, I. Omoronyia, B. Caillaud, A. Benvenista, N. Boulila, H. Dubois, J. Gacnik, S. Köhler, A. Kondeva, T. Kummerow, M. Oertel, P. Rehkop, M. Adedjouma, Definition and exemplification of Requirements Specification Language and Requirements Meta Model. CESAR project deliverable.
<http://www.cesarproject.eu/fileadmin/user_upload/CESAR_D_SP2_R21_M1_v1.000.pdf> (accessed).
- [217] P. Mohagheghi, V. Dehlen, T. Neple, Towards a Tool-Supported Quality Model for Model-Driven Engineering, 3rd Workshop on Quality in Modeling, Göteborg, 2009.
- [218] F. Molina, J. Pardillo, C. Cachero, A. Toval, An MDE Modelling Framework for Measurable Goal-oriented Requirements, *International Journal of Intelligent Systems. Special Issue on 'Goal-Driven Requirements Engineering'*. 25 (2010) 757-783.
- [219] F. Molina, J. Pardillo, A. Toval, Modelling Web-based systems requirements using WRM, 9th International Conference on Web Information Systems Engineering (WISE 2008). LNCS 5176., Auckland, New Zealand, 2008.
- [220] F. Molina, A. Toval, Integrating usability requirements that can be evaluated in design time into Model Driven Engineering of Web Information Systems, *Advances in Engineering Software*, 40 (2009) 1306-1317.
- [221] F. Molina, A. Toval, J. Pardillo, C. Cachero, Towards a requirement-aware Common Web Engineering Metamodel, *Latin American Web Conference (LA-WEB 2008)* Vila Velha, Espírito Santo, Brasil, 2008.
- [222] F.M. Molina. Una propuesta para el tratamiento de requisitos y modelos de calidad en Model Driven Engineering. PhD. Departamento de Informática y Sistemas. Universidad de Murcia. 2010.
- [223] M. Monperrus, B. Baudry, J. Champeau, B. Hoeltzner, J.-M. Jézéquel, Automated Measurement of Models of Requirements, *Software Quality Journal*, Online edition (2011).
- [224] A. Monzón, J.C. Dueñas, Experience-based Approach to Requirements Reuse in Product Families with DOORS®, *International Workshop on Requirements Reuse in System Family Engineering (at Eight International Conference on Software Reuse)*, Madrid, Spain, 2004.
- [225] M. Moon, H.S. Chae, T. Nam, K. Yeom, A Metamodeling Approach to Tracing Variability between Requirements and Architecture in Software Product Lines, 7th IEEE International Conference on Computer and Information Technology (CIT'07), IEEE, Fukushima, Japan, 2007.
- [226] M. Moon, K. Yeom, H.S. Chae, An Approach to Developing Domain Requirements as a Core Asset Based on Commonality and Variability Analysis in a Product Line, *IEEE Trans. on Soft. Eng.*, 31 (2005) 551-569.
- [227] A. Moreira, J. Araujo, J. Whittle, Modeling Volatile Concerns as Aspects, *Conference on Advanced Information Systems Engineering (CAiSE)*, Luxembourg, 2006.
- [228] A. Moreira, A. Rashid, J. Araujo, Multi-Dimensional Separation of Concerns in Requirements Engineering, 13th IEEE International Requirements Engineering Conference, Paris, France, 2005.
- [229] B. Moros, G. Casado, A. Toval, J. Nicolás, J. García, Prototipado de especificaciones OASIS en GemStone, II Jornadas en Ingeniería del Software, San Sebastián, 1997.
- [230] B. Moros, G. Casado, A. Toval, J. Nicolás, J. García, Una experiencia de desarrollo con GemStone, III Jornadas sobre Tecnología de Objetos, Sevilla, 1997.

- [231] B. Moros, J. Nicolás, J. García, A. Toval, Combining Formal Specifications with Design by Contract., *Journal of Object Oriented Programming (JOOP)*, 12 (2000) 16-22.
- [232] B. Moros, A. Toval, F. Rosique, P. Sánchez, Transforming and Tracing Reused Requirements Models to Home Automation Models, *Information and Software Technology*.
- [233] B. Moros, C. Vicente-Chicote, A. Toval, A Model-Driven Engineering Approach to Requirements Engineering: How These Disciplines May Benefit Each Other, 2nd International Conference on Software and Data Technologies (ICSOFT 2007), Barcelona, Spain, 2007.
- [234] B. Moros, C. Vicente-Chicote, A. Toval, Metamodeling Variability to Enable Requirements Reuse, Exploring Modeling Methods for Systems Analysis and Design Workshop in conjunction with CAISE'08, Montpellier, France, 2008.
- [235] B. Moros, C. Vicente-Chicote, A. Toval, REMM-Studio+: Extensiones para Modelar la Variabilidad y Permitir la Reutilización de Requisitos, A. Moreira, M.J. Suárez-Cabal, C.d.l. Riva, J. Tuya (Eds.) XIII Jornadas de Ingeniería del Software y Bases de Datos, Gijón, Spain, 2008, pp. 453-454.
- [236] B. Moros, C. Vicente-Chicote, A. Toval, REMM-Studio+: Modeling Variability to Enable Requirements Reuse, in: Q. Li, S. Spaccapietra, E. Yu, A. Olivé (Eds.) *Conceptual Modeling - ERO8-LNCS 5231*, Springer, 2008, pp. 530-531.
- [237] S. Mosser, M. Blay-Fornarino, R. France, Workflow design using fragment composition: crisis management system design through ADORE, in: *Transactions on aspect-oriented software development VII*, Springer-Verlag Berlin, Heidelberg 2010, pp. 200-233.
- [238] S. Mosser, G. Mussbacher, M. Blay-Fornarino, D. Amyot, From Aspect-oriented Requirements Models to Aspect-oriented Business Process Design Models, 10th International Conference on Aspect-Oriented Software Development, Porto de Galinhas, Pernambuco (Brazil), 2011.
- [239] P.A. Muller, P. Studer, J. Bezivin, Platform Independent Web Application Modeling and development with Netsilon, *Software & System Modeling*, 00 (2005) 1-19.
- [240] G. Mussbacher, D. Amyot, J. Araújo, A. Moreira, Requirements modeling with the aspect-oriented user requirements notation (AoURN): a case study, in: *Transactions on aspect-oriented software development VII. LNCS 6210*, Springer-Verlag Berlin, Heidelberg, 2010, pp. 23-68.

N

- [241] E. Navarro. ATRIUM: Architecture Traced from Requirements applying a Unified Methodology. Computing Systems Department. UCLM. 2007.
- [242] E. Navarro, A. Gómez, P. Letelier, I. Ramos, MORPHEUS: a supporting tool for MDD, 18th Int Conf on Information Systems Development (ISD2009), Springer, Nanchang, China, 2009.
- [243] E. Navarro, J.A. Mocholi, P. Letelier, I. Ramos, A metamodeling approach for requirements specification, *Journal of Computer Information Systems*, 46 (2006) 67-77.
- [244] C.J. Neill, P. Laplante, Requirements Engineering: The State of the Practice, *IEEE Software*. 2003, 20 (2003).
- [245] J. Nicolás. Una propuesta de gestión integrada de modelos y requisitos en líneas de productos software. PhD. Departamento de Informática y Sistemas. Universidad de Murcia. 2009.
- [246] J. Nicolás, J. Lasheras, A. Toval, B. Moros, Ingeniería de Requisitos basada en reutilización: una propuesta de aplicación a los sistemas teleoperados para la limpieza de cascos de buques, III Jornadas de Trabajo DYNAMICA, Almagro, Ciudad Real, 2005.
- [247] J. Nicolás, J. Lasheras, A. Toval, F.J. Ortiz, B. Álvarez, A Collaborative Learning Experience in Modelling the Requirements of Teleoperated Systems for Ship Hull Maintenance, *Learning Software Organizations + Requirements Engineering (LSO+RE 2006)*, Hannover. Alemania, 2006.

- [248] J. Nicolás, J. Lasheras, A. Toval, F.J. Ortiz, B. Álvarez, An integrated domain analysis approach for teleoperated systems, *Requirements Engineering Journal*, 14 (2009) 27-46.
- [249] J. Nicolás, B. Moros, J. Lasheras, A. Toval, SIREN: Un Método Práctico de Ingeniería de Requisitos Basado en Reutilización. Versión 2.0. DIS tech. report, TR DIS 1-2009, Departamento de Informática y Sistemas, Universidad de Murcia, 2009.
- [250] B. Nuseibeh, S. Easterbrook, *Requirements Engineering: A Roadmap, Future of Software Engineering*, Limerick, Ireland, 2000.
- [251] B. Nuseibeh, J. Kramer, A. Finkelstein, A Framework for Expressing the Relationships between Multiple Views in Requirements Specifications, *IEEE Transaction on Software Engineering* 20 (1994) 760-773.

O

- [252] OMG, Committed Companies and Their Products. <<http://www.omg.org/mda/committed-products.htm>> (accessed 2012).
- [253] OMG, MDA Guide Version 1.0.1. <http://www.omg.org/news/meetings/workshops/UML_2003_Manual/00-2_MDA_Guide_v1.0.1.pdf> (accessed February 2012).
- [254] OMG, Meta-Object Facility (MOF) Specification v2.0. Object Management Group, 2004.
- [255] OMG, Object Constraint Language (OCL) Specification v2.0. Object Management Group, 2006.
- [256] OMG, Software & Systems Process Engineering Metamodel specification (SPEM). Version 2.0. <<http://www.omg.org/spec/SPEM/2.0/>> (accessed junio 2012).
- [257] OMG, OMG Systems Modeling Language (OMG SysML™). Verion 1.2 <<http://www.omg.org/spec/SysML/1.2/>> (accessed February 2012).
- [258] OMG, Business Process Model and Notation (BPMN). Version 2.0. <<http://www.omg.org/spec/BPMN/2.0/>> (accessed septiembre 2012).
- [259] OMG, Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification. Version 1.1. <<http://www.omg.org/spec/QVT/1.1>> (accessed February 2012).
- [260] OMG, Requirements Interchange Format (ReqIF). Version 1.0.1. <<http://www.omg.org/spec/ReqIF/1.0.1/PDF/>> (accessed 2012).
- [261] OpenSource-IT.com, Open source MDD & MDD Software. <http://www.opensource-it.com/enterprise_solutions/open_source_mdd_mdd_software> (accessed 2012).
- [262] F. Ortiz, J.A. Pastor, B. Alvarez, A. Iborra, N. Ortega, D. Rodriguez, C. Conesa, Robots for hull ship cleaning, *IEEE Intl. Symposium on Industrial Electronics (ISIE 2007)*, Vigo, Spain, 2007.

P

- [263] R.F. Paige, G.K. Olsen, D.S. Kolovos, S. Zschaler, C. Power, Building Model-Driven Engineering Traceability Classifications, *ECMDA Traceability Workshop (ECMDA-TW)*, Sintef, Trondheim, 2008.
- [264] O. Pastor, J. Gómez, E. Insfrán, V. Pelechano, The OO-method approach for information systems modelling: from object-oriented conceptual modeling to automated programming, *Information Systems*, 26 (2001) 507-534.
- [265] O. Pastor, J.C. Molina, *Model-Driven Architecture in Practice: a Software Production Environment Based on Conceptual Modeling*, Springer, 2007.

- [266] D.E. Perry, A.A. Porter, L.G. Votta, Empirical Studies of Software Engineering: A Roadmap, The 22nd International Conference on Software Engineering (ICSE'00), Limerick, Ireland, 2000.
- [267] S.L. Pfleeger, B. Kitchenham, Principles of Survey Research. Part 1: Turning Lemons into Lemonade, ACM SIGSOFT Software Engineering Notes 26 (2001) 16-18.
- [268] M.F. Pimenta. Viewpoints and Goals: Towards an Integrated Approach. Manuel Filipe Pimenta. Universidade Nova de Lisboa. 2009.
- [269] F.A.C. Pinheiro, Requirements Traceability, in: Perspectives on Software Requirements (Kluwer International Series in Engineering and Computer Science, 753), Kluwer Academic Publishers, 2003.
- [270] M. Pinto, L. Fuentes, J.M. Troya, A Dynamic Component and Aspect-Oriented Platform, Computer Journal, 48 (2005) 401-420.
- [271] P. Pires, F. Delicato, R. C  be, T. Batista, J. Davis, J. Song, Integrating ontologies, model driven, and CNL in a multi-viewed approach for requirements engineering, Requirements Engineering, 16 (2011) 133-160.
- [272] Y. Pisan, Extending Requirements Specifications Using Analogy, ICSE 2000, Limerick, Ireland, 2000.
- [273] K. Pohl, PRO-ART: Enabling Requirements Pre-Traceability, IEEE Int. Conference on Requirements Engineering (ICRE'96), Colorado, USA, 1996.
- [274] K. Pohl, Process-Centered Requirements Engineering, in: Advanced Software Development Series, Research Studie Press Ltd. (RSP)/Wiley & Sons, Hertfordshire, UK, 1996.
- [275] K. Pohl, Requirements Engineering. Fundamentals, Principles and Techniques, Springer, 2010.
- [276] K. Pohl, G. B  ckle, F.J.v.d. Linden, Software Product Line Engineering: Foundations, Principles and Techniques, Springer, 2005.
- [277] K. Pohl, F.v.d. Linden, A. Metzger, Software Product Line Variability Management, 10th International on Software Product Line Conference IEEE Computer Society, Baltimore, Maryland, USA, 2006.
- [278] C. Ponsard, M. Delehay  , Towards a Model-Driven Approach for Mapping Requirements on AADL Architectures, IEEE International Conference on Engineering of Complex Computer Systems, Potsdam, Germany, 2009.
- [279] R.S. Pressman, Software Engineering: A Practitioner's Approach. 7th edition, McGraw-Hill, 2009.
- [280] L. Probasco, D. Leffingwell, Combining Software Requirements with Use Case Modeling. Rational White Paper, 1999.

R

- [281] B. Ramesh, M. Jarke, Toward Reference Models for Requirements Traceability, IEEE Transactions on Software Engineering, 27 (2001) 58-93.
- [282] A. Rashid, Aspect-Oriented Requirements Engineering: An Introduction, 16th IEEE International Requirements Engineering Conference, Barcelona, Spain, 2008.
- [283] A. Rashid, R. Chitchyan, Aspect-oriented requirements engineering: a roadmap, 13th international workshop on Early Aspects (EA'08), Leipzig, Germany, 2008.
- [284] A. Rashid, A. Moreira, J. Araujo, Modularisation and Composition of Aspectual Requirements, 2nd International Conference on Aspect-Oriented Software Development, Boston, MA, USA, 2003.
- [285] R. Ravichandar, J.D. Arthur, M. P  rez-Qui  ones, Pre-Requirement Specification Traceability: Bridging the Complexity Gap through Capabilities, International Symposium on the Grand Challenges in Traceability (GTC '07), Lexington, KY, USA, 2007.

- [286] D.J. Reifer, Is the Software Engineering State of the Practice Getting Closer to the State of the Art?. *IEEE Software* 2003., 20 (2003).
- [287] D.C. Rine, N. Nada, An empirical study of a software reuse reference model, *Information and Software Technology*, 42 (2000) 47-65.
- [288] RMS, Real Decreto 994/1999, de 11 de junio, por el que se aprueba el Reglamento de medidas de seguridad de los ficheros automatizados que contengan datos de carácter personal, *BOE*, 151 (1999) 24241.
- [289] F.J.F. Robert M. Groves, Jr., Mick P. Couper, James M. Lepkowski, Eleanor Singer, Roger Tourangeau, *Survey Methodology*. Second edition, John Wiley & Sons, Inc., 2009.
- [290] J. Robertson, S. Robertson, Requirements Management: a Cinderella Story, *Requirements Engineering Journal*, 5 (2000) 134-136.
- [291] J. Robertson, S. Robertson, VOLERE Requirements Specification Template. Edition 11. <<http://www.volere.co.uk/template.htm>> (accessed Septiembre 2012).
- [292] S. Robertson, Requirements Patterns via Events/Use Cases, http://www.systemsguild.com/GuildSite/SQR/Requirements_Patterns.html, (1996).
- [293] S. Robertson, J. Robertson, *Mastering the Requirements Process* (2nd edition), Addison-Wesley, 2006.
- [294] A. Rodríguez, E. Fernández-Medina, J. Trujillo, M. Piattini, Secure business process model specification through a UML 2.0 activity diagram profile, *Decision Support Systems*, 51 (2011) 446-465.
- [295] A. Rodríguez, I.G.-R.d. Guzmán, E. Fernández-Medina, M. Piattini, Semi-formal transformation of secure business processes into analysis class and use case models: An MDA approach, *Information and Software Technology*, 52 (2010).
- [296] F. Rosique. *Desarrollo integral de aplicaciones domóticas: una perspectiva metodológica*. PhD. Tecnologías de la Información y las Comunicaciones. Universidad Politécnica de Cartagena. 2012.
- [297] G. Rozenberg, *Handbook of Graph Grammars and Computing by Graph Transformation*, World Scientific, Netherlands, 1997.
- [298] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, *Empirical Software Engineering*, 14 (2009) 131-164.

S

- [299] J. Sánchez, J. García, M. Menarguez, RubyTL: A Practical, Extensible Transformation Language, in: *Model Driven Architecture - Foundations and Applications*. LNCS 4066, Springer, 2006, pp. 158-172.
- [300] Ó. Sánchez, F. Molina, J. García-Molina, A. Toval, ModelSec: A Generative Architecture for Model-Driven Security, *Journal of Universal Computer Science*, 15 (2009) 2957-2980.
- [301] P. Sánchez, M. Jiménez, F. Rosique, B. Álvarez, A. Iborra, A framework for developing home automation systems: From requirements to code, *Journal of Systems and Software*, 84 (2011) 1008-1021.
- [302] P. Sánchez, A. Moreira, L. Fuentes, J. Araújo, J. Magno, Model-driven development for early aspects, *Information and Software Technology*, 52 (2010) 249-273.
- [303] P. Sawyer, I. Sommerville, S. Viller, Requirements Process Improvement Through the Phased Introduction of Good Practice, *SOFTWARE PROCESS—Improvement and Practice*, 3 (1997) 19-34.
- [304] B. Schätz, A. Fleischmann, E. Geisberger, M. Pister, *Model-Based Requirements Engineering with AutoRAID*, *INFORMATIK 2005*, Rheinische Friedrich-Wilhelms-Universität Bonn, 2005.
- [305] K. Schmid, I. John, *A Practical Approach To Full-Life Cycle Variability Management*, *International Workshop on Software Variability Management (SVM 2003)*, Portland, Or., 2003.

- [306] K. Schmid, K. Krennrich, M. Eisenbarth, Requirements Management for Product Lines: Extending Professional Tools, 10th International Software Product Line Conference (SPLC'06), 2006.
- [307] D.C. Schmidt, Model-Driven Engineering, IEEE Computer, 39 (2006).
- [308] H. Schwarz, J. Ebert, V. Riediger, A. Winter, Towards Querying of Traceability Information in the Context of Software Evolution, R. Gimnich, U. Kaiser, J. Quante, A. Winter (Eds.) 10th Workshop Software Reengineering 2008, Bad Honnef, Germany, 2008.
- [309] SEI-CMU, CMMI for Development, Version 1.3. <http://www.sei.cmu.edu/reports/10tr033.pdf> (accessed February 2012).
- [310] E. Seidewitz, What Models Mean, IEEE Software, 20 (2003) 26-32.
- [311] X. Seignard, Model driven tools: the big list! <http://mdwhatever.free.fr/index.php/2010/06/model-driven-tools-the-big-list/> (accessed 2012).
- [312] B. Selic, The Pragmatics of Model-Driven Development, IEEE Software, 20 (2003) 19 - 25
- [313] B. Selic, J. Rumbaugh, Using UML for Modeling Complex Real-Time Systems. http://www.ibm.com/developerworks/rational/library/content/03July/1000/1155/1155_umlmodeling.pdf (accessed septiembre 2012).
- [314] D. Sellier, M. Mannion, J.X. Mansell, Managing requirements inter-dependency for software product line derivation, Requirements Engineering, 13 (2008) 299-313.
- [315] S. Sendall, W. Kozaczynski, Model Transformation - the Heart and Soul of Model-Driven Software Development, IEEE Software, 20 (2003) 42-45.
- [316] M. Shaw, The Coming-of-Age of Software Architecture Research, 23rd International Conference on Software Engineering, IEEE Computer Society, 2001, pp. 656-664a.
- [317] M. Shaw, What makes good research in Software Engineering?, International Journal of Software Tools for Technology, 4 (2002) 1-7.
- [318] M. Shaw, Writing Good Software Engineering Research Papers. Minitutorial, 25th International Conference on Software Engineering, IEEE Computer Society, 2003, pp. 726-736.
- [319] M.S. Shehata, A. Eberlein, H.J. Hoover, Requirements Reuse and Feature Interaction Management, 15th International Conference on Software & Systems Engineering and their Applications (ICSSEA'02), Paris, France, 2002.
- [320] E.J. Sierra, Del Hogar Digital a la Casa Red, Lulu.com, 2009.
- [321] A.R.d. Silva, J. Saraiva, D. Ferreira, R. Silva, C. Videira, Integration of RE and MDE paradigms: the ProjectIT approach and tools, IET Software, 1 (2007) 294-314.
- [322] G. Sindre, D.G. Firesmith, A.L. Opdahl, A Reused-based Approach to Determining Security Requirements, The Ninth International Workshop on Requirements Engineering: Foundation for Software Quality - REFSQ'03, 2003.
- [323] M. mia ek, J. Bojarski, W. Nowakowski, A. Ambroziewicz, T. Straszak, Complementary Use Case Scenario Representations Based on Domain Vocabularies in: Model Driven Engineering Languages and Systems. LNCS 4735, 2007, pp. 544-558.
- [324] J.M. Smith, Troubled IT Projects prevention and turnaround, IEEE, London, 2001.
- [325] M.d.S. Soares, J. Vrancken, Model-Driven User Requirements Specification using SysML, Journal of Software. Academy Publisher, 3 (2008) 57-68.
- [326] I. Sommerville, Software Engineering (7th edition), Addison-Wesley, 2004.
- [327] I. Sommerville, Software Engineering (9th edition), Addison-Wesley, 2011.
- [328] I. Sommerville, J. Ransom, An Empirical Study of Industrial Requirements Engineering Process Assesment and Improvement, ACM Transactions on Software Engineering and Methodology, 14 (2005) 85-117.
- [329] I. Sommerville, P. Sawyer, Requirements Engineering. A good practice guide, Wiley, 1997.
- [330] I. Sommerville, P. Sawyer, Viewpoints: principles, problems and a practical approach to requirements engineering, Annals of Software Engineering, 3 (1997) 101-130.

- [331] I. Sommerville, P. Sawyer, S. Viller, Viewpoints for requirements elicitation: a practical approach, IEEE International Conference on Requirements Engineering (ICRE'98), Colorado Springs, Colorado, 1998.
- [332] Standish-Group, CHAOS Report. 1994.
- [333] Standish-Group, CHAOS Summary 2009. 2009.
- [334] A. Stone, P. Sawyer, Finding Tacit Knowledge by Solving the Pre-Requirements Tracing Problem, 11th International Workshop on Requirements Engineering: Foundation for Software Quality (REFSQ'05), Porto, Portugal, 2005.
- [335] R.V.D. Straeten, T. Mens, S.V. Baelen, Challenges in Model-Driven Software Engineering, in: Models in Software Engineering. LNCS Volume 5421/2009, 2009, pp. 35-47.
- [336] L. Sun, G. Huang, Y. Sun, H. Song, H. Mei, An Approach for Generation of J2EE Access Control Configurations from Requirements Specification, The Eighth International Conference on Quality, IEEE Computer Society, Oxford, UK, 2008.

T

- [337] R. Tavakoli-Kolagari, M.-O. Reiser, Reusing Requirements: The Need for Extended Variability Models, International Symposium on Fundamentals of Software Engineering (FSEN 2007), Tehran, Iran, 2007.
- [338] A. Tikhomirov, A. Shatalin, GMF Best Practices, EclipseCon, 2007.
- [339] A. Toval, B. Moros, J. Nicolás, J. Lasheras, Eight Key Issues for an Effective Reuse-based Requirements Process, Computer Systems Science and Engineering, 23 (2008).
- [340] A. Toval, J. Nicolás, B. Moros, Requisitos Reutilizables de Seguridad en Sistemas de Información y Bases de Datos, in: Seguridad en Bases de Datos, Dintel, 2001, pp. 269-300.
- [341] A. Toval, J. Nicolás, B. Moros, SIREN: Un proceso de Ingeniería de Requisitos basado en reutilización, Jornadas de Ingeniería de Requisitos Aplicada, Sevilla, 2001.
- [342] A. Toval, J. Nicolás, B. Moros, SIREN: Un proceso de Ingeniería de Requisitos basado en reutilización, in: Applying Requirements Engineering, Catedral Publicaciones, Sevilla, 2002, pp. 57-72.
- [343] A. Toval, J. Nicolás, B. Moros, F. García, Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach, Requirements Engineering Journal. Springer, 6 (2002) 205-219.
- [344] A. Toval, A. Olmos, M. Piattini, Legal Requirements Reuse: A Critical Success Factor for Requirements Quality and Personal Data Protection, I.C. Press (Ed.) IEEE Joint International Conference on Requirements Engineering (ICRE'02 and RE'02), Essen, Alemania, 2002, pp. 9-13.

V

- [345] P. Valderas. A Requirements Engineering Approach for the Development of Web Applications. Sistemas Informáticos y Computación. Universidad Politécnica de Valencia. 2007.
- [346] P. Valderas, V. Pelechano, Introducing requirements traceability support in model-driven development of web applications, Information and Software Technology, 51 (2009) 749 - 768.
- [347] C. Vicente-Chicote, F. Losilla, B. Álvarez, A. Iborra, P. Sánchez, Applying MDE to the Development of Flexible and Reusable Wireless Sensor Networks, International Journal of Cooperative Information Systems, 16 (2007) 394-412.
- [348] C. Vicente-Chicote, B. Moros, A. Toval, REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting, Journal of Object Technology (JOT), 6 (2007).

- [349] C. Vicente-Chicote, B. Moros, A. Toval, REMM-Studio: an Integrated Model-Driven Environment for Requirements Specification, Validation and Formatting, TOOLS Europe 2007, Zurich, Switzerland, 2007.
- [350] C. Vicente-Chicote, B. Moros, A. Toval, REMM-Studio: Un entorno integrado para dar soporte a un enfoque de Ingeniería de Requisitos Dirigido por Modelos (tool demo), XII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'07) Zaragoza, Spain, 2007.
- [351] C. Videira, A.R.d. Silva, An overview of ProjectIT-RSL metamodel and prototype, 6ª Conferência da Associação Portuguesa de Sistemas de Informação (CAPSI'2005), Portugal, Bragança, 2005.
- [352] P.D. Villarreal, E. Salomone, O. Chiotti., Modeling and Specifications of Collaborative Business Processes using a MDA Approach and a UML Profile, in: Enterprise Modeling and Computing with UML, Idea Group Inc., 2007, pp. 13-45.
- [353] Visure, IRqA. <<http://www.visuresolutions.com/irqa-requirements-tool>> (accessed 2012).
- [354] M. Voelter, Best Practices for DSLs and Model-Driven Development, Journal of Object Technology, 8 (2009) 79-102.
- [355] R. Vogel, K. Mantell, MDA adoption for a SME: evolution, not revolution - Phase II, European Conference on Model Driven Architecture (ECMDA 2006), Bilbao, Spain, 2006.
- [356] Volere, Requirements Tools. <<http://www.volere.co.uk/tools.htm>> (accessed 2012).

W

- [357] M. Weber, J. Weisbrod, Requirements Engineering in Automotive Development: Experiences and Challenges, IEEE Software, 20 (2003) 16-24.
- [358] J. Whittle, J. Araujo, Scenario Modelling with Aspects, IEE Proceedings - Software Special Issue, 151 (2004) 157-172.
- [359] K.E. Wiegers, Read My Lips: No New Models!. IEEE Software, (1998).
- [360] K.E. Wiegers, Software Requirements, 2nd Edition, Microsoft Press, 2003.
- [361] C. Wohlin, M. Höst, K. Henningsson, Empirical Research Methods in Software Engineering, in: R. Conradi, A.I. Wang (Eds.) Empirical Methods and Studies in Software Engineering. LNCS 2765, Springer Berlin / Heidelberg, 2003, pp. 7-23.
- [362] C. Woody, Eliciting and Analyzing Quality Requirements: Management Influences on Software Quality Requirements. CMU/SEI-2005-TN-010, Software Engineering Institute, Carnegie Mellon University, 2005.

Y

- [363] R.K. Yin, Case Study Research. Design and Methods. 3rd edition, SAGE Publications, 2003.
- [364] E. Yourdon, Análisis estructurado moderno, Mexico:Prentice-Hall, 1993.
- [365] E. Yu. Modelling strategic relationships for process reengineering. Department of Computer Science. University of Toronto. 1995.
- [366] E.S.K. Yu, Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering Third IEEE International Symposium on Requirements Engineering (RE'97), Annapolis, MD, 1997.
- [367] Y. Yu, J.C.S.P. Leite, J. Mylopoulos, From Goals to Aspects: Discovering Aspects from Requirements Goal Models, 12th IEEE International Requirements Engineering Conference, Kyoto, Japan, 2004.
- [368] T. Yue, L.C. Briand, Y. Labiche, A systematic review of transformation approaches between user requirements and analysis models, Requirements Engineering, 16 (2011) 75-99.

Z

- [369] P. Zave, Classification of research efforts in Requirements Engineering, *ACM Computing Surveys*, XXIX (1997) 315-321.
- [370] H. Zhang, M.A. Babar, P. Tell, Identifying relevant studies in software engineering, *Information and Software Technology*, 53 (2011) 625-637.
- [371] L. Zhang, W. Jiang, Transforming Business Requirements into BPEL: a MDA-Based Approach to Web Application Development, *First IEEE International Workshop on Semantic Computing and Systems*, Huangshan, China, 2008.
- [372] W. Zhang, H. Mei, H. Zhao, J. Yang, Transformation from CIM to PIM: a feature-oriented component-based approach, *8th International Conference on Model Driven Engineering Languages and Systems*, Montego Bay, Jamaica, 2005.
- [373] Y. Zhu, Z. Huang, Z. Cao, H. Zhou, M. Yuan, An MDE Based Approach for Generating Software Architecture Models from Formal Specifications, *10th International Conference on Quality Software*, Zhangjiajie, China, 2010.
- [374] I. Zikra, J. Stirna, J. Zdravkovic, Analyzing the Integration between Requirements and Models in Model Driven Development, *BMMDS/EMMSAD 2011*, Springer, London, United Kingdom, 2011.
- [375] I. Zikra, J. Stirna, J. Zdravkovic, Bringing Enterprise Modeling Closer to Model-Driven Development, in: *The Practice of Enterprise Modeling*, Springer Berlin Heidelberg, 2011, pp. 268-282.

ANEXO I.

Trabajos excluidos en la RSL sobre metamodelos de requisitos

El criterio para la *inclusión* de un trabajo relevante en la RSL es que describa un metamodelo que permita representar la especificación textual de requisitos. El artículo debe proporcionar una representación gráfica de dicho metamodelo, utilizando algún formalismo, de manera que queden explícitos los conceptos y relaciones que incluye. En consecuencia, el metamodelo contendrá el concepto “Requisito” (con este o cualquier otro nombre) que se caracterice por una descripción textual.

Por tanto, quedan *excluidos* del estudio:

- CE1) El metamodelo de requisitos aparece en la propuesta como un componente tangencial y no con el objetivo de dar soporte a la especificación de requisitos. Algunos trabajos excluidos por este motivo podrían ser: 1) un artículo en el que se presenta un mecanismo para detectar inconsistencias entre modelos y proponen un metamodelo de requisitos como ejemplo, pero no pretendiendo contribuir al dominio de la gestión de requisitos; 2) trabajos en los que se representan en un metamodelo los elementos que forman parte de un proceso, pero no concretamente los artefactos relacionados con la especificación de requisitos.
- CE2) Propuestas en las que los requisitos textuales no estén representados en un modelo conforme a un metamodelo.
- CE3) Los trabajos que describan textualmente el metamodelo pero no proporcionan una representación gráfica del mismo, o la que se de sea incompleta.
- CE4) Trabajos que solamente traten la especificación de requisitos de un problema muy concreto (por ejemplo, requisitos para un problema de computación científica, o los requisitos para la definición de una transformación de modelos) y no la especificación de una aplicación de desarrollo de software.
- CE5) Trabajos que propongan una extensión a otros modelos generales de IR que dan soporte a la especificación de requisitos. Por ejemplo, una extensión para dar soporte a los requisitos de seguridad definidos en otro metamodelo. En este caso se incluirá en la RSL el metamodelo general.

CE6) Trabajos duplicados del mismo estudio. En el caso de que existan versiones diferentes de un mismo enfoque se toma la más completa o bien la que incluya el metamodelo de requisitos, si en la versión completa no se detalla.

En las siguientes tablas se muestran los trabajos encontrados por los motores de búsqueda que fueron excluidos, así como los motivos que nos llevaron a tomar esta decisión, tomando como base los criterios previamente establecidos.

Fuente	Autores	Título	Año	Motivo de exclusión
ACM	Blanc <i>et al.</i>	<i>Detecting model inconsistency through operation-based model construction</i>	2008	CE1. Proponen un mecanismo para detectar inconsistencias entre modelos.
	Cavalcanti <i>et al.</i>	<i>Towards metamodel support for variability and traceability in software product lines</i>	2011	CE1. Propuesta centrada en la definición de características de un SPL y el modelo de variabilidad.
	Ding <i>et al.</i>	<i>From textual use cases to service component models</i>	2011	CE2. Requisitos textuales (no representados en un modelo) se transforman en casos de uso.
	Goknil <i>et al.</i>	<i>Tool support for generation and validation of traces between requirements and architecture</i>	2010	CE6. El metamodelo se presenta en otro trabajo.
	Guerra <i>et al.</i>	<i>Engineering model transformations with transML</i>	2011	CE4. Modelo de requisitos para las transformaciones de modelos.
	Guerra <i>et al.</i>	<i>transML: a family of languages to model model transformations</i>	2010	CE6. Versión más reciente en 2011
	Koch <i>et al.</i>	<i>Model transformations from requirements to web system design</i>	2006	CE6. El metamodelo se describe en trabajos de Escalona <i>et al.</i> de 2006.
	Lammari <i>et al.</i>	<i>A conceptual meta-model for secured information systems</i>	2011	CE5. El metamodelo proporciona una taxonomía de requisitos de seguridad.
	Lee y Zhao	<i>Domain requirements elicitation and analysis - an ontology-based approach</i>	2006	CE1. El metamodelo se utiliza para guiar la elicitación de requisitos, no para modelar los requisitos.
	Peraldi-Frati y Albinet	<i>Requirement traceability in safety critical systems</i>	2010	CE3. No muestran el metamodelo formalmente.
	Savolainen y Kuusela	<i>Violatility analysis framework for product lines</i>	2001	CE3. El metamodelo es incompleto.
	Streitferdt	<i>Traceability for system families</i>	2001	CE7. Trabajo de 2 páginas.
IEEE	Abrahao e Insfran	<i>A Metamodeling Approach to Estimate Software Size from Requirements Specifications</i>	2008	CE2. Utilizan modelos UML como modelos de requisitos.
	Baundry <i>et al.</i>	<i>Model-Driven Engineering for Requirements Analysis</i>	2007	CE2. Metamodelo para un lenguaje de natural restringido CE6. Versión posterior Brottier <i>et al.</i> 2010 (MP)
	Bragança Machado	<i>Adopting Computational Independent Models for Derivation of Architectural Requirements of Software Product Lines</i>	2007	CE2. La propuesta se basa en la transformación de diagramas de casos de uso en diagramas de componentes.
	Denger <i>et al.</i>	<i>Higher Quality Requirements Specifications through Natural Language Patterns</i>	2003	CE2. Proponen un metamodelo para sentencias de requisitos.
	Escalona Aragón	<i>NDT: A Model-Driven Approach for Web Requirements</i>	2008	CE2. Requisitos para el web

Fuente	Autores	Título	Año	Motivo de exclusión
IEEE	Geisberger <i>et al.</i>	<i>Interdisciplinary Requirements Analysis Using the Model-based RM Tool AUTORAID</i>	2007	CE6. En GS hemos encontrado una versión posterior
	Jingbai <i>et al.</i>	<i>A Context Awareness Non-functional Requirements Metamodel Based on Domain Ontology</i>	2008	CE2. El metamodelo describe la forma en la que se tienen que escribir las sentencias de requisitos.
	Li <i>et al.</i>	<i>Requirements Engineering for Scientific Computing: A Model-Based Approach</i>	2011	CE4. Metamodelo para los requisitos de problemas de computación científica.
	Lu <i>et al.</i>	<i>A Model-based Object-oriented Approach to Requirement Engineering (MORE)</i>	2007	CE7. Tiene 4 páginas. Analizamos una versión posterior (de 2008) como desviación del protocolo.
	Molina <i>et al.</i>	<i>Towards a Requirements-Aware Common Web Engineering Metamodel</i>	2008	CE6. Analizamos una versión posterior de 2010.
	Nissen <i>et al.</i>	<i>Managing Multiple Requirements Perspectives with Metamodels</i>	1996	CE3. No proponen ningún metamodelo.
	Wei <i>et al.</i>	<i>Combining Domain-Driven Approach with Requirement Assets for Networked Software Requirements Elicitation</i>	2008	CE2. El metamodelo representa las sentencias de los requisitos.
SD	Molina y Toval	<i>Integrating usability requirements that can be evaluated in design time into Model Driven Engineering of Web Information Systems</i>	2009	CE6. Analizamos una versión posterior de 2010.
	Rolland <i>et al.</i>	<i>Guiding Lyee user requirements capture</i>	2003	CE4. Propuesta muy particular para una herramienta del entorno comercial Japonés.
	Schneider <i>et al.</i>	<i>A modeling language to support early lifecycle requirements modeling for systems engineering</i>	2012	CE3. Metamodelo presentado informalmente.
	Valderás y Pelechano	<i>Introducing requirements traceability support in model-driven development of web applications</i>	2009	CE2. Modelo de requisitos basados en tareas.
	Xiao y Greer	<i>Adaptive Agent Model: Software Adaptivity using an Agent-oriented Model-Driven Architecture</i>	2009	CE2. Modelado de procesos, objetivos y reglas del negocio.
MP	Brottier <i>et al.</i>	<i>Composing models at two modeling levels to capture heterogeneous concerns in requirements</i>	2010	CE2. Metamodelo para un lenguaje de natural restringido
	Goknil <i>et al.</i>	<i>Semantics of trace relations in requirements models for consistency checking and inferencing</i>	2011	CE6. El metamodelo se presenta en Goknil <i>et al.</i> 2008 (seleccionado en la RSL)
	Hatebur y Heisel	<i>Making Pattern- and Model-Based Software Development More Rigorous</i>	2010	CE4. Centrado en el <i>Problem Frame Approach</i> y no en un enfoque de IR general.
	Laleau <i>et al.</i>	<i>A first attempt to combine SysML requirements diagrams and B</i>	2009	CE5. No presenta ningún metamodelo nuevo. Extiende el modelo de requisitos SysML para dar soporte a los conceptos del modelo de KAOS que se pueden transformar en una especificación formal en B.

Fuente	Autores	Título	Año	Motivo de exclusión
MP	Maiden <i>et al.</i>	<i>Generating requirements from systems models using patterns: a case study</i>	2005	CE2. El metamodelo es para soportar las sentencias de requisitos generadas a partir de modelos i*.
	Maiden <i>et al.</i>	<i>Model-Driven Requirements Engineering: Synchronising Models in an Air Traffic Management Case Study.</i>	2004	CE2. El metamodelo sirve para sincronizar los modelos de sistema i*, descripciones de casos de uso y modelo de actividades.
	Markov <i>et al.</i>	<i>Requirements engineering process improvement: an industrial case study</i>	2011	CE3. El artículo dice que se define un metamodelo de requisitos que no se muestra.
	Molina <i>et al.</i>	<i>A UML Profile for Modelling Measurable Requirements</i>	2008	CE6. Analizamos una versión posterior de 2010.
	Molina <i>et al.</i>	<i>Modelling Web-Based Systems Requirements Using WRM</i>	2008	CE6. Analizamos una versión posterior de 2010.
	Moros <i>et al.</i>	<i>REMM-Studio+: Modeling Variability to Enable Requirements Reuse</i>	2008	CE7. Tiene dos páginas.
	Navarro <i>et al.</i>	<i>MORPHEUS: A Supporting Tool for MDD</i>	2011	CE6. El metamodelo se presenta en Navarro <i>et al.</i> 2006, que introducimos como desviación del protocolo.
	Nicolás <i>et al.</i>	<i>An integrated domain analysis approach for teleoperated systems</i>	2009	CE1. El metamodelo representa los artefactos para la especificación de una línea de productos.
	Penzenstadler <i>et al.</i>	<i>A Requirements Reference Model for Model-Based Requirements Engineering in the Automotive Domain</i>	2009	CE3. No se presenta formalmente un metamodelo de requisitos
	Perrouin <i>et al.</i>	<i>Composing Models for Detecting Inconsistencies: A Requirements Engineering Perspective</i>	2009	CE6. Se puede encontrar una versión más reciente del trabajo en Brottier <i>et al.</i> 2010
	Piprani <i>et al.</i>	<i>An Adaptable ORM Metamodel to Support Traceability of Business Requirements across System Development</i>	2008	CE3. Presentan el esquema de la base de datos ORACLE que han utilizado para almacenar los requisitos de manera que sean trazables.
	Santos <i>et al.</i>	<i>A Knowledge-Based and Model-Driven Requirements Engineering Approach to Conceptual Satellite Design</i>	2009	CE5. No presenta un metamodelo nuevo, sino que utiliza el modelo de requisitos de SysML.
Wiley	Kuruganti y Manci	<i>Building and applying requirements models</i>	2003	CE3. Metamodelo descrito informalmente.
GS	Aguilar <i>et al.</i>	<i>An MDA Approach for Goal-oriented Requirement Analysis in Web Engineering</i>	2010	CE2. Los requisitos se expresan en modelos i*
	Brottier <i>et al.</i>	<i>Producing a Global Requirement Model from Multiple Requirement Specifications</i>	2007	CE6. Se puede encontrar una versión más reciente del trabajo en Brottier <i>et al.</i> 2010
	Escalona y Koch	<i>Metamodeling the Requirements of Web Systems</i>	2006	CE6. Se puede encontrar una versión más completa en Koch 2006 de ACM
	Fernández y Mújica	<i>Model-based development of security requirements</i>	2011	CE3. No se presenta un metamodelo de manera formal.
	Harbouche <i>et al.</i>	<i>A transformation approach for collaboration based requirement models</i>	2012	CE2. Los requisitos se representan mediante diagramas de actividades y colaboraciones UML

Fuente	Autores	Título	Año	Motivo de exclusión
GS	Hurtado <i>et al.</i>	<i>Metamodel of Requirements' Types Classifications for Improving the Software Development Process (SDP): An Ontological Approach</i>	2006	No disponible. Solicitado a los autores, sin respuesta.
	Kabanda <i>et al.</i>	<i>A Requirements Metamodel Framework for Enhancing Product Adoption</i>	2007	CE4. Extienden el metamodelo propuesto por Moon <i>et al.</i> [226] para representar los requisitos sociales.
	Loniewski <i>et al.</i>	<i>An Agile Method for Model-Driven Requirements Engineering</i>	2011	CE2. Los requisitos se representan mediante escenarios y casos de uso
	Luttikhuisen <i>et al.</i>	<i>WP3, Derivation of Products and Evolution of System Families. Task 3.1: Requirements Modelling and Traceability</i>	2001	CE6. El metamodelo de requisitos que adoptan es el propuesto por Pohl [274]
	López <i>et al.</i>	<i>Metamodeling for Requirements Reuse</i>	2002	CE2. El metamodelo no incluye los requisitos textuales.
	Robinson y Volkov	<i>A meta-model for restructuring stakeholder requirements</i>	1997	CE4. Metamodelo para la resolución de conflictos.
	Schätz <i>et al.</i>	<i>Model-Based Requirements Engineering with AutoRAID.</i>	2005	CE6. Los trabajos de Geisberger <i>et al.</i> son posteriores.
	Streitferdt	<i>Traceability for system families</i>	2001	CE7. Sólo dos páginas
	Sánchez <i>et al.</i>	<i>Introducing Safety Requirements Traceability Support in Model-Driven Development of Robotic Applications</i>	2011	CE5. Modelan los conceptos relacionados con <i>safety</i> que puede complementar cualquier metamodelo de requisitos.
	Sánchez <i>et al.</i>	<i>ModelSec: A Generative Architecture for Model-Driven Security.</i>	2009	CE5. Modelan los conceptos relacionados con seguridad que puede complementar cualquier metamodelo de requisitos.
	Moros <i>et al.</i>	<i>Metamodeling Variability to Enable Requirements Reuse</i>	2008	CE6. El origen del trabajo, donde se presenta el metamodelo es [234]
	Videira y da Silva	<i>An overview of ProjectIT-RSL metamodel and prototype</i>	2005	CE2. Presenta un metamodelo para definir un lenguaje de requisitos restringido

*No basta saber, se debe también aplicar.
No es suficiente querer, se debe también hacer.*

(Johann Wolfgang Goethe)