Universidad de Murcia
Facultad de Informática

PROYECTO INFORMÁTICO

COTSRE+: Un método de desarrollo basado en componentes y requisitos

Autor:

María José Casalins Pina
mjcasalins@gmail.com

Tutores:

José Ambrosio Toval
atoval@um.es

Miguel Ángel Martínez Aguilar
mmart@um.es

Departamento de Informática y Sistemas

Fecha: 30 de Junio de 2010

Tabla de Contenidos

TABLA DE CONTENIDOS **1**

ÍNDICE DE FIGURAS **3**

ÍNDICE DE TABLAS **4**

RESUMEN **5**

1. INTRODUCCIÓN Y ORGANIZACIÓN DEL PROYECTO **7**

1.1 INTRODUCCIÓN 7
1.2 ORGANIZACIÓN DEL PROYECTO 8
1.3 AGRADECIMIENTOS 9

2. OBJETIVOS Y METODOLOGÍA **9**

2.1 ANÁLISIS DE LOS OBJETIVOS 9
2.2 METODOLOGÍA Y HERRAMIENTAS 9
2.2.1 Proceso de desarrollo *9*
2.2.2 Herramientas utilizadas *10*

3. DESARROLLO BASADO EN COMPONENTES **11**

3.1 CARACTERÍSTICAS DE LOS COMPONENTES 11
3.2 COMPONENTES Y REQUISITOS 12
3.3 EQUIPO DE DESARROLLO 13
3.4 SELECCIÓN DE COMPONENTES 13
3.4.1 Propuestas para la selección de componentes *13*
3.4.2 Técnicas de selección de componentes *14*
3.4.3 Conclusiones *16*
3.5 SELECCIÓN/CLASIFICACIÓN DE COMPONENTES DE UN REPOSITORIO 17
3.6 COMPONENTES EN UML 18
3.7 PROCESOS DE DESARROLLO BASADO EN COMPONENTES 18
3.7.1 Proceso de Qureshi y Hussai *19*
3.7.2 Proceso de Chesman y Daniels *19*
3.7.3 Comparación con el desarrollo de software tradicional *20*

4. UN PROCESO DE DBC: PROCESO DE CHESSMAN Y DANIELS **22**

4.1 EL PROCESO DE CHESSMAN Y DANIELS 22
4.1.1 Fase de requisitos *26*
4.1.2 Fase de especificación *26*
4.1.3 Fase de aprovisionamiento *32*
4.1.4 Fase de ensamblaje y pruebas *33*
4.2 ESTUDIO DE CÓMO AFECTAN LOS REQUISITOS A CADA UNA DE LAS FASES DEL PROCESO. 33
4.2.1 Fase de requisitos *34*
4.2.2 Fase de especificación *35*
4.2.3 Fase de aprovisionamiento *37*
4.2.4 Fase de ensamblaje y pruebas *38*
4.2.5 Tabla – Resumen *38*
4.3 VALORACIÓN DEL PROCESO DE CHESSMAN Y DANIELS 39

5. COTSRE+: HACIA UN MÉTODO DE DESARROLLO BASADO EN COMPONENTES Y REQUISITOS **40**

5.1 LIMITACIONES DE COTSRE 42
5.2 SPEM 42
5.3 TÉCNICA DE CLASIFICACIÓN Y RECUPERACIÓN 44
5.4 ESPECIFICACIÓN DEL PROCESO COTSRE+ CON SPEM 45
5.4.1 Fase de requisitos *47*
5.4.2 Fase de especificación de componentes *51*
5.4.3 Fase de aprovisionamiento y desarrollo *59*
5.4.4 Fase de ensamblamiento y despliegue *62*

5.5 APORTACIONES DE COTSRE+ SOBRE EL PROCESO DE CHESSMAN Y DANIELS 64

6. EDICIÓN DE COTSRE+ CON EPF COMPOSER 66

6.1 CONTENIDO DE MÉTODO (METHOD CONTENT) 67
6.1.1 Categorías 67
6.1.2 Roles 68
6.1.3 Productos de trabajo 69
6.1.4 Tareas 70
6.1.5 Guías 71
6.2 PROCESOS 72
6.2.1 Ciclo de vida 73
6.2.2 Diagramas 75
6.3 CONFIGURACIÓN DE MÉTODO 76
6.4 PUBLICACIÓN 78

7. PUBLICACIÓN WEB DE COTSRE+ 79

7.1 INTRODUCCIÓN 79
7.2 CICLO DE VIDA DE COTSRE+ 79
7.3 ROLES 81
7.4 PRODUCTOS DE TRABAJO 82
7.5 TAREAS 83
7.6 GUÍAS 85
7.7 ACERCA DE 85

8. COTSREAPP: SELECCIÓN DE COMPONENTES DE COTSRE+ 87

8.1 MANUAL DE USUARIO 87
8.1.1 Casos de Uso 87
8.1.2 Requisitos 90
8.1.3 Características 92
8.1.4 Componentes 93
8.1.5 COTSRE+ 96
8.2 EJEMPLOS DE SELECCIÓN DE COMPONENTES 98
8.2.1 Ejemplo 1 98
8.2.2 Ejemplo 2 100
8.2.3 Ejemplo 3 102
8.2.4 Ejemplo 4 103
8.3 CALIDAD DEL MÉTODO 105

9. CONCLUSIONES Y VÍAS FUTURAS 106

9.1 CONCLUSIONES 106
9.2 VÍAS FUTURAS 106

BIBLIOGRAFÍA Y REFERENCIAS 108

ANEXO I: PLANTILLA DE CASOS DE USO 111

ANEXO II: MODELO CONCEPTUAL PARA COTSRE APP 112

ANEXO III: TÉRMINOS DE SPEM 2.0 Y EPF COMPOSER 113

Índice de Figuras

Figura 1. Plantillas de SIREN 8
Figura 2. Proceso de Qureshi y Hussain 19
Figura 3. Proceso de Chessman y Daniels 20
Figura 4. Evolución 22
Figura 5. Interfaces componentes 23
Figura 6. Visiones del componente 23
Figura 7. Relaciones estructurales de los componentes 24
Figura 8. Tipos de contrato 25
Figura 9. Fase de identificación de componentes 27
Figura 10. Fase de interacción de componentes 29
Figura 11. Fase de especificación de componentes 30
Figura 12. Elementos básicos de SPEM 43
Figura 13. Estructura de paquetes de SPEM 43
Figura 14. Estructura de Method Framework 44
Figura 15. Diagrama de fases SPEM 46
Figura 16. Diagrama actividad de la fase de requisitos 48
Figura 17. Diagrama de actividad de fase de identificación de componentes 52
Figura 18. Diagrama de actividad fase de interacción de componentes 55
Figura 19. Diagrama de actividad de la fase de especificación de componentes 57
Figura 20. Diagrama de actividad de la fase de aprovisionamiento 60
Figura 21. Diagrama de actividades de la fase de despliegue y pruebas 63
Figura 22. Entorno de Eclipse Process Framework Composer (EPF) 66
Figura 23. Estructura de la librería COTSRE 67
Figura 24. Estructura de las categorías de COTSRE 68
Figura 25. Pestaña creación de un rol 69
Figura 26. Pestaña creación de un producto de trabajo 70
Figura 27. Pestaña creación de una tarea 71
Figura 28. Pestaña creación de una guía 72
Figura 29. Pestaña creación de un proceso 73
Figura 30. Pestaña Ciclo de Vida COTSRE+. Fases, Iteraciones y Objetivos 74
Figura 31. Pestaña Ciclo de Vida COTSRE+. Actividades y Fases 74
Figura 32. Editor de Diagramas de Actividad 75
Figura 33. Editor Diagrama de Detalle de Actividad 76
Figura 34. Pestaña configuración 77
Figura 35. Configuración de las vistas 77
Figura 36. Página de inicio 79
Figura 37. Vista del Ciclo de Vida COTSRE+ 79
Figura 38. Vista Iteración. Pestaña Work Breakdown Structure 1 80
Figura 39. Vista Iteración. Pestaña Work Breakdown Structure 2 80
Figura 40. Vista Fase. Pestaña Work Breakdown Structure 1 80
Figura 41. Vista Fase. Pestaña Work Breakdown Structure 2 81
Figura 42. Vista Objetivo 81
Figura 43. Vista conjunto de roles 81
Figura 44. Vista listado de roles 82
Figura 45. Vista rol 82
Figura 46. Vista dominio de productos de trabajo 82
Figura 47. Vista listado de productos de trabajo de un dominio 83
Figura 48. Vista Producto de Trabajo 83
Figura 49. Vista Disciplinas 83
Figura 50. Vista tareas de una disciplina 84
Figura 51. Vista Tarea 1 84
Figura 52. Vista Tarea 2 84
Figura 53. Vista Listado Plantillas 85
Figura 54. Vista Plantilla 85
Figura 55. Vista Acerca De 86
Figura 56. Menú CotsreApp 87
Figura 57. Submenú de Casos de Uso 87
Figura 58. Nuevo Actor 88

Figura 59. Mensajes Nuevo Actor *88*
Figura 60. Nuevo Caso de Uso *88*
Figura 61. Mensaje Caso de Uso *89*
Figura 62. Buscar Caso de Uso *89*
Figura 63. Mensaje Buscar Caso de Uso *89*
Figura 64. Ver caso de uso *90*
Figura 65. Mensaje de borrado de caso de uso *90*
Figura 66. Submenú Requisito *90*
Figura 67. Nuevo Requisito *91*
Figura 68. Requisito guardado *91*
Figura 69. Buscar Requisito *91*
Figura 70. Mensaje de requisito no seleccionado *92*
Figura 71. Ver Requisito *92*
Figura 72. Mensaje informativo requisito guardado *92*
Figura 73. Submenú Característica *92*
Figura 74. Nueva Característica 1 ..... *93*
Figura 75. Nueva Característica 2 ..... *93*
Figura 76. Submenú Componente *93*
Figura 77. Nuevo Componente. Pestaña Componente *94*
Figura 78. Mensaje informativo componente guardado *94*
Figura 79. Nuevo Componente. Pestaña Características *94*
Figura 80. Nuevo Componente. Pestaña Requisitos *95*
Figura 81. Buscar Componentes *95*
Figura 82. Mensaje no selección *95*
Figura 83. Ver Componente *96*
Figura 84. Submenú Proyecto *96*
Figura 85. COTSRE+ Vacío *97*
Figura 86. Mensaje no selección de requisito y/o caso de uso *97*
Figura 87. Buscar Proyecto *98*
Figura 88. Ver Proyecto *98*
Figura 89. Ejemplo 1. Matriz de Selección *99*
Figura 90. Mensaje varios candidatos *99*
Figura 91. Ejemplo 1. Matriz de Decisión *99*
Figura 92. Ejemplo 1. Componente Seleccionado *100*
Figura 93. Guardar Proyecto *100*
Figura 94. Ejemplo 2. Matriz de Selección *100*
Figura 95. Ejemplo 2. Matriz de Decisión con empate de componentes *101*
Figura 96. Mensaje empate en Matriz de Decisión *101*
Figura 97. Ejemplo 2. Matriz de Decisión *101*
Figura 98. Ejemplo 2. Componente Seleccionado *102*
Figura 99. Ejemplo 3. Matriz de Selección *102*
Figura 100. Mensaje componente encontrado *103*
Figura 101. Ejemplo 3. Componente Seleccionado *103*
Figura 102. Ejemplo 4. Matriz de Selección *103*
Figura 103. Mensaje no componentes. *104*
Figura 104. Ejemplo 4. Matriz de Decisión *104*
Figura 105. Ejemplo 4. Componente Seleccionado *105*

Índice de Tablas

Tabla 1. Resumen técnicas de selección de componentes *16*
Tabla 2. Matriz de Selección *41*
Tabla 3. Características de los componentes *41*
Tabla 4. Matriz de Decisión *41*

Resumen

Este proyecto tiene como objetivo la generación de un modelo de desarrollo basado en componentes y requisitos que tenga en cuenta la reutilización. A partir de un método inicial llamado COTSRE utilizado para seleccionar componentes a partir de una serie de requisitos y características se ha diseñado un proceso que abarca todo el ciclo de vida de un componente, al que llamaremos COTSRE+. Para ello se ha estudiado un proceso de desarrollo de componentes existente creado por Chessman y Daniels y hemos ampliado COTSRE a partir de él para obtener COTSRE+. COTSRE+ se ha formalizado utilizando la notación estándar de OMG llamada SPEM y se ha ampliado la selección de componentes definida en COTSRE para que además de los requisitos y características, puedan utilizarse los casos de uso para seleccionar componentes. Para automatizar este proceso de selección y catalogar los requisitos, casos de uso y componentes se ha desarrollado una herramienta de escritorio llamada CotsreApp. Para la publicación del método se ha desarrollado una web con ayuda de la herramienta EPF Composer de Eclipse.

RECONOCIMIENTO DE COAUTORÍA

Este PFC, realizado por M. J. Casalins Pina se basa en los proyectos de investigación DÉDALO TIN 2006-15175-C05-03 y PANGEA TIN2009-13718-C02-02, ambos subvencionados por el Ministerio de Ciencia e Innovación (MICINN), y dirigidos por el Dr. José Ambrosio Toval Álvarez, así como en otros resultados de investigaciones dirigidas por el mismo junto con el segundo director de este proyecto. Los resultados concretos de este PFC se han obtenido en régimen de coautoría entre los tutores del mismo y el alumno/a que los implementa y así es aceptado de mutuo acuerdo por todos ellos.

1. Introducción y organización del proyecto

1.1 Introducción

Como es bien conocido, la reutilización tiene un papel muy importante en la Ingeniería del Software. La reutilización puede verse como una estrategia que produce o ayuda a producir un sistema mediante el nuevo uso de algún elemento procedente de un esfuerzo de desarrollo anterior. Los beneficios de la reutilización de componentes y demás partes del software como los requisitos, diseño… han atraído la atención de un número considerable de empresas encargadas de desarrollar sistemas software. Por este motivo, ha surgido recientemente un gran interés por crear y utilizar modelos de procesos para desarrollo de software que contemplen la reutilización.

Aunque el concepto de reutilización no es nuevo, recientemente ha emergido, como elemento central de este, el concepto de componente software reutilizable. Las características de los componentes de software reutilizables han obligado a crear nuevos modelos de desarrollo de software basados en la reutilización.

Estos procesos buscan la solución a un problema. Esta solución no es única, es por esto que los métodos deben ir evolucionando y adaptándose a las nuevas necesidades de los diferentes tipos de usuarios y los nuevos conceptos de la Ingeniería de Software.

Otra de las deficiencias encontradas en los métodos basados en la reutilización es que no especifican de forma detallada el ciclo de vida de un componente de software reutilizable. Estos métodos se centran en la reutilización del componente y no en su desarrollo individual.

Dentro del Departamento de Informática y Sistemas se ha definido un método de Ingeniería de Requisitos llamado SIREN [5] [6] [7], *Simple Reuse of Software Requirements*, el cual prescribe un proceso general para la reutilización de requisitos basado en la Ingeniería de Requisitos y en estándares de la Ingeniería del Software. Se basa en la utilización de un repositorio de requisitos que está estructurado por dominios (o catálogo vertical, de un dominio específico de aplicación) y perfiles (o catálogo horizontal, aplicado a diferentes dominios de aplicación). Si se parte de una colección de requisitos que ya fueron especificados para otros proyectos informáticos se reducirá el tiempo de recoger los requisitos de la aplicación actual.

De momento hay dos perfiles:

- Seguridad compatible con MAGERIT (Metodología de Análisis y Gestión de Riesgos del Ministerio de Administraciones Públicas).
- Protección de datos compatible con la Ley Orgánica de Protección de Datos de Carácter Personal (LOPD) y el Reglamento de Medidas de Seguridad (RMS).

Y un dominio desarrollado para SIREN:

- Catálogo de Historias Clínicas (en fase inicial).

SIREN está compuesto por una jerarquía de 5 documentos de especificación de requisitos, no todos obligatorios, que se relacionan según vemos en la siguiente figura. Todas estas plantillas están basadas en los estándares de IEEE.

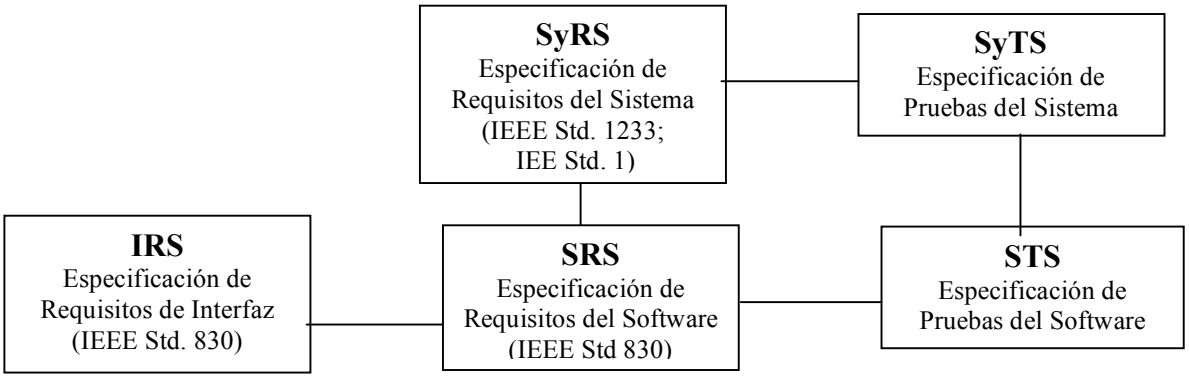


Figura 1. Plantillas de SIREN

Además, SIREN se complementa con una herramienta CARE de gestión de requisitos llamada SIRENTOOL [8].

Pero SIREN sólo abarca el dominio de la reutilización de requisitos por lo que se definió de manera inicial un método de selección de componentes a partir de requisitos llamado COTSRE [2, 3, 4]. En este proyecto se aborda la ampliación de COTSRE para crear un método de desarrollo basado en componentes y requisitos siendo la selección de componentes para su reutilización una parte importante de él, además de ampliar el proceso de selección teniendo en cuenta también los casos de uso. Para elaborar este método se ha realizado un estudio del proceso definido por John Chessman y John Daniels [1] para basarnos en él a la hora de ampliar y definir el nuevo método que llamaremos COTSRE+. Finalmente, COTSRE+ se complementa con una herramienta software que automatiza el proceso de selección de componentes además de incluir un repositorio de requisitos, componentes y casos de uso.

1.2 Organización del proyecto

En esta sección se detalla la organización del proyecto describiendo cada uno de los nueve capítulos.

En el Capítulo 1 se hace una introducción al proyecto situándolo en su contexto y exponiendo su motivación, además de describir la estructura del mismo brevemente.

En el Capítulo 2 se describe en detalle los objetivos perseguidos en este proyecto, los objetivos alcanzados y la metodología utilizada para la realización del mismo.

En el Capítulo 3 se hace una introducción al desarrollo basado en componentes y un estudio de las principales técnicas existentes en selección de componentes y procesos de desarrollo basado en componentes.

En el Capítulo 4 se estudia en detalle un proceso concreto de desarrollo basado en componentes definido por Chessman y Daniels, que es el elegido como base para crear COTSRE+.

En el Capítulo 5 se presenta, define y amplía COTSRE+ a partir del proceso de Chessman y Daniels, utilizando la especificación SPEM.

En el Capítulo 6 se explica como se ha realizado la edición de COTSRE+ utilizando la herramienta recomendada por OMG llamada EPF Composer.

En el Capítulo 7 se presenta la publicación web de COTSRE+ llevada a cabo con la herramienta EPF Composer.

En el Capítulo 8 se presenta la aplicación de escritorio, CotsreApp, para automatizar la selección de componentes a partir de requisitos, casos de uso y características y se acompaña de una serie de ejemplos de uso.

Por último, en el Capítulo 9 expondremos las conclusiones y posibles vías de trabajo futuro.

1.3 Agradecimientos

Me gustaría agradecer a mis padres y mi hermano todo su apoyo y paciencia durante este tiempo. A mis amigos por los ánimos que me han dado siempre. Y a mis tutores de este proyecto Ambrosio y Miguel Ángel por la gran ayuda prestada. Gracias a todos.

2. Objetivos y metodología

2.1 Análisis de los objetivos

El objetivo principal del proyecto es la propuesta de un método de desarrollo basado en componentes y requisitos a partir de una definición inicial existente de un método de selección de componentes, denominado COTSRE.

Objetivo 1. Seleccionar y estudiar un método de desarrollo basado en componentes que esté bien estructurado y sea sencillo, eligiéndose para ello el propuesto por Chessman y Daniels.

Objetivo 2. Ampliar COTSRE basándose en este método y especificarlo utilizando SPEM, para obtener un método más completo y mejor definido, COTSRE+.

Objetivo 3. COTSRE inicialmente definía un método de selección de componentes a partir de requisitos textuales, así que otro objetivo ha sido la ampliación del método de selección teniendo en cuenta también los casos de uso.

Objetivo 4. Complementar la selección de componentes de COTSRE+ con la realización de una herramienta Java, llamada CotsreApp, que automatiza este proceso de selección, además de servir de repositorio de requisitos, casos de uso y componentes.

2.2 Metodología y herramientas

2.2.1 Proceso de desarrollo

El método de trabajo que se ha seguido consta de las siguientes actividades (algunas de ellas han sido realizadas en paralelo):

1. Lectura de documentación relativa al desarrollo basado en componentes para introducirme en la materia.
2. Estudio de métodos generales de selección y clasificación de componentes.
3. Estudio de modelos concretos de desarrollo basado componentes: modelo de Qureshi y Hussain, Chessman y Daniels comparando contra el desarrollo de software de manera tradicional.
4. Estudio en detalle del proceso Chessman y Daniels.
5. Lectura de documentación de SPEM y EPF Composer.
6. Ampliación de COTSRE y definición de COTSRE+ basado en Chessman y Daniels utilizando SPEM para realizar una especificación completa, ayudándome de la herramienta EPF Composer para su edición y publicación web del mismo.
7. Desarrollo de una herramienta de escritorio en Java para la selección automática de componentes, llamada CotsreApp. Diseñar y documentar ejemplos de uso.

8. Redactar la memoria del proyecto.

2.2.2 Herramientas utilizadas

En este apartado se detallan las herramientas utilizadas para la realización de este proyecto.

Principalmente para la redacción del proyecto se ha utilizado el procesador de texto de Microsoft Word 2004 [34] para el sistema operativo MAC OS [35] y para visualizar documentos PDF, Vista Previa también de MAC.

Para la especificación de COTSRE+ se ha utilizado SPEM [28] y un editor llamado EPF Composer [31] el cual no tiene versión para MAC por lo que se ha utilizado su versión para Windows. Este editor permite publicar el método en una web, cuyas pruebas de visualización se han hecho utilizando el navegador Mozilla Firefox [36].

La aplicación de escritorio de selección de componentes de COTSRE+ se ha programado en el lenguaje Java y se ha hecho uso de una base de datos sencilla llamada H2 [32] para guardar los datos necesarios. Para su desarrollo se ha utilizado el entorno de Eclipse [30] para MAC.

3. Desarrollo Basado en Componentes

3.1 Características de los componentes

El desarrollo basado en componentes surgió a finales de los 90 como un enfoque basado en la reutilización para el desarrollo de sistemas software.

Un componente software es una pieza identificable de software, reutilizable, reemplazable, documentada y cuya implementación está oculta y solo provee servicios a través de una o varias de sus interfaces, además, puede unirse a otros componentes y formar nuevos componentes [9].

Las principales características de los componentes son [9, 10, 11]:

- Identificable: Los componentes deben tener una identidad clara e identificable.
- Rastreable: Los componentes deben mantener su identidad y deben poder ser encontrados con facilidad ya estén dentro de otro componente o de una aplicación, permitiendo de esta manera su reemplazo o reutilización.
- Reemplazable: Puede ser reemplazado por una nueva versión u otro componente que ofrezca los mismo servicios o funciones sin impactar en la aplicación.
- Accedido sólo a través de sus interfaces: Las aplicaciones sólo heredan el comportamiento ofrecido a través de sus interfaces específicas, y no heredan la implementación como un todo. La interfaz de los componentes se expresa en términos de operaciones parametrizadas.
- Documentación: Para permitir la reutilización, los servicios ofrecidos a través de la interfaz deben ser documentados de manera precisa de tal manera que sea posible entender no sólo como utilizarla sino cuales son los servicios que provee.
- Los servicios ofrecidos no deben cambiar: Después de que los servicios son ofrecidos estos son inmutables. La implementación física de un componente puede cambiar, los servicios ofrecidos a través de sus interfaces no.
- Independencia del lenguaje: Los componentes pueden ser reutilizados independientemente del lenguaje de programación. Esta es una gran oportunidad para la reutilización y consumo de componentes si permiten ser reutilizados por lenguajes y/o herramientas en las cuales no han sido implementados. De hecho su código fuente no tiene por qué estar disponible.
- Reutilizados dinámicamente: Los componentes pueden ser reutilizados dinámicamente dentro de las aplicaciones.
- Ofrecen un servicio genérico: Al ofrecer un servicio que sea muy genérico que pueda ser utilizado de muchas maneras, incrementa las posibilidades de reutilización pero también aumenta la posibilidad de que necesiten ser especializados antes de ser utilizados.

La principal dificultad que se presenta en el desarrollo basado en componentes es el problema de mantenimiento y evolución [12]. Lo normal es que el código fuente de los componentes no esté disponible, y si lo está este puede ser muy complejo, y puesto que los requisitos de la aplicación cambian es casi imposible cambiar los componentes para que reflejen estos requisitos. A veces, los proveedores de los componentes ofrecen actualizaciones y soporte para modificaciones. La opción de cambiar los requisitos para ajustarse a los componentes disponibles por lo general no es posible puesto que la aplicación está en funcionamiento. Entonces se ha de llevar a cabo un trabajo adicional para utilizar los componentes y con el tiempo esto supone unos incrementos en el coste de mantenimiento.

Los COTS, *Comercial off-the-shelf,* son componentes ofrecidos por un fabricante externo. Los beneficios de utilizar productos COTS son significativos puesto que estos sistemas ofrecen mucha más funcionalidad a la persona que los utiliza. Se pueden ahorrar meses y años de esfuerzo en implementación si se reutiliza un sistema existente por lo que los tiempos de desarrollo se reducen bastante.

Existen varias características de los componentes que llevan a la reutilización:

- El componente debe reflejar abstracciones estables del dominio. Éstas son conceptos fundamentales en el dominio de la aplicación que cambian lentamente. Por ejemplo, en un sistema para un colegio pueden ser alumnos, expedientes, profesorado...
- Los componentes deben ocultar la forma en que se representa su estado y deben proveer operaciones que permitan tener acceso y actualizar el estado. Siguiendo con el anterior ejemplo podrían ser dar de alta un alumno, modificar datos del profesor...
- El componente debe ser independiente en todo lo posible, un componente no debe necesitar de otros para operar, pero en la práctica esto sólo es posible para componentes muy pequeños, los grandes suelen depender de otros.
- Un componente no debe tratar una excepción sino que ésta debe de ser parte de su interfaz. No deben tratar excepciones por sí mismos puesto que diversas aplicaciones tendrán diferentes técnicas de manejo de excepciones, de modo que el componente debe especificar qué excepciones pueden producirse al utilizarlo y publicar éstas como parte de la interfaz.

3.2 Componentes y Requisitos

En este trabajo, lo que tratamos de hacer es conseguir una correspondencia entre los requisitos y los componentes software elevando de esta manera el nivel de abstracción en el desarrollo de un sistema software.

Cuando se desarrolla un sistema, los principales obstáculos nos los encontramos en la especificación y la gestión de los requisitos del cliente y en el entorno de funcionamiento.

En la Ingeniería del Software Basada en Componentes la reutilización es una parte importante ya que se puede reutilizar componentes que satisfagan una serie de requisitos. Uno de los procesos críticos en la Ingeniería del Software es la selección de componentes. En DBC (Desarrollo Basado en Componentes) los requisitos nos dan criterios para evaluar y seleccionar un conjunto de componentes candidatos con criterios de aceptación para que sea el usuario el que decida qué componente le ofrece mejores beneficios al reutilizarlo para su sistema. Para la selección de los componentes debemos tener en cuenta estas características:

- Cobertura de dominio: Los componentes deben poder satisfacer todo o una parte de la funcionalidad requerida por el cliente.
- Restricciones de tiempo: No debe consumirse mucho tiempo en la selección de componentes.
- Valoración de costes: Debe considerarse el coste de seleccionar un componente, por ejemplo si debe pagarse la licencia, coste de soporte y actualización...
- Garantía del vendedor: Si es ofrecido por una gran empresa, si es utilizado por muchas aplicaciones, si ofrece buenas actualizaciones...

3.3 Equipo de desarrollo

Se debe constituir un equipo multidisciplinar que combine roles técnicos y no técnicos para el proceso de desarrollo y selección de componentes [13].

Los roles técnicos:

- *Ingenieros de requisitos*: Obtener, analizar y definir los diferentes objetivos y requisitos de los *stakeholders*.
- *Ojeadores del mercado*: Debe clasificar los tipos de productos disponibles en el mercado, identificando a su vez los cambios sustanciales que pueden impactar o influenciar sobre el sistema de información.
- *Component screener*: Capaz de buscar componentes candidatos que se adapten a los requisitos que necesiten de un análisis más detallado.
- *Component evaluator*: Posee un alto perfil técnico capaz de aplicar técnicas y procesos que permitan evaluar detalladamente los productos candidatos.
- *Component customizer*: Que provenga del lado del proveedor, capaz de personalizar el componente cada vez que sea necesario.

Los roles no técnicos:

- *System client*: quien será encargado de determinar y validar los requisitos.
- COTS suplier: Provee información detallada de versiones parciales de los COTS durante el análisis.
- *Manager*: El cual comparte y administra responsabilidades dentro del equipo técnico de la organización.
- *Lawyer*: Provee asistencia en el momento de redactar los contratos y estudiar la licencia y mantenimiento de los componentes.

3.4 Selección de componentes

Si estamos interesados en crear componentes reutilizables también es importante tener claro cómo se va a realizar la selección de los mismos a partir de unos parámetros de búsqueda. Un objetivo de este proyecto es crear un buen método de selección, para ello identificamos y comentamos algunas de las principales propuestas y técnicas de selección de componentes.

3.4.1 Propuestas para la selección de componentes

Según Mary Gorman [25] el primer paso es identificar la necesidad que motiva la expedición comercial. Antes de decidir qué software comprar es necesario definir los objetivos y éxitos de negocio y obtener y analizar el estado actual de la organización. Además se puede recoger y filtrar esta información utilizando una variedad de modelos de negocio.

Antes de comenzar la búsqueda de los componentes COTS que son necesarios para nuestro sistema sería ideal hacernos una serie de preguntas sobre que es lo que realmente necesitamos.

- *¿Qué?* Se deben descubrir las funciones clave y crear un mapa de relaciones con los flujos de información entre funciones.
- *¿Quién entra en contacto con qué funciones?* Se añadirán los clientes y proveedores externos al mapa de relación. Estos serán los actores de las funciones, que podrán ser tanto asesores como proveedores.

- *¿Qué características se incluirán?¿Cuáles son las definiciones de los términos?* Describir los términos clave (creación de un glosario) en un modelo conceptual para mostrar sus relaciones de alto nivel.
- *¿Cómo?* Creación de un mapa de procesos que muestre la secuencia de los procesos a través de las funciones. Se crea un manual con los pasos necesarios en cada proceso.
- *¿Cuándo?* Conocer cuándo se producen los eventos que desencadenan la ejecución de proceso. Es interesante definir un estado para cada proceso como "iniciado", "en proceso", "enviando"… Se necesita saber qué eventos debe responder el COTS.

Por lo que vemos, los requisitos son la base para encontrar el componente COTS más adecuado, tanto los de usuario (*Stakeholders*, actores, glosario, diagrama de contexto, modelo de datos, reglas de negocio, eventos de activación, y escenarios) cómo los no funcionales (atributos de calidad, diseño e implementación e interfaz). A estos últimos requisitos se les debe dar una prioridad para guiar la selección.

Además, estos requisitos no nos valdrán solo para la selección sino también para la evaluación de los COTS.

3.4.2 Técnicas de selección de componentes

A continuación se explican las principales técnicas de selección de componentes:

Off-The-Shelf Option (OTSO) [17, 18]

Esta técnica soporta la búsqueda, evaluación y selección de componentes software reutilizables, y proporciona técnicas para definir los criterios de evaluación, comparando los costes y beneficios de las diferentes alternativas y ayuda en la toma de decisiones.

Los principales principios en los que se basa este método son:

- Definición explicita de las tareas en el proceso de selección y de los criterios de entrada y de salida.
- Definición de los criterios de evaluación de forma incremental, jerárquica y detallada.
- Un modelo para comparar el coste y valor asociado a cada alternativa.
- Ayuda en la toma de decisiones para analizar y resumir los resultados de la evaluación.

Sus fases principales son:

1. Búsqueda: Busca e identifica los componentes candidatos para la reutilización según un conjunto de criterios jerárquicos formados por requisitos (requisitos funcionales, características de calidad, aspectos de negocio y relevantes de la arquitectura).
2. Filtrado: Selecciona un conjunto más especifico de los componentes encontrados.
3. Evaluación: Evalúa las alternativas seleccionadas por los criterios de evaluación.
4. Análisis de resultados: Es el proceso de toma de decisiones.
5. Despliegue: Uso de la alternativa seleccionada para el desarrollo del producto software.
6. Valoración: Evalúa la decisión tomada para aprender para futuras selecciones de componentes.

Procurement-Oriented Requirements Engineering (PORE) [19]

Su ciclo de vida del modelo de proceso tiene seis procesos genéricos que a su vez están definidos a tres niveles: nivel universal, nivel mundial y nivel atómico.

- *Nivel Universal*: Los procesos que describen el nivel de orientación general para los actores en el proceso. Cada uno describe una secuencia uniforme de los procesos.
- *Nivel Mundial*: Los procesos que son relevantes para un proceso iterativo de los requisitos de adquisición y producto de evaluación y selección. Cada uno guía la secuencia de tareas durante las adquisiciones de productos.
- *Nivel Atómico*: Los procesos que son específicos para cada método, procedimientos, técnicas y herramientas que están disponibles en los procesos de nivel mundial.

Utiliza un proceso iterativo de captura de requisitos y selección de componentes. Los componentes que no satisfacen los principales requisitos dados por el cliente son rechazados y eliminados de la lista de candidatos para sucesivas iteraciones. A su vez los requisitos se irán refinando y serán más detallados.

PORE rechaza un componente de acuerdo con la conformidad o no con los siguientes objetivos de calidad:

- Requisitos del cliente atómicos esenciales.
- Requisitos del cliente atómicos no esenciales.
- Requisitos del cliente no atómicos complejos.
- Requisitos del usuario del cliente.

COTS-based Requirements Engineering (CRE) [20]

Al igual que el anterior se crea un proceso iterativo de selección de componentes y refinamiento de requisitos. Cada fase de este método está orientada por objetivos predefinidos. Cada fase tiene una plantilla en la que existen guías y técnicas para la adquisición y modelado de requisitos, para la evaluación de productos.

El método se compone de las siguientes fases:

1. Identificación: Se definen los objetivos basado en ciertos factores de influencia (requisitos de usuario, arquitectura de la aplicación, restricciones y objetivos de proyecto, disponibilidad de productos, infraestructura de la organización).
2. Descripción.
3. Evaluación.
4. Aceptación.

Selecting Components Against Requirements (SCARLET) [21]

Propone un proceso con una herramienta automatizada que guíe el proceso de selección de requisitos de componentes software.

Se establecen cuatro objetivos esenciales para la selección de componentes en base al incumplimiento de diferentes tipo de requisitos del cliente:

- Requisitos simples del cliente.
- Requisitos simples del cliente que necesitan acceder al componente software que a su vez necesita acceder a él a través de pruebas o uso.
- Requisitos simples del cliente que son demostrables por medio del uso de pruebas a corto plazo.
- Requisitos de cliente complejos con dependencias a otros requisitos existentes.

Para conseguir estos objetivos SCARLET sigue los siguientes procesos:

1. Obtener información sobre requisitos del cliente, componentes software y contratos del proveedor.
2. Analizar la información obtenida para asegurar que ésta es completa y correcta.
3. Usar esta información para tomar decisiones sobre el cumplimiento de requisitos-componentes.
4. Rechaza uno o más componentes candidatos como incumplimiento con los requisitos del cliente.

COTS-Aware Requirements Engineering and Software Architecting (CARE/SA) [22]

Este método soporta de forma iterativa el emparejamiento, clasificación y selección de componentes usando una representación de componentes como un conjunto de requisitos (funcionales y no funcionales) y su arquitectura.

En este modelo se identifican 5 actividades:

1. Definición de objetivos.
2. Emparejamiento de componentes.
3. Clasificación de componentes.
4. Selección de componentes.
5. Negociación de cambios.

3.4.3 Conclusiones

Se ha elaborado un cuadro-resumen de las técnicas de selección presentadas en el anterior apartado. [2]

	OTSO	**PORE**	**CRE**	**SCARLET**	**CARE/SA**
Año de publicación	1995	1999	2001	2002	2004
Adquisición de requisitos	NO	SI	PARCIAL	SI	SI
Describe requisitos no funcionales	PARCIAL	PARCIAL	SI	SI	SI
Evaluación de componentes	SI	SI	SI	SI	SI
Análisis de toma de decisiones	SI	SI	SI	SI	SI
Basado en rechazo	NO	NO	NO	SI	NO
Selección múltiple	NO	NO	NO	SI	NO
Identificación de componentes	SI	SI	SI	SI	SI
Facilidad de uso	SI	PARCIAL	PARCIAL	NO	NO
Herramienta CARE	NO	NO	NO	SI	NO
Aceptación del componentes	SI	SI	SI	SI	PARCIAL
Téc. Análisis de Características	NO	SI	NO	SI	NO
Define métricas a utilizar	NO	NO	NO	NO	NO
Negociación de requisitos	NO	NO	NO	NO	PARCIAL

Tabla 1. Resumen técnicas de selección de componentes

De todas las técnicas parece que la más completa es SCARLET pero características tan importantes como la facilidad de uso y una actividad de negociación de requisitos no las cumple.

3.5 Selección/Clasificación de componentes de un repositorio

Hemos hablado en el anterior apartado de distintas técnicas y propuestas para seleccionar componentes. En este apartado hablaremos de distintas formas de clasificar componentes en un repositorio de manera que después su selección, independientemente de la técnica utilizada, sea lo más rápida y completa posible.

El proceso de clasificación consiste en catalogar un componente en un repositorio mediante una estructura de representación interna para que este pueda ser recuperado en un futuro y ser reutilizado. La recuperación de componentes consiste en la búsqueda y selección de componentes presentes en el repositorio que cumplen con una serie de características o requisitos dados por el usuario. En los mecanismos de recuperación también se pueden incluir métricas para asegurar la facilidad con que un componente puede ser reutilizado o adaptado, como frecuencia de reutilización y/o adaptación, calidad de la documentación, el lenguaje en el que se ha desarrollado, versión…

A continuación, se exponen las técnicas de selección y clasificación de componentes de un repositorio, más importantes [23].

- *Indización*: Es un proceso de clasificación realizado mediante un análisis conceptual del componente. Este proceso se ayuda de un vocabulario controlado el cual establece un conjunto de términos que pueden ser usados para realizar la consulta o representar la información. Los términos del vocabulario pueden relacionarse mediante operadores lógicos (y, o, no).
- *Indización automática*: Las palabras clave son extraídas de forma automática de las propias descripciones del componente. Se utilizan métodos estadísticos y lingüísticos para la recuperación e indización de la información.
- *Facetas*: Se crea un esquema de clasificación que especifica algunos atributos llamados facetas que se usan como descriptores del componente. El esquema se compone de 4 facetas: función realizada por el componente, objetos manipulados por la función, estructura de datos donde la función tiene lugar y sistema al que pertenece la función. Los catálogos se construyen manualmente y la recuperación se realiza especificando términos para cada una de las facetas. Los componentes recuperados se ordenan mediante un valor de similitud el cual se representa mediante un grafo de distancia conceptual o con lógica difusa.
- *Marcos de conocimiento*: Se realiza un análisis lexicográfico, sintáctico y semántico de las especificaciones en lenguaje natural de los componentes. Se apoyan en una base de conocimiento para almacenar la información semántica sobre un dominio de aplicación específico. Esta base es muy poderosa pero a la vez es muy compleja de construir.
- *Especificaciones formales*: Las consultas que se hacen al repositorio son especificaciones formales de requisitos y éstas devuelven una lista de componentes que a su vez están especificados formalmente. Estos requisitos de estos componentes coinciden con los requisitos introducidos en la consulta. La ventaja de este tipo de consultas radica en que las especificaciones se centran en el comportamiento del componente, no presentan ambigüedad y proporcionan una mayor precisión. Su mayor desventaja es que muchos componentes no vienen especificados formalmente.
- *Redes neuronales:* Estructuran el repositorio de acuerdo a una similitud funcional de los componentes que almacena de manera que si unos componentes presentan un comportamiento parecido se almacenan cerca dentro del repositorio.

- *Hojear:* Emplea un método de visualización rápida en el que es el propio usuario el que maneja el catálogo para recuperar los componentes candidatos. Esto lo hace confuso e impracticable cuando el repositorio contiene gran cantidad de componentes.
- *Hipertexto:* La información se organiza como una red de nodos que se interconectan por medio de enlaces y relaciones, de manera que el usuario puede navegar por la red siguiendo los enlaces establecidos guiándose por la semántica de cada enlace. Su principal desventaja es que requiere de muchos recursos humanos para crear la red.

3.6 Componentes en UML

Sus siglas significan Lenguaje Unificado de Modelado (UML) [29]. Es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad. Muchos procesos de desarrollo basado en componentes utilizan este lenguaje para modelar los componentes (como el que presentaremos más adelante de Chessman y Daniels) durante su creación, por eso se cree conveniente hacer un breve resumen de en qué consiste. Aún cuando todavía no es un estándar oficial, está respaldado por el OMG (*Object Management Group*). Es un lenguaje gráfico para visualizar, especificar, construir y documentar un sistema de software. Es importante resaltar que UML es un lenguaje para especificar y no para describir métodos o procesos. Se utiliza para definir un sistema de software, para detallar los artefactos en el sistema y para documentar y construir.

En UML 2.0 existen 13 tipos diferentes de diagramas. Los diagramas de estructura para enfatizar los elementos que deben existir en el sistema (diagrama de clases, componentes, objetos, estructura compuesta, de despliegue y de paquetes), diagramas de comportamiento que enfatizan lo que debe suceder en el sistema (diagrama de actividades, casos de uso y estados) y los diagramas de interacción que enfatizan sobre el flujo de control y de datos entre los elementos del sistema (diagrama de secuencia, de colaboración, de tiempos y de vista de interacción).

Los Componentes UML utilizan los estereotipos para extender UML de modo que cada elemento puede tener un estereotipo añadido. El nombre del estereotipo va encerrado entre <<>>. Los estereotipos son una manera de dejar a los usuarios crear nuevas clases de elementos de modelado UML Por ejemplo, para los diagramas de tipos de datos se utilizan los estereotipos <<*type*>> y <<*datatype*>>; para los diagramas de especificaciones de interfaz se utilizan estereotipos como <<*interface type*>> o <<*info type*>>; para los diagramas de especificaciones de componente se utilizan estereotipos como <<*comp spec*>> o <<*offers*>>. En el diagrama de arquitectura de componentes se pueden utilizar todos los estereotipos.

3.7 Procesos de desarrollo basado en componentes

Por último, en este apartado vamos a presentar dos procesos de desarrollo basados en componentes frente al desarrollo de software de manera tradicional, siendo el de Chessman y Daniels el que ampliaremos en la siguiente sección ya que es en el que nos basamos para definir COTSRE+.

3.7.1 Proceso de Qureshi y Hussai

Un ejemplo de proceso de desarrollo basado en componentes es el propuesto por Qureshi y Hussain [26]. Basándose en el modelo de orientación a objetos crean uno basado en componentes que sigue un ciclo de vida en espiral el cual tiene las siguientes fases:

- *Comunicación*: Se le comunica al cliente el comienzo del proyecto para proceder a la recogida de requisitos. Se desarrollan los casos de uso iniciales.
- *Planificación*: Se prepara un documento de especificación del proyecto compuesto por la viabilidad y el análisis de riesgo. Este documento se presenta al cliente para que de el visto bueno a la propuesta y se comience la fase de análisis.
- *Análisis y selección de componentes*: El analista recoge los requisitos básicos y trata de identificar y seleccionar los componentes que puedan ser reusados a partir de un repositorio. Se identifican las relaciones entre los componentes y sus propiedades y comportamientos. En esta fase se pretende reusar el mayor número de componentes posibles en vez de perder el tiempo en algo que ya está inventado. De esta manera, la productividad y eficiencia del desarrollo se incrementa. El repositorio es utilizado para almacenar y manejar los componentes reutilizables, sus funciones son: clasificación, búsqueda, modificación, prueba, implementación, control de versiones, cambios de control y documentación actualizada y consistente. Además podrán utilizarse más de uno siendo estos categorizados según el dominio de los componentes para hacerlos más efectivos.
- *Ingeniería y prueba*: Los componentes modificados se modifican según los requisitos del nuevo sistema a desarrollar y se prueban. Los nuevos componentes son diseñados, desarrollados y probados en unidades básicas.
- *Evaluación*: El cliente evalúa y verifica si el software cumple sus requisitos o no.

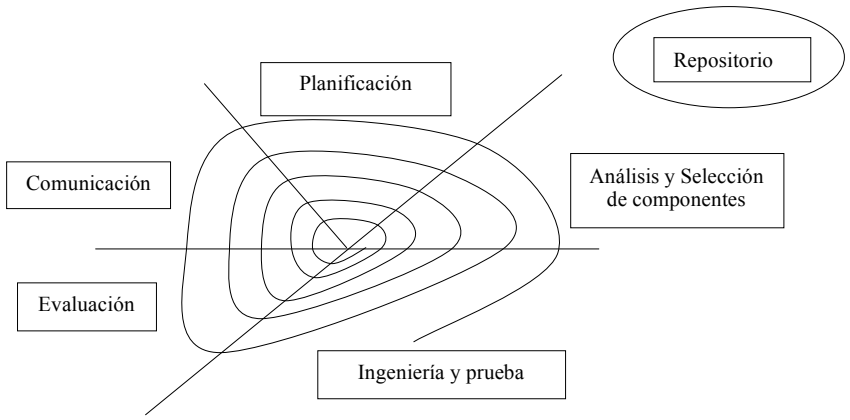


Figura 2. Proceso de Qureshi y Hussain

3.7.2 Proceso de Chesman y Daniels

En la siguiente figura podemos ver un esquema con las etapas que forman el proceso de desarrollo basado en componentes de Cheesman y Daniels [1], se muestra un esquema a alto nivel de este proceso. Los bloques representan conjuntos de actividades que dan lugar a resultados tangibles, las flechas gruesas representan su secuenciación y

las flechas finas representan el flujo de elementos generados que transfieren información entre ellas.

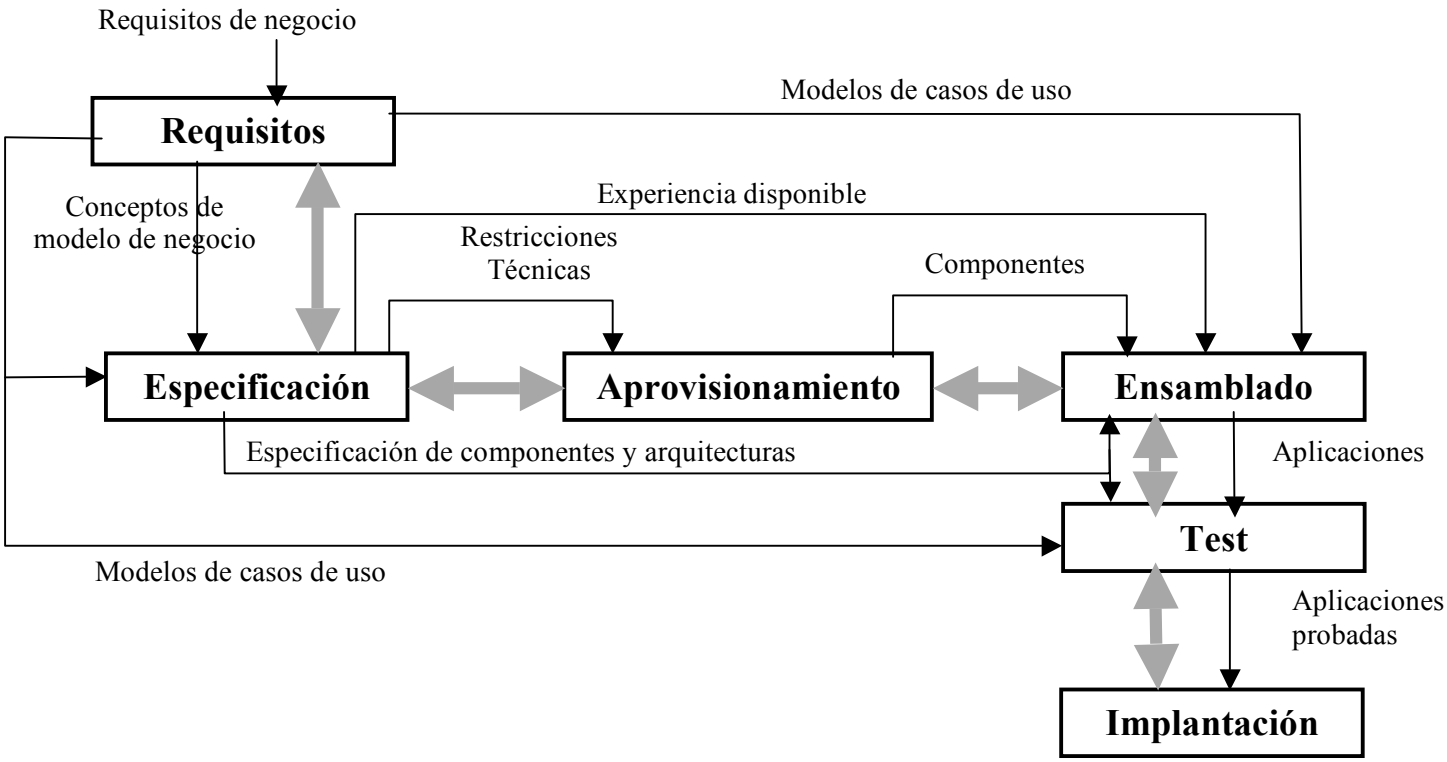


Figura 3. Proceso de Chessman y Daniels

Como observamos en la figura este proceso se divide en 6 fases:

1. *Fase de requisitos*, donde se identifican los requisitos de los componentes, de la arquitectura del software y del sistema en general.
2. *Fase de especificación*, donde se realiza la especificación de los componentes y de la arquitectura de software. Esta a su vez se divide en 3 etapas:
   2.1. *Identificación de componentes*.
   2.2. *Interacción de componentes*.
   2.3. Especificación de componentes.
3. *Fase aprovisionamiento de componentes*, donde se buscan y seleccionan componentes.
4. *Fase de ensamblaje* de los componentes encontrados y producidos.
5. *Fase de pruebas*.
6. *Fase de implantación* de la aplicación del sistema.

3.7.3 Comparación con el desarrollo de software tradicional

Las ventajas más importantes del modelo de desarrollo basado en componentes respecto al método tradicional [11, 25] son:

- Reutilización: No hay necesidad de volver a realizar trabajo ya hecho. Pérdida de tiempo que podríamos aprovechar en desarrollar nuevas funcionalidades.
- Interoperabilidad: Los componentes pueden comunicarse entre sí con lo que se facilita la integración de componentes en otras aplicaciones.

- Actualización: Reemplazo automático de componentes sin necesidad de cambios en el código.
- Menor complejidad. Gracias a la reutilización.
- Tiempo en vigencia.
- Coste efectivo. Se reduce el coste de desarrollo.
- Eficiente
- Confiabilidad: Los componentes que se reutilizan ya han sido probados y validados.
- Mejora de la calidad.

Algunos inconvenientes que se encuentran son [11]:

- Los desarrolladores que han adquirido un componente es muy posible que no tengan acceso al código fuente para modificar cualquier funcionalidad.
- Las especificaciones y documentación ofrecidas por el fabricante de un componente pueden ser escasas e incompletas.

4. Un proceso de DBC: Proceso de Chessman y Daniels

En este capítulo vamos a estudiar en detalle el proceso de Chessman y Daniels [4], en el primer apartado presentaremos el proceso, en el segundo haremos un estudio de cómo los requisitos influyen en el proceso y en el último mostraremos nuestra valoración.

4.1 El proceso de Chessman y Daniels

Los trabajos anteriores de John Cheesman estuvieron relacionados con *Microsoft Repository Open Information Mode*l (OIM), en el meta modelo *Catalysis* junto a Desmond d'Souza y Alan Cameron, en UML en el que hizo una gran contribución y en *Advisor* (inspirado en *Catalysis*) que es un método de desarrollo basado en componentes, realizado junto a a John Dodd.

John Daniels es uno de los pioneros de los conceptos de orientación a objetos. A comienzo de los 90 desarrolló junto a Steve Cook el método *Syntropy* (del que se inspira *Catalysis*), y después continuó con OCL [33] y UML.

John Cheesman y John Daniels abordan el problema de diseñar y especificar un proceso de desarrollo basado en componentes. Su extensión pragmática de UML captura conceptos importantes de los componentes: las especificaciones de componentes, interfaces componentes, implementaciones componentes y objetos componentes.

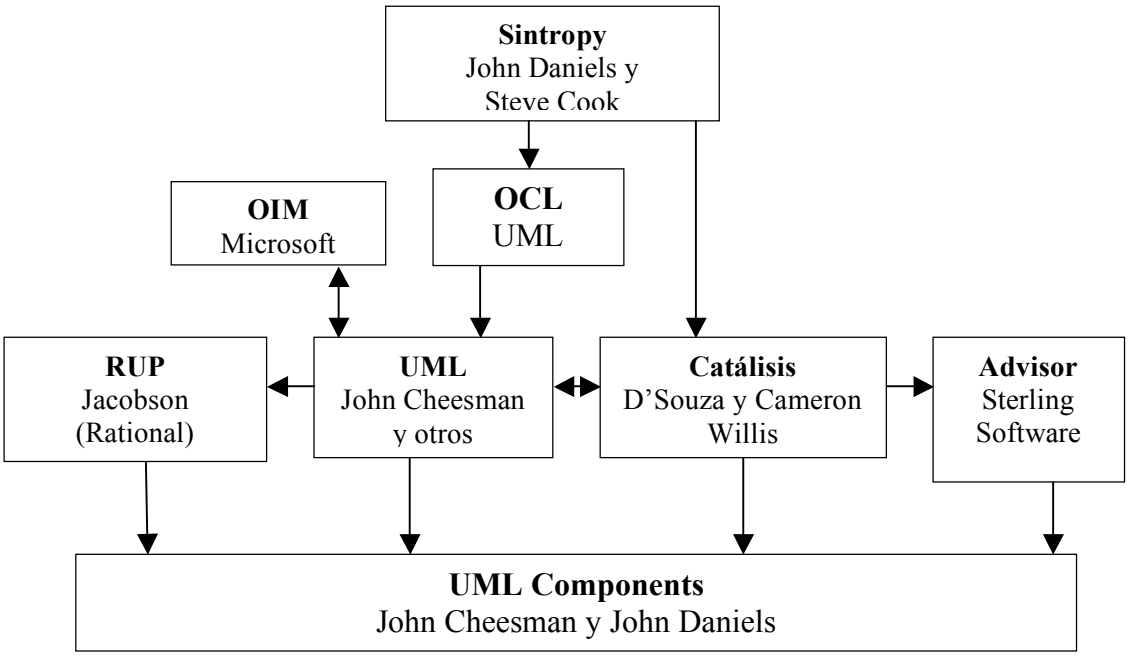


Figura 4. Evolución

El diseño por contrato es una idea básicamente simple y obligatoria para diseñar sistemas como cajas abstractas cooperando entre ellas. Los contratos enriquecen las operaciones de una interfaz mediante el uso de pre y post condiciones.

El proceso completo de desarrollo del componente cubre más que solo la especificación, se encarga de todas las actividades desde las reuniones para captar requisitos hasta el despliegue final del sistema.

Como ya hemos visto, uno de los fines del desarrollo basado en componentes es que el componente pueda ser reutilizable, pero además uno de los objetivos principales

que persiguen los autores es que sea fácilmente reemplazable. Cuando estamos construyendo una aplicación y utilizamos un componente puede ser que algún requisito cambie y que sea necesario sustituir el componente por otro para cumplir con los nuevos requisitos, pero sin que se vea afectado el sistema. Esto es tenido en cuenta a lo largo del proceso que proponen.

Los componentes son unidades de software que se basan en una serie de principios específicos:

1. Unificación de datos y funciones: Un objeto software se compone de datos y las funciones que manejan esos datos.
2. Encapsulación: El cliente no tiene conocimiento de la implementación interna del componente.
3. Identidad: Cada componente tiene una única identidad.

Además, esto hace que posean interfaces para poder utilizarlos, que junto con estos principios son los que permiten que un componente pueda ser actualizado o reemplazado por otro con un menor impacto sobre los clientes de ese componente. En la siguiente figura lo veremos mejor. Tenemos un componente que es utilizado por un cliente, pero en un determinado momento reemplazamos el componente por otro pudiendo ser utilizado con la misma interfaz por el cliente o por otros clientes que a su vez pueden utilizar el anterior componente.

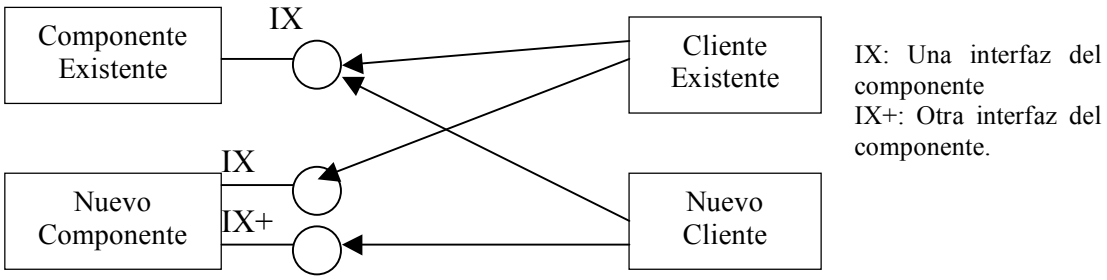


Figura 5. Interfaces componentes

Cheesman y Daniels identifican diferentes visiones en las que puede aparecer el término componente en las etapas de desarrollo de un sistema software, en concreto identifican hasta 5 formas de componente como se puede ver en la siguiente figura y que explicamos a continuación.

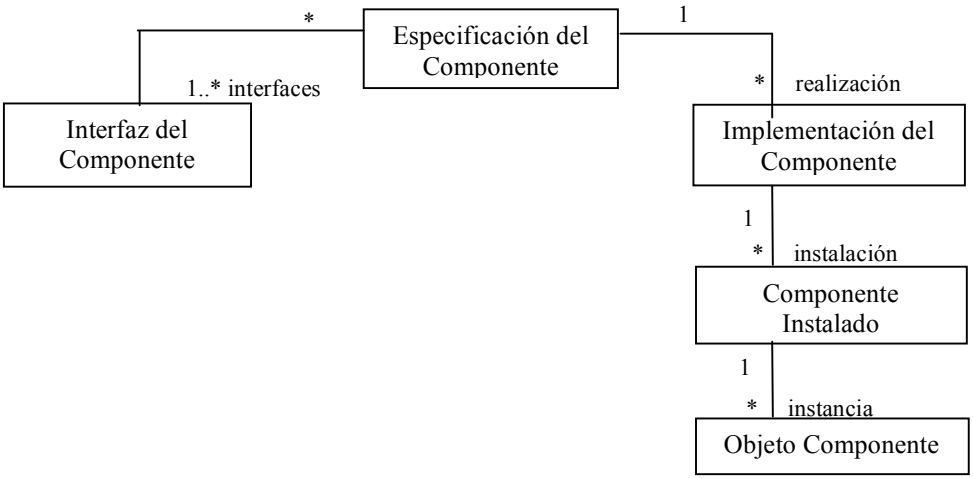


Figura 6. Visiones del componente

- *Especificación del componente*, que representa la especificación de una unidad software que describe el comportamiento de un conjunto de objetos componente

y que define una unidad de implementación. El comportamiento es definido por un conjunto de interfaces. Una especificación del componente se realiza como una implementación del componente.

- *Interfaz de componente*, es una definición de un conjunto de operaciones sobre los datos que pueden ser ofrecidos por un objeto componente.
- *Implementación del componente*, es una implementación de una especificación del componente.
- *Componente instalado*, es una copia instalada de una implementación de un componente.
- *Objeto componente*, que es una instancia de un componente aislado. Es un objeto con su propio estado e identidad única que lleva a cabo el comportamiento implementado. Un componente instalado puede tener múltiples objeto componentes o uno solo.

Arquitectura de componentes

Definimos arquitectura de componentes como un conjunto de componentes software a nivel de aplicación con sus relaciones estructurales y sus dependencias de comportamiento.

De las cuatro capas Cheesman y Daniels trabajan sólo con las dos últimas explicadas.:

- *Interfaz de usuario*: La capa de presentación de la información para los usuarios y la captura de entradas. Los sistemas externos pueden ser tanto usuarios como otros componentes.
- *Dialogo de usuario*: La capa de dirección del diálogo de usuario en una sesión.
- *Servicios del sistema*: La capa de representación externa del sistema, puede ser usada por varios diálogos de usuario.
- *Servicios del negocio*: Implementa las reglas de negocio, información y transformaciones.

Una arquitectura de componentes puede ser utilizada para modelar una aplicación de software (a partir de componentes) o para modelar un sistema software como un conjunto de aplicaciones (consideradas estas como componentes y subcomponentes). Las relaciones estructurales son asociaciones y relaciones de herencia entre especificaciones de componente e interfaces de componente, y relaciones de composición entre componentes. Las dependencias de comportamiento son relaciones de dependencia: (a) entre componentes, (b) entre componentes e interfaces, (c) entre interfaces, y (d) entre subcomponentes y componentes. Esto lo podemos ver de forma resumida en la siguiente figura.

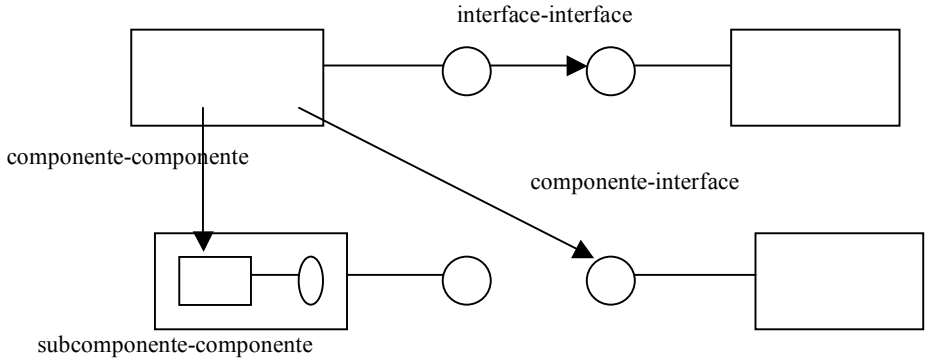


Figura 7. Relaciones estructurales de los componentes

Además, una arquitectura de componentes se puede centrar en especificaciones de componente, en implementaciones de componente o en objetos de componente. Un diagrama de una arquitectura de especificaciones de componente contiene sólo especificaciones de componente e interfaces. Un diagrama de una arquitectura de implementaciones de componente muestra las dependencias que existe entre las implementaciones de un componente en particular. Y por último, un diagrama de una arquitectura de objetos de componente especifica la relación entre las instancias de cada componente.

Contratos

En otras actividades, que no tienen nada que ver con el desarrollo, se utilizan contratos en los que se llega a un acuerdo entre el propietario y el cliente sobre los servicios que ofrece. También podríamos considerar una relación similar entre las compañías que desarrollan software y los componentes. En este sentido, las compañías se verán como entidades que proporcionan servicios (componentes) a sus clientes (que pueden ser otras compañías, organizaciones o clientes independientes) y que también pueden depender de los servicios de otras compañías. De igual forma que en la vida real los contratos sirven para cerrar acuerdos alcanzados entre dos o más partes (compañías, organizaciones, personas), éstos (los contratos) también pueden ser utilizados como acuerdos formales de especificación entre partes software.

Estos contratos y credenciales son otra forma de referirse a la información funcional (concretamente la semántica) y a la información extra-funcional de la especificación de un componente software. Los contratos son una forma de garantizar que las partes software de un sistema, desarrolladas por diferentes personas y posiblemente de diferentes organizaciones, puedan funcionar correctamente de forma conjunta.

Un contrato describe la relación entre una interfaz del objeto componente y un cliente, y se describe el contrato en forma de interfaz. La especificación contiene descripciones sintácticas de las operaciones de la interfaz y la descripción del comportamiento

Cheesman y Daniels identifican dos tipos de contratos, los contratos de uso y de realización.

- Los *contratos de uso* son entre la interfaz de un componente y sus clientes. En estos contratos se detallan el listado de operaciones sobre los datos internos incluyendo sus datos y definiciones, que son las que el cliente utiliza. También refleja la pre y post condiciones de manera que si el cliente cumple con la precondición, el componente cumplirá la postcondición.
- Los *contratos de realización* son entre una especificación de componente y su implementación. Es un contrato en tiempo de diseño, con el se especifica que la implementación debe cumplir una serie de requisitos reflejados en este contrato.

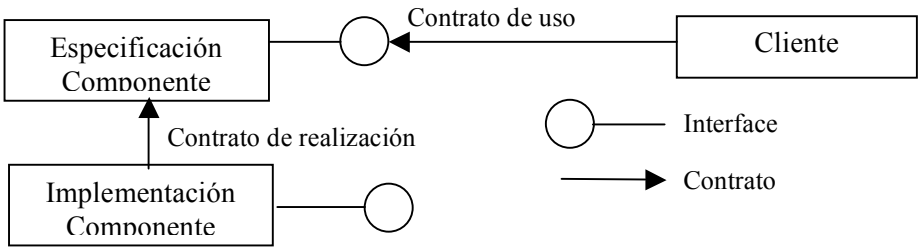


Figura 8. Tipos de contrato

4.1.1 Fase de requisitos

Lo primero de todo el desarrollador del sistema debe entrevistarse con los usuarios finales, analistas de la empresa y expertos en el dominio para crear un proceso de negocio.

La descripción del proceso de negocio seguramente presentará términos que son necesarios aclarar para un mejor entendimiento del proceso. Con estos términos podemos realizar un modelo de concepto de negocio definido en notación UML. Este diagrama no representará ningún componente software, aunque es posible que luego existan elementos en el software con el mismo nombre, esto sería una decisión de diseño. Cada concepto puede tener asignado unos atributos, aunque en este diagrama no es obligatorio.

Debemos tener una visión del sistema que identifique los límites del sistema, qué será responsabilidad del software y que no, cómo trabaja el usuario con el sistema, qué representa para él el sistema... Esta visión consiste en una pequeña descripción con palabras de lo que se espera del sistema y una idea de su funcionamiento. Esta descripción sería proporcionada por el cliente del sistema y es la base de la que se parte.

Una vez que tenemos la visión y el modelo del concepto de negocio podemos asignar responsabilidades a la definición del proceso de negocio. Esta clase de decisiones de responsabilidad a menudo se hacen demasiado rápido pero nos ayudan inicialmente y luego podemos ir refinándolas.

Más o menos podemos tener identificados los actores del sistema y el rol que juegan en él. Es hora de realizar los casos de uso. Podemos mostrar cómo los actores se relacionan con nuestro modelo de concepto del negocio usando generalización.

Con los casos de uso reflejamos claramente cuales son los límites del software y permite que nosotros digamos cómo el software cubrirá sus responsabilidades. Crear casos de uso permite adentrarnos en nuestra visión del sistema. Son una especificación funcional del sistema software, tratando el software como una caja negra con respecto a su estructura interna y organización. Un evento provoca una secuencia de pasos en un proceso, esta debe estar numerada para que quede claro el orden que se sigue para tratar el evento. Este guión puede tener extensiones para casos especiales que pudieran ocurrir, como por ejemplo excepciones.

Al modelar los casos de uso puede ser que nos demos cuenta de la necesidad de crear nuevos actores y nuevas funcionalidades del software.

Sería necesario añadir una sección que tratara la calidad de servicio. Cada caso de uso tendrá unas expectativas que cumplir en el área de la seguridad y el rendimiento. Por ejemplo,

Se introducirá un usuario y contraseña para entrar en la aplicación. (Seguridad).

La aplicación no soportará más de 100 usuarios conectados a la vez. (Rendimiento).

4.1.2 Fase de especificación

La fase de especificación está dividida en tres fases: identificación, interacción y especificación.

4.1.2.1. Identificación de componentes

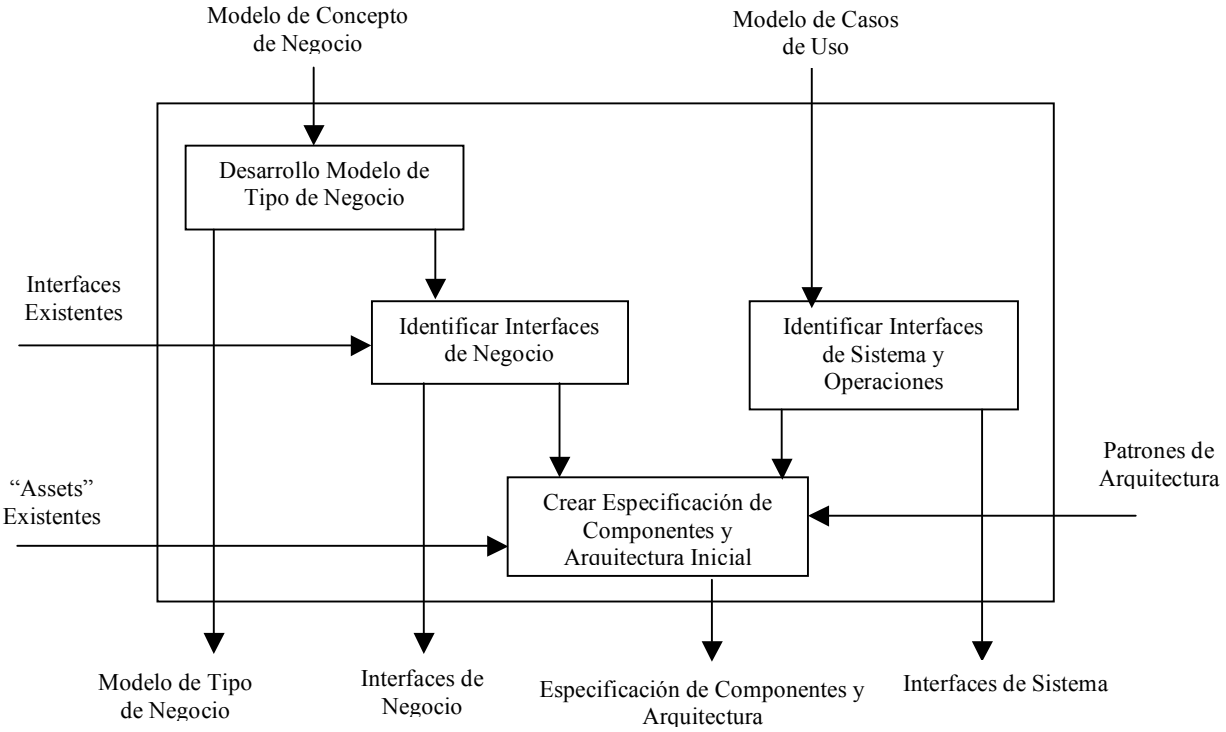


Figura 9. Fase de identificación de componentes

Esta fase tiene como entradas el modelo de concepto del negocio y el modelo de casos de uso creados en la fase de requisitos. El objetivo principal de esta fase es crear un conjunto inicial de interfaces y especificaciones de componentes que formen una arquitectura inicial de componentes, qué información necesita el componente para ser manejado, qué interfaces son necesarias para manejarlo, que componentes necesitan suministrar esa funcionalidad y cómo encajaría todo esto.

La separación en dos capas distintas, la capa de servicios del sistema y la capa de los servicios de negocio que explicamos anteriormente va a quedar clara aquí en esta fase. Vamos a identificar las interfaces del sistema y los componentes del sistema en la capa de servicios del sistema y las interfaces de negocio y los componentes en la capa de servicios de negocio.

Del modelo de casos de uso surgen las interfaces del sistema y sus operaciones iniciales y del modelo de concepto del negocio enfocamos la información y los procesos asociados que el sistema necesitará manejar. Cada uno de los casos de uso y para cada paso de ellos, consideramos si hay responsabilidades del sistema que deben ser moldeadas, si es así lo representamos con una o más operaciones de la interface del sistema apropiada. Debemos tener en cuenta también las extensiones que puedan tener los casos de uso para identificar las operaciones. No hace falta en esta fase identificar los parámetros de las operaciones encontradas, esto se hará en la fase de interacción de componentes.

Usamos el modelo de negocio para crear otro modelo de tipo de negocio que representará la vista global del sistema, el cual nos ayuda a desarrollar un conjunto de interfaces del negocio. Las interfaces del negocio son abstracciones de la información que deben ser manejadas por el sistema, el proceso para identificarlas se compone de los siguientes pasos:

1. *Crear el modelo de tipo de negocio:* Cogemos el modelo de concepto del negocio que nos proporciona la fase de requisitos y lo transformamos en un modelo de tipo del negocio. Para ello hacemos una copia del modelo de negocio, podríamos hacer el diagrama de tipo sobre él pero es mejor guardar una copia para tener como un historial del proceso de desarrollo del sistema. El modelo de tipo de negocio es representado por un diagrama de clase de UML, semejante al del modelo de concepto del negocio. Este modelo de tipo del negocio contendrá información más específica que el anterior.
2. *Refinar el modelo de negocio y refinar cualquier regla adicional con las restricciones:* Refinamos el modelo de tipo de negocio añadiendo y eliminando elementos. De esta manera mejoramos el modelo con cualquier detalle que haya sido omitido en el nivel de concepto, en particular los detalles de los atributos en cada tipo, definiendo un conjunto de tipos de datos para usar en este modelo y definiendo las restricciones como asociaciones múltiples.
3. *Definir las reglas de negocio:* Ahora empezamos a añadir al modelo cualquier regla de negocio que se requiera, escribiendo restricciones y añadiendo nuevos atributos. Las restricciones pueden ser escritas utilizando el lenguaje natural pero también podemos utilizar lenguajes como OCL (*Object Constraint Language Specification*) [33] para hacer una restricción en un lenguaje formal. Si decidimos definirlas en OCL deberán ser escritas entre llaves {} indicando sobre que asociación se aplica la regla.
4. *Identificar los núcleos principales de negocio:* El propósito de identificarlos es tener conocimiento de que información será dependiente de que otra. Es un paso necesario para luego poder designar responsabilidades para las interfaces. Un tipo principal es aquel que tiene existencia independiente del sistema. Para identificar los tipos principales les damos el esterotipo <<core>> de UML. Todos los otros tipos proveen los detalles de los tipos principales.
5. *Crear las interfaces de la empresa y asignar responsabilidades:* Creamos una interfaz de negocio por cada tipo principal en el modelo de tipo de negocio. Cada interfaz de negocio maneja la información representada por el tipo principal y sus tipos detallados. A estas interfaces las llamamos IxxxMgt, donde xxxx representa el nombre del tipo principal que manejan, por ejemplo, si tenemos el tipo principal Subasta su interfaz manejadora sería ISubastaMgt.

   Para las asociaciones debemos ser capaces de reducir las dependencias. Cuando existe una relación entre dos tipos manejados por diferentes interfaces se produce una asociación inter-interface, y en estos casos debemos tomar la decisión de quien será el que guarde la información para que se produzca la mínima dependencia. Asignamos direcciones de referencia a las asociaciones para definir de forma más precisa que información tiene que mantener cada interfaz.

Hemos identificado las interfaces del sistema que son obtenidas de los casos de uso y tenemos las interfaces de negocio obtenidas del modelo de tipo de negocio. Las interfaces del sistema son creadas directamente en sus carpetas de especificación de interfaces mientras que las interfaces de negocio serán creadas en el modelo de tipo del negocio. Además debemos añadir interfaces adicionales que son parte del entorno donde se ejecutará el sistema.

El siguiente paso será crear una arquitectura inicial de especificación del componente. Crearemos un conjunto inicial de especificaciones del componente y constituiremos una idea de cómo podría encajar bien todo junto. El componente es la unidad de ejecución y de reemplazamiento en un sistema componente. Esta parte es

muy importante porque debemos especificar componentes de manera que tenga sentido para desarrollarlo o comprarlo.

Para crear esta arquitectura podemos recibir varias entradas:

- Las interfaces del de negocio y sistema.
- Especificaciones de componentes existentes que vayamos a reutilizar.
- Arquitecturas de especificación de componentes que necesitemos adaptar.
- Componentes de pago.

Lo normal es utilizar una interfaz por cada especificación de componente aunque también podemos encontrar que sea necesario crear múltiples interfaces sobre la misma especificación de componente. Esto podría llegar a darse, por ejemplo, porque la interfaz del componente es muy compleja.

4.1.2.2 Interacción de componentes

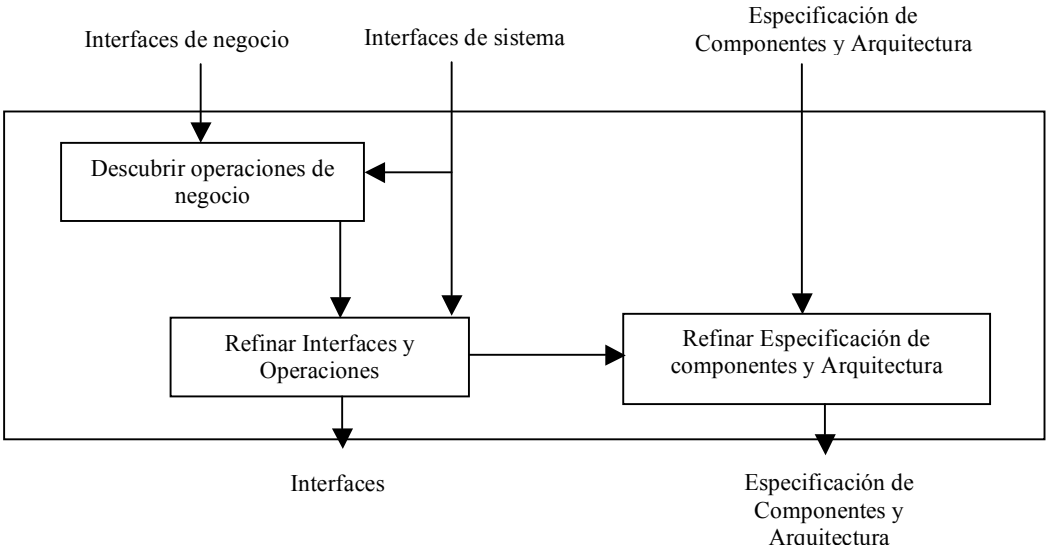


Figura 10. Fase de interacción de componentes

En esta fase determinamos como trabajarán en conjunto los componentes para conseguir la funcionalidad requerida. El modelo de interacción lo usamos para definir las interacciones que tienen lugar dentro del sistema, para refinar las definiciones de las interfaces existentes, para identificar cómo son usadas las interfaces y para descubrir nuevas interfaces y operaciones.

Empezamos fijándonos en las operaciones del negocio. No conocemos su signatura todavía, ni como serán implementadas usando los componentes de negocio. Tampoco tenemos identificados las operaciones necesarias en las interfaces de sistema.

Es importante recalcar que estamos tratando de producir una especificación, no un diseño de implementación y que debemos ser cuidadosos para evitar una sobreespecificación.

Para descubrir las operaciones de interfaz de negocio tomamos cada operación de la interfaz de sistema por turno y dibujamos uno o más diagramas de colaboración que trazan cualquier restricción en los flujos de la ejecución que resulta de una invocación de esa operación. Cada diagrama de colaboración debe indicar una o más interacciones donde cada interacción indica un flujo de ejecución posible. Así que si hay algún flujo de interacción alternativo importante se necesitará dibujar las interacciones.

Empezamos definiendo la signatura de cada operación, en ella deben aparecer los parámetros que necesita para ejecutarse y la salida que produce. Además debemos

definir los tipos de datos con los que esta operación va a trabajar y si es necesario los creamos. Una decisión importante es decidir quién es el encargado de llamar a esta operación, debemos definir claramente las responsabilidades y delegaciones de cada componente.

Debemos asegurar que las referencias intercomponentes son legítimas, haciendo una buena asignación de la responsabilidad. Tenemos varias alternativas para conseguirlo:

- La responsabilidad puede ser destinado al objeto componente almacenando la referencia, dejándole la responsabilidad total y exclusiva.
- La responsabilidad puede ser asignada al objeto componente que posee la referencia.
- La responsabilidad puede ser dado a un tercer objeto componente que esté más arriba de la cadena de llamada.
- También podríamos permitir y tolerar las referencias inválidas.
- O rechazar la supresión de información.

Según las dependencias elegidas entre nuestros componentes elegiremos la alternativa que mejor se adapte a nuestro sistema.

Durante el proceso de creación de la especificación de las operaciones no nos hemos preocupado de la minimización de las llamada, las dependencias cíclicas, la normalización de las operaciones e interfaces, usar dibujos de diseños existentes... Es normal, porque esta ha sido una fase de descubrimiento, ahora es el momento de realizar la factorización.

La factorización en una interfaz involucra dividir sus responsabilidades entre dos o más interfaces, estas serían subtipos. La factorización es aplicable en las operaciones en el sentido de que buscamos la generalidad y no la redundancia en las operaciones de interfaz donde son apropiadas.

Debemos tener cuidado de no tratar de prever futuros requisitos para construir una capacidad extra en las operaciones y las interfaces. Cuando los nuevos requisitos aparecen el soporte puede ser proveído añadiendo nuevas interfaces.

4.1.2.3 Especificación de componentes

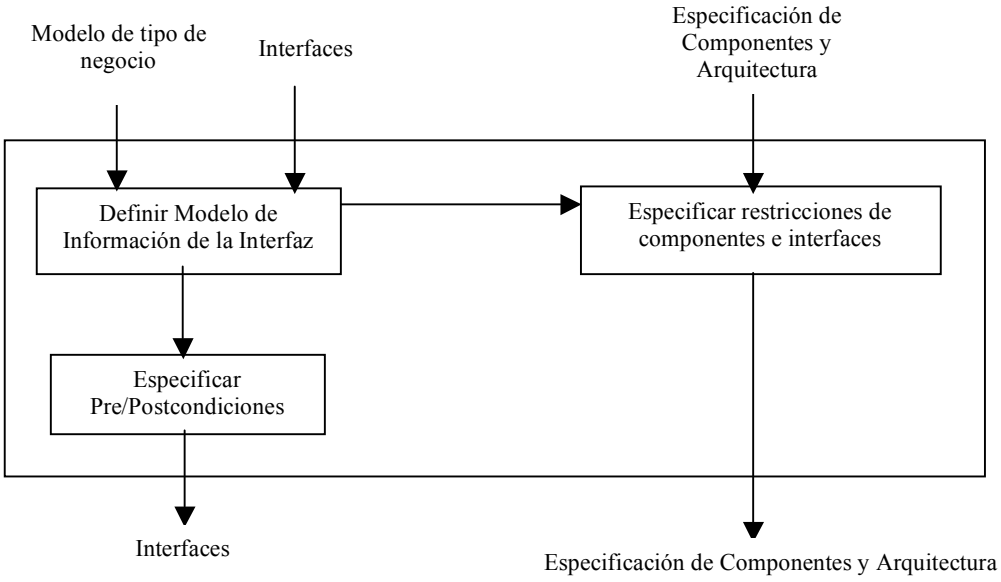


Figura 11. Fase de especificación de componentes

En esta fase definiremos los contratos de uso y realización. Un contrato de uso es definido por una especificación de interfaz, examinaremos lo que hace una interfaz y la dividiremos en varias partes. Un contrato de realización es definido por una especificación de componente, pero las especificaciones de componente pueden también contener las restricciones sobre la manera en que las interfaces son implementadas.

Una interfaz es un conjunto de operaciones. Cada operación define algún servicio o función que un objeto componente efectuará para un cliente. Una operación entonces representa un contrato entre el cliente y un objeto componente.

Una operación especifica una acción individual que un objeto componente hará para un cliente. Esto tiene varias facetas:

- Los parámetros de entrada: Especificar la información proveída o pasada al objeto componente.
- Los parámetros de salida: Especificar la información actualizada o devuelta por el objeto componente.
- Cualquier cambio dará como resultado el estado del objeto componente.
- Cualquier restricción que sea aplicable.

La operación tiene que especificar cómo las entradas, las salidas y el estado del componente están relacionados y qué efectos de la llamada a la operación tiene en esa relación. No incluyen la información sobre las interacciones entre el objeto componente y las operaciones de otros objetos componentes que son exigidos, en una implementación específica, para completar la operación.

Tenemos que representar el estado del objeto componente sobre el cuál la interfaz depende, cada interfaz tiene un modelo de información de la interfaz. El modelo de la información de interfaz tiene que contener justo lo que permiten las operaciones de interfaz especificadas.

Cada operación tiene una pre y postcondición, esto nos ayuda a especificar el efecto de una operación pero sin escribir el algoritmo o su implementación. Sería como la letra pequeña en un contrato con el cliente. Especifica al detalle que hará la operación y siempre deben ir juntas.

La postcondición especifica el efecto de la operación si la condición previa o precondición es verdadera. La precondición no es una condición bajo la cual deba llamarse a la operación, la invocación de la operación es totalmente independiente de ella. Es la condición bajo la que la operación garantiza que la postcondición es verdadera. Si la precondición es falsa cuando la operación es invocada la postcondición no se cumplirá y el resultado será incierto.

La precondición representa las suposiciones en las que se basa la operación para el correcto funcionamiento, y la postcondición representa las garantías contractuales que las operaciones marcan si esas suposiciones están bien fundadas. Las suposiciones son la responsabilidad del cliente de la operación mientras que las garantías son la responsabilidad del proveedor de la operación.

La postcondición nos garantiza que si la condición previa es verdadera, algo no mencionado en la postcondición es asumido que no cambiará. Por el contrario, si la condición previa es falsa, el efecto es indeterminado realmente y el cliente no puede hacer ninguna suposición sobre que puede o no haber cambiado.

En UML las condiciones de las precondiciones y las postcondiciones suelen estar escritas en el lenguaje OCL, de esta manera las expresiones de las condiciones tienen una interpretación inequívoca, no habrá dudas de lo que se quiere decir. Las expresiones OCL pueden referenciar los parámetros de la operación, el resultado de la operación y el estado del objeto componente. Al resto de cosas no pueden referirse.

Lo que se suele hacer es esbozar las condiciones en el lenguaje humano primero y después hacer una definición formal en OCL basándose en lo anterior, para volver a refinar lo escrito en lenguaje humano y hacer modificaciones en OCL de nuevo.

Una técnica útil cuando se escriben las condiciones es dibujar el antes y el después en el diagrama de instancias y subrayar los cambios de estado que ocurren. Estos diagramas deben ajustarse al modelo de información de la interfaz. El estado antes no ayuda para que definamos la precondición y el estado después a definir la postcondición.

Un invariante es una restricción adjunta a un tipo que debe ser validada por todas las instancias del tipo. Los invariantes pueden ser expresados gráficamente utilizando la notación de UML. El invariante también puede ser escrito en OCL o utilizar el lenguaje natural.

Las interfaces del sistema también deben ser especificadas en relación con su modelo de información de interfaz local. Como con cualquier otra interfaz, el modelo de información de interfaz de un sistema necesita contener la información justa para permitir que las operaciones sean especificadas. Este será un subconjunto del modelo de tipo de negocio. Se empieza realizando una copia del modelo de tipo de negocio y se adjuntan las reglas de negocio como invariantes con el propósito de formar parte del software. Entonces las reglas de negocio podrían ubicarse en las interfaces del sistema solamente, en las interfaces de negocio o en ambas, dependiendo de si se quiere que esa regla sea visible a los clientes o no, o no es importante que estén repetidas en los dos tipos de interfaz.

En la especificación de componentes se considera la información de especificación adicional de la que el implementador del componente y el ensamblador necesitan ser conscientes, sobre todo de las dependencias de un componente con otras interfaces.

Para cada especificación de componente debemos de decir qué interfaces realizadas son soportadas. También necesitamos confirmar cualquier restricción con respecto a qué otras interfaces son usadas por una realización. Estas restricciones aparecerían en la arquitectura del diagrama componente como flechas de dependencia.

Las restricciones sobre como debe ser implementado una operación en particular son definidas en las interacciones. Las interacciones de componentes definen las restricciones a nivel de especificación. Todas las realizaciones de los componentes deben respetar estas restricciones, lo cual es útil después si se quiere reemplazar el componente perteneciente a un sistema muy complejo.

Las interacciones que hacen las restricciones sobre las especificaciones de componentes son fragmentos de las interacciones que dibujamos durante la operación de descubrimiento. Son fragmentos que comienzan con un objeto componente de la clase que está siendo especificada, recibiendo un mensaje y mostrando solamente las interacciones directas de ese componente.

4.1.3 Fase de aprovisionamiento

Esta fase se encarga del aprovisonamiento de componentes para cubrir la especificación requerida. El propósito de este aprovisionamieto es averiguar si necesitamos crear un código fuente para las implementaciones o si existe algún componente ya existente que cumpla con la especificación y que podamos comprar o utilizar. Estas especificaciones se han tratado que sean lo más independiente posible de las tecnologías actuales, pero es posible que a la hora de implementar un componente en alguna de ellas nos encontremos que no nos ofrecen exactamente lo que nuestra

especificación requiere. Con tecnología no solo nos referimos al lenguaje de programación, sino también a las características del entorno donde se ejecuta, la seguridad, las transacciones...

Los dos ambientes actuales más extendidos son COM+ de Microsoft para la plataforma Windows y Enterprise JavaBeans que utiliza el lenguaje Java y es independiente de la plataforma.

Se debe hacer un mapeado entre nuestras especificaciones y la tecnología en la que se tenga en cuenta esto:

- Tipos de parámetros de la operación, la clase y restricciones de referencias. Tendremos dos tipos de parámetros, cualquier dato que se pasarán por valor y los objetos que se pasarán por referencia, así que sin considerar el lenguaje de implementación debemos atenernos a estos dos tipos.
- Mecanismo de manejo de excepciones y errores.
- Herencia de interfaz y restricciones soportadas.
- Secuencia de operación.
- Propiedades de la interfaz.
- Mecanismo de creación de objetos.
- Manejo de eventos.

4.1.4 Fase de ensamblaje y pruebas

El ensamblaje es el proceso de colocar los componentes juntos y los elementos del software existente en un sistema de trabajo, y diseñar una interface para este sistema basada en los casos de uso para formar la aplicación.

Las entradas a esta fase serían los modelos de los casos de uso, los componentes, los elementos existentes y la especificación de componentes y su arquitectura, dando como salida la constitución de la aplicación del sistema.

Esta fase es la encargada de definir la interface de usuario y el diálogo lógico de usuario de los que hablábamos en apartados anteriores.

Una vez realizada la aplicación pasaría por una fase de testeo para comprobar su funcionamiento y encontrar posibles errores.

4.2 Estudio de cómo afectan los requisitos a cada una de las fases del proceso.

En el proceso descrito vemos toda la evolución del proceso de creación de una serie de componentes para un sistema, empezamos con una sencilla especificación del sistema y a partir de ahí vamos identificando los componentes, sus atributos, operaciones, sus dependencias, como interaccionan entre ellos... Empezamos por un componente general hasta tenerlo bien diseñado. Lo mismo ocurriría con los requisitos. Siguiendo este mismo desarrollo podemos ir descubriendo a la vez que diseñamos los componentes, cuales son los requisitos de los componentes. Conforme añadamos o cambiemos algo en los componentes esto se verá reflejado en sus requisitos.

Los requisitos podemos dividirlos en dos tipos si atendemos a la funcionalidad, requisitos funcionales y no funcionales.

- *Requisitos funcionales*: Describen la funcionalidad o los servicios que el sistema proveerá, sus entradas y sus salidas, excepciones, etc.
- *Requisitos no funcionales*: Se refieren a las propiedades emergentes del sistema como la fiabilidad, el tiempo de respuesta, la capacidad de almacenamiento, la

capacidad de los dispositivos de I/O, y la representación de datos que se utiliza en las interfaces del sistema.

También podemos encontrar otros tipos de requisitos:

- *Requisitos estructurales*: Tienen que ver con la arquitectura del sistema, con sus aspectos estáticos.
- *Requisitos dinámicos*: Son los que indican un cambio de estado, evolución en el tiempo, etc.
- *Requisitos temporales*: Indican duración, precedencia en el tiempo, etc.

También podrían clasificarse en requisitos del hardware y del software, pero para este caso sólo veremos los del software.

4.2.1 Fase de requisitos

El desarrollador realiza una serie de entrevistas con los usuarios finales del sistema y los expertos del dominio para captar los principales requisitos del sistema. A partir de la información obtenida podremos ir generando los primeros requisitos. Estos serán muy generales pero nos darán una visión de lo que se espera del sistema y los componentes.

De la descripción del proceso del sistema se genera un modelo de conceptual del negocio. Este modelo al ser muy general y esquemático no nos aporta muchos requisitos pero nos sirve para identificar los términos mas relevantes del sistema. Por ejemplo, si nuestra aplicación se encarga de gestionar un videoclub, con este diagrama identificamos a los clientes, las películas, los videojuegos, pagos, vendedor,... Así que podemos identificar los primeros requisitos sobre lo que tiene el sistema o cómo se relaciona.

"Un videoclub tiene uno o más usuarios."

"Un usuario puede retirar una película."

Estos requisitos se irán refinando en las siguientes fases, ya que este modelo es muy esquemático y se utiliza para tener una mejor comprensión del sistema al principio. Más tarde puede ser que desechemos algunas entidades o nos demos cuenta que algunas relaciones pueden variar, por lo que los requisitos se verán modificados también.

A partir de la visión del sistema podemos reconocer los requisitos sobre los límites del sistema. Estos requisitos nos serán de gran utilidad para saber exactamente qué es lo que se desea de los componentes de la aplicación. Por ejemplo, "La aplicación gestionará los préstamos de los usuarios." Y por ejemplo, no ordenará las películas de la estantería. Esto no se modelará como requisito porque una vez tengamos los requisitos de la visión sabremos qué es lo que debe hacer.

Con los casos de uso vamos a recoger los requisitos funcionales del sistema a partir de las funciones que pueden realizar los actores sobre él. En el ejemplo del videoclub podemos ver claramente que una funcionalidad sería la de alquilar una película. Según los pasos que creemos que debe seguir el sistema para realizar el alquiler podemos identificar muchos requisitos. Por ejemplo, un guión podría ser el siguiente.

1. El dependiente introduce el DNI del usuario.
2. El sistema busca al usuario en una base de datos.
3. El sistema comprueba los datos del usuario.
4. El dependiente introduce el nombre de la película.
5. El sistema realiza la reserva y devuelve un comprobante.

Todos estos pasos de este sencillo guión ya forman un requisito cada uno en el que se nos describe lo que se espera del sistema y como interactúa el usuario con él. En

un caso de uso descrito a alto nivel la descripción es muy general, normalmente se condensa en dos o tres frases. Es útil para comprender el ámbito y el grado de complejidad del sistema. Más adelante iremos refinando estos requisitos, por ejemplo, podremos decidir de que manera se busca al usuario en la base de datos, cómo se introducen los datos exactamente...

Las excepciones hemos dicho que también son requisitos funcionales, ¿que ocurre si el usuario no se encuentra en la base de datos? El guión especial que se sigue también tiene una serie de requisitos con iguales características que los anteriores, pero que se corresponderán con los casos excepcionales.

Cuando uno realiza casos de uso puede identificar nuevos componentes del sistema o relaciones que añaden nuevos requisitos o invalidan o refinan los requisitos ya hallados al principio. Por ejemplo, podemos ver claramente que podemos añadir una función para registrar nuevos usuarios del videoclub o nuevas entidades como el comprobante que se le da al cliente. Requisitos para estas nuevas funcionalidades serían,

"El sistema proporciona un comprobante con los datos del alquiler."

"El sistema gestiona las altas, bajas y modificaciones de los datos de los clientes."

En la sección de calidad de servicio se pueden añadir los primeros requisitos no funcionales que identifiquemos relativos con la seguridad y rendimiento del sistema. Correspondiente a la seguridad podríamos tener "Se introducirá el número del DNI del cliente." y al rendimiento, "La aplicación no soportará a más de 10 reservas a la vez."

4.2.2 Fase de especificación

4.2.2.1. Identificación de componente

En esta fase trabajaremos con los requisitos encontrados anteriormente e iremos refinándolos y añadiendo nuevos conforme vayamos profundizando en la especificación del sistema.

Con los casos de uso hemos recopilado una gran cantidad de requisitos de las funciones que podrán efectuar los actores del sistema. En esta fase se identifican las operaciones que tendrá cada funcionalidad con los que añadimos los requisitos de las operaciones. Estos requisitos serán sencillos ya que sabemos qué operaciones podemos utilizar pero todavía no vamos a identificar ni sus parámetros ni sus pre y postcondiciones hasta la fase de interacción.

El modelo de tipo de negocio es creado a partir del modelo conceptual de negocio con lo que cogemos los requisitos que hemos hallado a partir de este modelo y los adaptamos al nuevo modelo. En este modelo puede ser que nos demos cuenta que no necesitamos alguna entidad o que alguna relación ha cambiado entonces los requisitos de los que dependieran también deben eliminarse o modificarse para adaptarse al modelo de tipo de negocio. Este modelo al tener más detalle hace que los requisitos puedan enriquecerse también. En este modelo aparecen también los atributos de cada entidad, si en nuestro ejemplo teníamos una entidad Cliente podremos añadirle requisitos que forman parte de él. Si identificamos que el cliente tiene un atributo que es Nombre y Apellidos, podremos añadir un requisito que sea "El sistema guardará el nombre y apellidos de un cliente".

Al definir las reglas de negocio aparecen requisitos para relacionar unos atributos de una entidad con otros. Por ejemplo, la entidad Cliente tiene un DNI y al

igual que una entidad de Registro, podríamos tener un requisito que fuera “El dni del cliente coincidirá con un dni del registro para realizar el alquiler”.

El tener identificados los núcleos principales del sistema nos permite saber el punto de partida a la hora de añadir requisitos, para el sistema de un videoclub los más importantes podrían ser el Cliente y Productos (Películas y Videojuegos). Si empezamos a crear los requisitos para estas entidades los requisitos para el resto se derivan de las relaciones entre la entidad con la entidad principal. Por ejemplo, los requisitos para una entidad Alquiler se derivan de la relación existente entre los datos del cliente y el producto que alquila, “Un usuario registrado podrá alquilar 3 películas a la vez.”

Para las interfaces de cada entidad se crean requisitos sobre las operaciones que podremos realizar sobre estas entidades, estos requisitos nos servirán para saber como acceder a un tipo.

Todos estos requisitos de este modelo nos ayudan a la hora de crear la arquitectura inicial del sistema. Con esta arquitectura identificamos los componentes que son necesarios crear o comprar, por lo que si tenemos bien definidos los requisitos de un tipo podremos escoger mejor un componente que se adapte a nuestras necesidades. De momento sabemos ya los requisitos generales que deseamos que cumplan los atributos de cada tipo o componente y los requisitos de sus interfaces.

A partir de esta fase podríamos ir creando un catálogo de requisitos de manera ordenada, ya que tenemos más o menos decididos cuales serán las diferentes entidades podemos ordenar los requisitos de nuestro catálogo por entidades.

4.2.2.2 Interacción de componentes

En las fases anteriores hemos identificado las operaciones de la interfaz de negocio y los requisitos que las especifican, pero estos requisitos eran generales ya que no sólo conocemos la signatura de la operación. Como en esta fase se definen mejor como serán estas operaciones podremos crear unos requisitos más específicos.

Si tenemos una operación de alquilarPelicula, su entrada podría ser el DNI del usuario y la salida un número que indique si la reserva se ha realizado con éxito o no. Por ejemplo, el requisito correspondiente a los parámetros de entrada (en este caso sólo uno) “Se introducirá el DNI para realizar el alquiler”. Y los correspondientes al parámetro de salida, “La salida será 0 si se realiza con éxito la reserva” y ”La salida será -1 si no se ha podido realizar el alquiler”.

En esta fase también podemos añadir los requisitos sobre la responsabilidad y delegación de cada componente. Si creemos que la responsabilidad de llamar a la operación alquilarPelicula() debería ser de la entidad Vendedor, debemos definir un requisito para dejarlo claro. Podríamos refinar el requisito anterior que hemos definido sobre la salida de esta manera, “El vendedor introducirá el DNI para realizar el alquiler”, de manera que este requisito queda más claro ahora y es más específico. Antes era más general no habíamos decidido de quien era responsabilidad de llamar a la operación.

En nuestro sistema seguramente tendremos operaciones para agregar y eliminar entidades, claramente debemos tener una operación que dé de alta a un usuario nuevo y otra que lo elimine. Estas operaciones también habrán sido reconocidas ya en esta fase. En estas es muy importante hacer una buena asignación de responsabilidad y que quede claramente reflejado en sus requisitos

Ya tenemos definido el conjunto de operaciones del negocio, ahora tenemos que repasar lo que hemos hecho hasta ahora incluido las interfaces y la especificación de

componentes y arquitectura. En este proceso de factorización se deben tener en cuenta los requisitos ya que si eliminamos alguna operación para eliminar dependencias cíclicas, por ejemplo, deberemos modificar los requisitos pertenecientes a las operaciones implicadas para adaptarlos a la nueva situación. Posiblemente en la factorización no se añadan nuevos requisitos si no que se modifiquen o eliminen los ya existentes.

4.2.2.3 Especificación de los componentes

En esta fase se realizaban los contratos de uso y realización. Los requisitos correspondientes al contrato de uso servirán para llegar a un acuerdo sobre la utilización de los componentes del sistema, cuanto más claros estén los requisitos de esta parte mejor uso se harán de los componentes. Los requisitos de los contratos de realización servirán para que los programadores implementen los componentes según estas especificaciones. Si estos requisitos no son bien definidos o no son completos posiblemente no obtengamos el resultado esperado del funcionamiento de estos componentes.

Al término de esta fase cada componente debe tener definido completamente sus operaciones, para que el conjunto de requisitos sea completo deberemos tener requisitos sobre sus parámetros de entrada, parámetros de salida, cambios sobre el estado del componentes, y sus restricciones o invariantes.

Las definiciones de las precondiciones y postcondiciones ya estén escritas en lenguaje humano o en OCL nos ayudarán definir correctamente los requisitos relativos al funcionamiento esperado de la operación.

Con los requisitos hallados a partir del invariante podremos saber cual es la evolución del estado de los datos dados como entrada.

Con todos estos requisitos dejaremos bien definidos lo que se espera de cada operación.

4.2.3 Fase de aprovisionamiento

Los requisitos que aparecen en esta fase dependen mucho del tipo de tecnología que decidamos elegir para desarrollar el componente. Primeramente en esta fase a partir de los requisitos de cada componente que hemos diseñado podemos buscar si existe en el mercado algún componente ya sea de pago o gratis que sus requisitos sean compatibles con los nuestros. Aquí podemos ver la importancia del buen diseño de requisitos realizado, si el diseño ha sido bueno nos será fácil identificar si existe algún componentes compatible con nuestra especificación o si sería posible adaptar algún requisito para que nuestra especificación se adecue a la especificación del requisito. Para ello tendremos que modificar algún requisito de manera que lo generalicemos o tal vez sea necesario eliminar alguno. Por ejemplo, es posible que encontremos un componente que implementa la operación de alquilar una película pero entre sus requisitos para los parámetros de entrada encontramos uno que nos dice “Se introducirá un número de DNI y el nombre del usuario”, nosotros puede ser que no tuviéramos contemplado la posibilidad de añadir como entrada el nombre de usuario pero puede ser que no nos importe añadirla para utilizar este componente.

Después de realizar la búsqueda de componentes para utilizar en el sistema es posible que no encontremos ninguno o no nos interese modificar nuestros requisitos para reutilizar alguno en concreto encontrado. Por lo que deberemos elegir una tecnología a utilizar para la implementación del componente. Dependiendo de que

tecnología utilicemos tendremos que crear unos requisitos u otros. Estos requisitos tendrán que ver con la implementación de la solución, puede ser que se vean afectados otros requisitos de la operación. Si utilizamos un lenguaje orientado a objetos tal vez nos interese que el parámetro de entrada sea un objeto Usuario con sus datos.

Los requisitos no funcionales no se refieren a las funciones específicas que entrega el sistema sino a propiedades como la fiabilidad, respuesta en el tiempo, capacidad de almacenamiento... Todas estas propiedades están muy ligadas al tipo de tecnología que decidamos utilizar para el desarrollo de nuestros componentes, por lo que estos requisitos se crearán en esta fase para ayudar a decidir que tecnología es más preferible utilizar para nuestro problema.

4.2.4 Fase de ensamblaje y pruebas

Los requisitos que se recogen en esta fase no son los del componente en sí sino del sistema en general, ya que es ahora cuando se define la interfaz de usuario y el dialogo lógico de usuario. Por lo que estos requisitos no afectaran a la implementación de cada componente aparentemente.

Para la fase de pruebas podemos definir requisitos para cada componente y a nivel de sistema. Los requisitos de cada componente nos servirán para crear un buen diseño de pruebas del componente y comprobar su perfecto funcionamiento.

4.2.5 Tabla – Resumen

Fase de Requisitos

- Requisitos muy generales de la aplicación.
- Se encuentran principalmente requisitos funcionales.
- Requisitos no funcionales de seguridad y rendimiento del sistema.
- Análisis de Casos de Uso.

Fase de Especificación

Identificación de componentes

- Se eliminan o modifican requisitos para adaptarse al modelo de tipo de negocio.
- Requisitos más detallados.
- Primeros requisitos de las operaciones del sistema.
- Se identifican requisitos de las entidades de la aplicación.
- Requisitos para relacionar unos atributos de una entidad con otros.
- Requisitos de las interfaces de acceso a las entidades.
- Se estructura el catálogo de requisitos según componentes.

Interacción de componentes

- Requisitos más específicos para las operaciones.
- Identificación de requisitos sobre la responsabilidad y delegación de cada componente.
- Factorización de requisitos al término de la fase.

Especificación de componentes

- Se recogen los requisitos de los contratos de uso y realización.
- Requisitos específicos de las operaciones, sobre sus parámetros de entrada, parámetros de salida, cambios en el estado del componente, y sus restricciones o invariantes.
- Utilizamos OCL para escribir requisitos si es necesario.

Fase de aprovisionamiento

- Se refinan los requisitos para adaptarse a la tecnología elegida.
- Los requisitos se utilizan para la selección de componentes reutilizables.
- Adaptamos los requisitos si es necesario de los componentes a reutilizar.
- Los requisitos no funcionales sobre propiedades como la fiabilidad, respuesta en el tiempo, capacidad de almacenamiento…

Fase de ensamblaje

- Requisitos de la interfaz de usuario y diálogo lógico de usuario.

Fase de pruebas

- Identificación de requisitos sobre las pruebas del sistema.

4.3 Valoración del proceso de Chessman y Daniels

A lo largo de este capítulo hemos visto el proceso propuesto por John Daniels y John Chessman para la creación de componentes. En el primer apartado se ha presentado el proceso explicando detalladamente sus fases: fase de requisitos, especificación, aprovisionamiento, ensamblaje y pruebas. En el proceso que siguen John Daniels y John Chessman queda todo muy bien definido y se va siguiendo un proceso de descubrimiento para crear los componentes de la aplicación.

Después, hemos realizado un estudio de cómo los requisitos afectan a todo el proceso. La creación de los requisitos junto con los componentes es algo importante, debido a que estos evolucionan a la vez que el componente se va creando. Estos requisitos, al haber seguido un proceso de creación en paralelo con el componente, nos pueden ayudar en la selección de los componentes adecuados para nuestra aplicación. A lo que se quiere llegar es a una mejor utilización de los componentes y su reutilización, si los componentes y los requisitos han seguido un proceso ordenado de identificación y diseño nos será más fácil manejar el componente a través de sus requisitos y gracias a estos podremos comprar o copiar componentes para reutilizarlos.

Como hemos realizado en nuestro estudio, se comienza por una pequeña especificación y se van encontrando los primeros requisitos más generales a la vez que vamos identificando los componentes, sus operaciones entre ellos... Hasta llegar a la implementación de nuestro sistema basado en componentes con unos requisitos bien definidos. Lo más positivo es que al término del proceso tendremos un catálogo de requisitos para cada componente muy completo.

Un punto negativo que se encuentra en este proceso es que se empiece la fase de pruebas tan tarde, si empezará un poco más al principio o incluso en paralelo con las primeras fases se podrían definir los requisitos necesarios que deberían cumplir las pruebas de los componentes e ir refinándolos desde el principio junto con los demás requisitos.

5. COTSRE+: Hacia un método de desarrollo basado en componentes y requisitos

En la Universidad de Murcia se diseñó un método de selección de componentes llamado COTSRE (*ComponenTs Selection method based on Requirements Engineering*) [2,3], el cual está basado en el método SIREN. Este método se basa en estándares de la disciplina y en el uso de catálogos de requisitos reutilizables. Actualmente este método está definido de forma incompleta y solo para lo que sería la fase de selección de un componente de un determinado repositorio.

COTSRE suministra un método ágil para los desarrolladores del software a la hora de escoger un componente a partir de unos requisitos. Inicialmente se propone que cada requisito del catálogo de requisitos reutilizables contenga un atributo llamado por ejemplo "fuente", que indique el lugar donde se puede encontrar un componente software (o varios) que lo cumplan. De esta manera, los desarrolladores buscarán los requisitos que necesitan dentro del catálogo de requisitos. Una vez tengan estos, mirarán el atributo "fuente" de cada uno en busca de los componentes que los cumplan. Con toda probabilidad no encontraremos sólo uno, sino un conjunto inicial de posibles candidatos. A la hora de seleccionar qué componente se adapta mejor al conjunto de requisitos necesarios juegan un papel muy importante los requisitos no funcionales como pueden ser el lenguaje de programación, el sistema operativo, la memoria disponible, el tiempo de ejecución, tipo de software (libre o propietario)... Estos requisitos imponen fuertes restricciones en la aplicación final a desarrollar por lo que deben utilizarse a la hora de escoger el componente [24].

Para ayudar en el proceso de selección se propone hacer una matriz de selección para tener una visión rápida de la relación entre los requisitos que son necesarios que cumpla la aplicación y los componentes seleccionados.

Por ejemplo, podemos tener una aplicación que hará uso de un servidor de correo electrónico y necesitamos seleccionar un componente que cumpla con unos requisitos prefijados como pueden ser los siguientes:

- Requisito A: *La aplicación deberá usar los protocolos SMTP, POP y MIME para enviar y recibir emails.*
- Requisito B: *La aplicación de email deberá proveer de la creación de firmas digitales.*
- Requisito C: *La aplicación deberá estar implementada usando el paradigma de programación Builder C++ o Microsoft Visual C++.*
- Requisito D: *La aplicación deberá poder ser ejecutada en el sistema operativo Windows XP.*

El atributo "fuente" del requisito A nos indica que hay una serie de componentes que lo cumplen (*Easy Mail Objects v6.5, Easy Mail .Net 3.0, PowerTCP Mail .NET v2.1.4, EMail Wizard Toolpack v3, Catalyst Internet Email v4.5, Rebex Mail .NET v1.0.3793, PowerTCP Mail For ActiveX v 2.9.4.1*), el "fuente" del requisito B tiene a *Email Wizard Toolpack v3* y *Catalyst Internet Email v6.0*, etc. y así sucesivamente vamos mirando el atributo de cada requisito y rellenando la matriz de selección para tener una visión más global. Estos ejemplos los hemos extraído de la web *http://www.componentsource.com*.

Componente	Requisito A	Requisito B	Requisito C	Requisito D
Easy Mail Objects v6.5	X		X	
Easy Mail .Net v3.0	X			X

PowerTCP Mail .NET v3.1	X		X	X
EMail Wizard Toolpack v3	X	X	X	X
Catalyst Internet Email v6.0	X	X	X	X
Rebex Mail .NET v1.0.3793	X		X	X
PowerTCP Mail For ActiveX v 2.9.4.1	X			X

Tabla 2. Matriz de Selección

El analista visualiza en la matriz de selección el componente o componentes que cumple todos los requisitos deseados. En este caso se produce un empate entre los componentes *Email Wizard Toolpack v3* y *Catalyst Internet Email v6.0*. Por lo que debemos desempatar de alguna forma realizando una matriz de decisión. En este ejemplo, se ha producido un empate pero podría ser que sólo hubiéramos encontrado un componente candidato con lo que este componente automáticamente se hubiera convertido en el candidato a seleccionar y el método COTSRE terminaría. Pero también podría darse el caso de que ningún componente cumpliera con todos los requisitos, en este caso el cliente podría descartar uno o varios requisitos o continuar el método COTSRE con los componentes que más cumplan requisitos para desempatar.

Siguiendo el ejemplo, realizamos la matriz de decisión a partir de los componentes finalistas en donde los enfrentamos con las características que son deseables para esos componentes. Por ejemplo su coste, usabilidad, calidad de su documentación, confianza en el proveedor y así como otras consideraciones que el analista crea que son deseables en un componente para su elección. Podemos establecer 3 valores posibles para medir el grado de satisfacción en escala de mayor a menor estos valores serían *altamente deseable* (3), *deseable* (2) y *opcional* (1). Cada componente tendrá asociado por cada característica que indica su grado de cumplimiento, los valores son *incumplido* (-1), *cumplimiento favorable* (1) y *cumplimiento altamente favorable* (2).

Características	**eMail Wizard Toolpack v3.0**	**Catalyst Internet Mail v6.0**
Coste	1	2
Confianza en el vendedor	2	1
Usabilidad	2	2
Calidad de Documentación	1	2

Tabla 3. Características de los componentes

Podemos utilizar para la selección del componente la técnica WSM (*Weighted Scoring Method*), método de puntuación ponderado [40]. Este método consiste en dar unas puntuaciones a los requisitos y unos pesos a los componentes. De manera que para obtener la puntuación de un componente hacemos $c1 = w_1s_1+w_2s_2+...+w_ns_n$

Componentes	**Puntuación**
eMail Wizard Toolpack v3.0	1·3+2·2+2·3+1·2=15
Catalyst Internet Mail v6.0	2·3+1·2+2·3+2·2=18

Tabla 4. Matriz de Decisión

El que más puntuación nos da es *Catalyst Internet Mail v6.0*, con lo que finalmente este es el componente que cumple mejor con los requisitos y características deseables. Gracias a COTSRE hemos encontrado el componente que mejor se adapta a nuestros objetivos de manera rápida y sencilla, analizando las posibles soluciones y ahorrándonos tiempo de desarrollo de un componente desde el principio.

5.1 Limitaciones de COTSRE

Como vemos gracias a este método encontramos de manera sencilla el componente más adecuado a una serie de requisitos introducidos por el usuario. Pero este método está definido de una forma muy inicial. Existen casos que no tiene en cuenta, como por ejemplo, no se contempla la posibilidad de que no existan componentes que cumplan con todos los requisitos introducidos y que la selección deba hacerse a partir de un conjunto de componentes que cumplan con un tanto por ciento de los requisitos.

Un punto interesante sería que el método COTSRE seleccionara también componentes a partir de un caso de uso. Si solo introducimos requisitos para la selección puede ser que nos ofrezca como candidato un componente que no cumpla con el caso de uso que se ha diseñado para él. De esta manera la búsqueda sería más completa y con más posibilidades de que el componente seleccionado sea el más adecuado.

En este proyecto se propone ampliar el método de selección introduciendo casos de uso y definir que ocurriría en casos, como el que hemos presentado anteriormente, que la selección se haga entre componentes que no cumplen con todos los parámetros de búsqueda (tanto requisitos y/o caso de uso) introducidos.

Además de lo dicho anteriormente, queremos que COTSRE se convierta en un método de desarrollo basado en componentes y requisitos que abarque desde la recogida de requisitos hasta la puesta en práctica del componente y en el que la reutilización sea una parte importante del proceso. Este método se va a basar en el proceso de desarrollo de Chessman y Daniels explicado en este documento y que llamaremos COTSRE+. Se ha propuesto este proceso porque es sencillo y bien estructurado en fases, además vamos a utilizar el estándar de SPEM, para describir COTSRE+.

5.2 SPEM

SPEM, *Software Process Engineering Metamodel*, [28] es un estándar de OMG cuyo principal objetivo es el de proporcionar un marco formal para la definición de procesos de desarrollo de sistemas y software así como todos los elementos que lo componen. La última versión de este estándar es la 2.0. Se ha decidido seguir este estándar para especificar nuestro método COTSRE+. En este apartado haremos un pequeño resumen de él.

SPEM se centra en 3 elementos básicos: rol, producto de trabajo y tarea.

- Tarea: Representan el esfuerzo a hacer.
- Rol: Representan quien lo hace.
- Producto de trabajo: Representan las entradas que se utilizan en las tareas y las salidas que se producen.

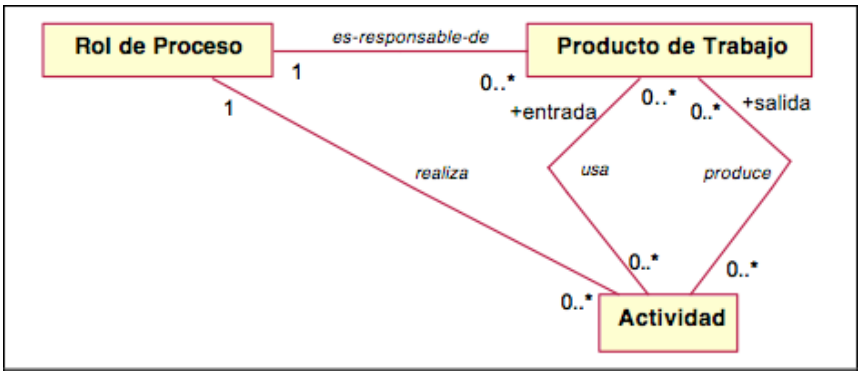


Figura 12. Elementos básicos de SPEM

El metamodelo de SPEM se divide en 7 paquetes y se relacionan como se ve en la siguiente figura. Además de un metamodelo para ingeniería de procesos es un marco de trabajo conceptual que provee de los conceptos necesarios para modelar, documentar, presentar, publicar, gestionar intercambiar y realizar métodos y procesos software. Está orientado a ingenieros de procesos, jefes de proyectos, gestores de proyectos… que son los responsables de mantener e implementar procesos para sus organizaciones o para proyectos concretos.

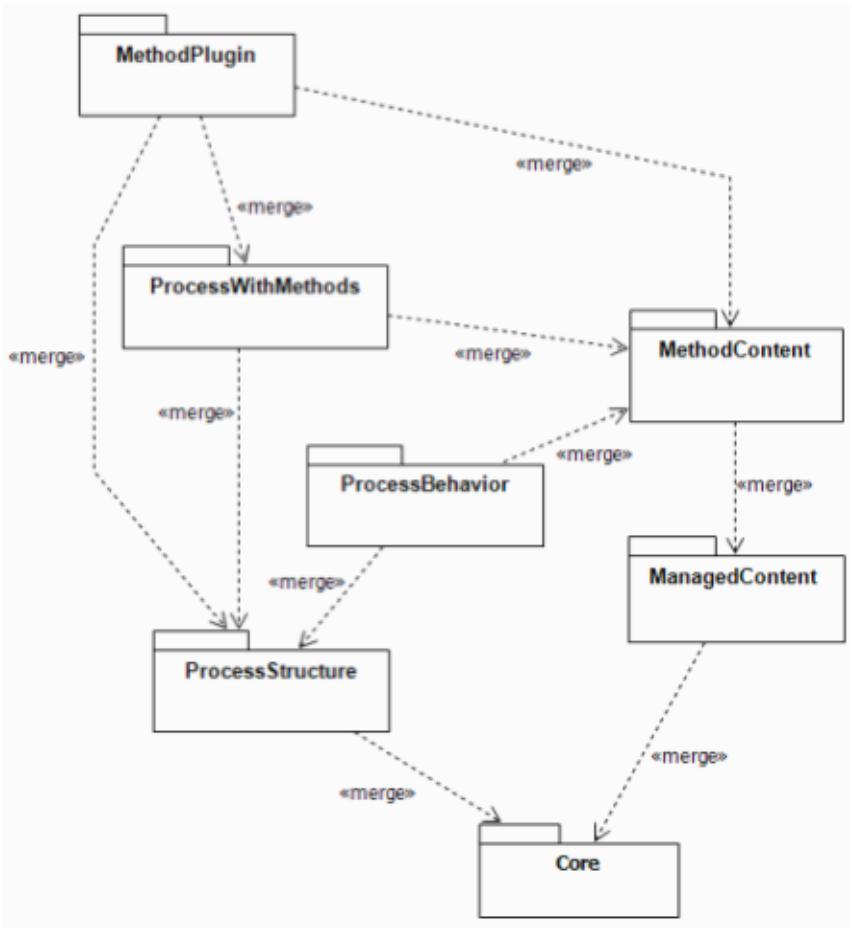


Figura 13. Estructura de paquetes de SPEM

A continuación pasamos a detallar cada paquete.

- *Core*: Contiene todas las clases y abstracciones que constituyen la base para el resto de paquetes. Este paquete define clases para dos capacidades, una es para

crear cualificaciones definidas por el usuario que permiten establecer tipos diferentes entre las instancias de una clase y una colección de clases abstractas para definir el trabajo.

- *Process Structure*: Este paquete define la base para todos los modelos de procesos. Define la estructura de desglose de trabajo mediante el anidamiento de actividades y dependencias de precedencia entre ellas. Esta estructura también incluye referencias a la lista de roles que realizan cada actividad y a los productos de trabajo.
- *Process Behaviour*: Este paquete representa el comportamiento de los procesos como diagramas de actividad, máquinas de estado…
- *Managed Content*: Permite incorporar y gestionar descripciones en lenguaje natural, documentos y otras informaciones útiles. Existe cierta información que no puede ser formalizada mediante modelos así que este paquete se ofrece para ello.
- *Method Content*: Incluye los conceptos para crear elementos de método, esto es, alguien (rol) hace algo (tarea) para obtener algo (producto de trabajo) basándose en algo (guía). Estos elementos permiten describir como se alcanzan los objetivos del proceso haciendo qué tareas, por qué roles, usando qué recursos y obteniendo qué resultados.
- *Process with Methods*: Contiene los elementos necesarios para integrar los conceptos del paquete *Process Structure* con los conceptos y elementos del paquete *Content Method*.
- *Method Plugin*: Este paquete incluye los conceptos para diseñar, gestionar y mantener repositorios y librerías de *Method Content* y *Process with Methods* que sean mantenibles a gran escala, reutilizables y configurables.

Con SPEM se establece una clara separación entre los elementos utilizados para definición formal de un método y los elementos utilizados para describir la aplicación de dicho método a un proceso dentro de un proyecto concreto.

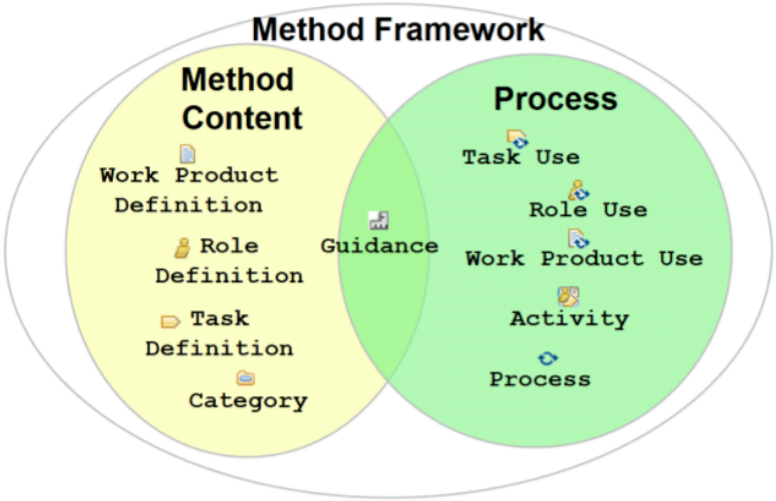


Figura 14. Estructura de Method Framework

5.3 Técnica de clasificación y recuperación

Una técnica de clasificación y recuperación de calidad debe cumplir con dos criterios importantes, precisión y recuperación [23]. Deseamos que los componentes encontrados sean lo más parecido a los componentes que buscamos y que la cantidad de

componentes candidatos no sea ni muy pequeña ni muy grande. Si el conjunto total es pequeño puede ser difícil decidir que componente se elige en caso de que se requiera una adaptación. Si el conjunto es muy grande la elección puede llegar a ser intratable.

Por eso en COTSRE+ utilizamos las especificaciones formales. Por un lado en el proceso de desarrollo de un componente se va revisando su especificación formal de manera que a la finalización del proceso esta es muy completa y se añadirá al repositorio esta información junto al componente. Así será más fácil de clasificar. Además cuando un usuario desee realizar una consulta al repositorio si añade los requisitos que desea que se cumpla encontrará una lista de candidatos que cumplen exactamente con sus requerimientos. De modo que el ciclo de desarrollo de un componente desde su inicio hasta su posible reutilización queda bien definido.

Pero podemos tener el problema de que el repositorio sea demasiado amplio por lo que además se podría utilizar una indización automática. Los componentes se encontrarían ordenados por diferentes catálogos según el área a la que pertenecen y a su vez se podría hacer una ordenación según el tipo de requisitos que cumplen estos componentes.

5.4 Especificación del proceso COTSRE+ con SPEM

Partiendo del proceso de Chessman y Daniels, hemos completado el método de selección de componentes COTSRE para convertirlo en un método de desarrollo basado en componentes y requisitos bien definidos utilizando la notación de SPEM y llamándolo ahora COTSRE+. Básicamente se respetan las fases de Chessman y Daniels pero desarrollamos más la fase de aprovisionamiento, especificando como sería la búsqueda de componentes reutilizables y aportando una aplicación software que automatice este proceso. La búsqueda de componentes que inicialmente proponía COTSRE ha sido ampliada para que incluya búsquedas por casos de uso además de por requisitos. Las fases de ensamblaje y despliegue de Chessman y Daniels se han fusionado ya que creemos que las actividades que realizan están muy relacionadas entre sí. La fase de pruebas se ha eliminado realizando las pruebas individuales de cada componente en la fase de aprovisionamiento y las pruebas de la aplicación desarrollada en la fase de ensamblaje y despliegue. Otro cambio introducido ha sido en la finalización de cada fase, hemos añadido una actividad de documentación para guardar un histórico del proceso de desarrollo. En la primera fase, la de requisitos, el documento generado se utilizará como contrato con los clientes. De esta forma, al finalizar el desarrollo de una aplicación utilizando COTSRE+, se tenga una documentación completa del mismo.

A continuación se muestra un diagrama de fases del proceso de desarrollo basado en componentes que proponemos para COTSRE, realizado con la notación de SPEM.

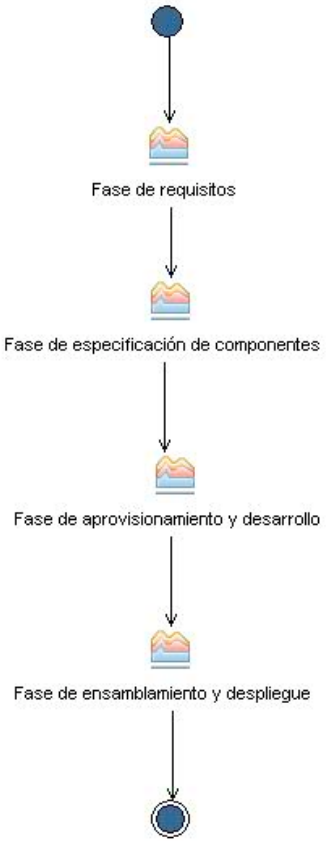


Figura 15. Diagrama de fases SPEM

Los principales actores o roles que interactúan con COTSRE+ los hemos dividido en dos grupos, los pertenecientes a la parte del cliente y los de la parte técnica. Esta ha sido una de las principales aportaciones de COTSRE+ ya que Chessman y Daniels no definen bien cuales son los roles existentes a lo largo de su proceso.

Los roles del cliente son:

- *Usuarios finales*: Son los usuarios de la aplicación acabada. Ellos aportan información general sobre que es lo que esperan del sistema y como quieren interactuar con él.
- *Expertos en el dominio*: Son expertos en el dominio de la aplicación, ofrece información sobre las particularidades del ámbito del sistema.

Los roles técnicos son:

- *Ingeniero de requisitos*: Es el encargado de recoger la información aportada por los usuarios finales y los expertos en el dominio para crear los requisitos de la aplicación que se va a desarrollar.
- *Analista:* Persona encargada de realizar la especificación de los componentes de la aplicación.
- *Seleccionador*: Se encarga de seleccionar un componente de un catálogo a partir de unos requisitos y un caso de uso.
- *Testeador*: Encargado de diseñar y realizar pruebas a los componentes de manera individual y a la aplicación en conjunto.
- *Programador*: Se encarga de implementar los componentes y la aplicación según unas especificaciones dadas.
- *Documentalista*: Se encarga de realizar la documentación al final de cada fase.

Todas las fases han sido divididas en actividades y éstas a su vez pueden estar divididas en una o varias tareas. En los siguientes subapartados pasamos a explicar en detalle cada una de ellas. Marcaremos con un "+" aquellas actividades, tareas o productos de trabajo que sean una aportación o mejora de COTSRE+ al proceso de Chessman y Daniels.

5.4.1 Fase de requisitos

El objetivo de esta fase es obtener toda la información posible de los usuarios finales y expertos del dominio y detectar cuales son los requisitos y casos de uso del sistema. Esta fase es muy importante ya que a partir de toda esta información se podrá comenzar el proceso de análisis de los componentes de la aplicación.

Los productos de trabajo definidos en esta fase son:

- *Visión del sistema*: Es una descripción de unas pocas líneas que nos ofrecen los usuarios del sistema. Se describen los límites del sistema y lo que se espera de la aplicación final.
- *Requisitos del negocio*: Son los requisitos no funcionales que debe cumplir la aplicación.
- *Modelo conceptual del negocio*: Es un mapa de relaciones entre las entidades más importantes del sistema identificadas. Este no necesita ser detallado aunque en algunas entidades pueden aparecer atributos.
- *Requisitos de seguridad*: Pertenecen a los requisitos de calidad de servicio hacen referencia a las medidas de seguridad que debe adoptar el sistema.
- *Requisitos de rendimiento*: Pertenecen a los requisitos de calidad de servicio hacen referencia al rendimiento esperado del sistema.
- *Información del cliente*: Los usuarios finales y los expertos en el dominio ofrecen información sobre que es lo que se desea implementar y cuales son sus necesidades.
- *Proceso de negocio*: Es un diagrama en el que se muestra de forma esquemática cuales son los procesos que forman parte de la aplicación. La notación utilizada es un diagrama de actividad en UML.
- *Proceso de negocio con responsabilidades*: Es el proceso de negocio refinado en el que se han especificado de quien es la responsabilidad de ejecutar cada proceso.
- *Plantilla de documento del proyecto(+)*: Es una plantilla ofrecida por COTSRE+. Esta plantilla está dividida en apartados para tener documentado el proceso de desarrollo de la aplicación. Cada apartado se corresponde con una fase del proceso, y dentro de cada uno aparecerán los productos de trabajo creados y que son relevantes para la siguiente fase.
- *Documentación del proyecto provisional(+)*: Es la plantilla del documento de proyecto rellenada pero sin aprobar por los clientes.
- *Documentación del proyecto aprobada(+)*: Es el documento del proyecto aprobado por los clientes.
- *Plantilla de caso de uso(+)*: Es un documento ofrecido por el método COTSRE+ para redactar un caso de uso, podemos verla en el anexo II. Contiene:
  - Nombre: El nombre del caso de uso que se describe.
  - Iniciador: Actor que inicia la acción.
  - Objetivo: Objetivo del caso de uso.

    - Sucesos del escenario principal: Es una guía de los pasos que sigue la acción. Irán numerados para una mejor comprensión.
    - Extensiones: Son pasos alternativos que pueden ocurrir debido a un fallo o una excepción, son muy importantes para poder tener una descripción completa de la acción y saber como solucionar posibles errores.
- *Actores*: Los actores son una entidad externa al sistema que guarda una relación con este y le demandan una funcionalidad. Además de incluir a las personas físicas pueden incluir a sistemas externos tales como el tiempo.
- *Diagrama de casos de uso*: Un diagrama de casos de uso nos sirve para resumir como interactúan los actores con el sistema. En él aparecerán todos los casos de uso, los actores y sus relaciones (inclusión, extensión o generalización).
- *Descripción de los casos de uso*: Es una copia de la plantilla de casos de uso rellenada en la que se describe la funcionalidad de un caso de uso.
- *Entidades*: Las entidades son los objetos principales del modelo de negocio del sistema.

Hemos estructurado esta fase en cinco actividades las cuales a su vez se dividen en tareas. Estas actividades pueden verse en el siguiente diagrama y que a continuación detallamos.

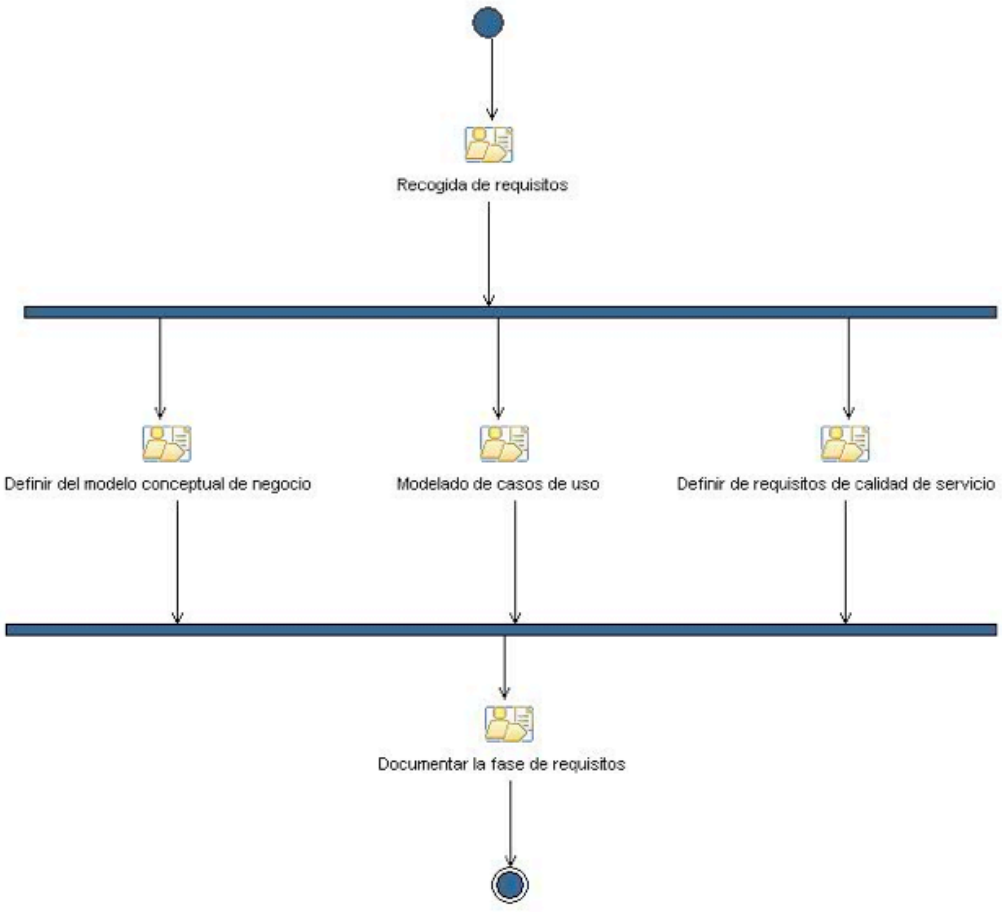


Figura 16. Diagrama actividad de la fase de requisitos

- **Actividad 1.1: Recogida de requisitos**

Se comienza realizando una serie de entrevistas con los usuarios finales y los expertos en el dominio para captar cuales son los requisitos de la aplicación a generar y qué es exactamente lo que necesitan. Se podrán realizar una o más entrevistas, tantas como sean necesarias hasta entender bien que es lo que se va a desarrollar. El propósito

de estas entrevistas es tener una visión global del sistema, definir bien sus límites, escenarios, cuales son los usuarios, su funcionamiento… Tener una base bien definida de lo que se quiere desarrollar. Crearemos un proceso de negocio en alguna notación que sea fácilmente entendible por los clientes como podría ser UML.

El cliente nos puede hacer un pequeño resumen de lo que es su visión del sistema, que explique en unas pocas líneas en lenguaje natural que es lo que necesita, cual es su funcionamiento esperado. A partir de esto se pueden identificar los límites del sistema, que es trabajo del software y que no, cómo trabaja el usuario con el sistema, qué representa para él. Con esto actualizamos el proceso de negocio con las responsabilidades de cada parte. Aunque esta decisión de quien es responsable de qué se haga muy rápido, permite tener una base de la que partir para luego ir refinándola conforme conozcamos más el sistema.

Las tareas de esta actividad son:

- *Entrevistas con los usuarios*: Se realizan las entrevistas entre el ingeniero de requisitos y los usuarios finales y expertos en el dominio.
  - Entrada: Información del cliente.
  - Salida: Requisitos de negocio, visión del sistema, proceso de negocio.
- *Definir límites del sistema*: El ingeniero de requisitos decide que es responsabilidad del sistema y de los usuarios finales, para tener claro que es lo que se debe desarrollar y se desarrolla un proceso de negocio.
  - Entrada: Requisitos negocio, visión del sistema, proceso de negocio.
  - Salida: Proceso de negocio con responsabilidades

- **Actividad 1.2: Definir del modelo conceptual de negocio**

Recogida toda la información se desarrolla el modelo conceptual del negocio. Para ello en esta actividad el ingeniero de requisitos identifica cuales son las entidades a modelar y una vez hecho esto vemos que relaciones existen entre ellas. Este diagrama no necesita estar detallado, podemos añadir algún atributo a las entidades pero esto no es necesario, lo que se desea conseguir son las entidades básicas que maneja la aplicación.

Las tareas de esta actividad son:

- *Identificar entidades del sistema*: El ingeniero de requisitos identifica cuales son las entidades principales del sistema que va a modelar. Es el primer paso para crear un modelo de concepto del negocio.
  - Entrada: Requisitos del negocio, Proceso de negocio con responsabilidades.
  - Salida: Entidades.
- *Definir el modelo conceptual del negocio*: El ingeniero de requisitos a partir de la entidades encontradas, los requisitos de negocio y el proceso de negocio deduce las relaciones que existen entre entidades y crea un modelo conceptual del negocio. Opcionalmente puede añadir a las entidades algún atributo detectado.
  - Entrada: Entidades, Requisitos del negocio, Proceso de negocio con responsabilidades.
  - Salida: Modelo Conceptual del negocio.

- **Actividad 1.3: Modelado de casos de uso**

Se describen los casos de uso que son una buena práctica para identificar los requisitos funcionales del sistema. Primero se comienza identificando quienes son los

actores del sistema, que rol juegan en él y cuales son los escenarios. Con los requisitos recogidos del cliente y el proceso de negocio con responsabilidades podemos ya identificar cuales son los casos de uso de la aplicación y se dibuja un diagrama de casos de uso donde se vea que actores interactúan con estos casos. Para hacer la descripción de cada caso de uso se ponen a disposición de los ingenieros de requisitos unas plantillas para rellenar.

Las tareas de esta actividad son las siguientes:

- *Identificar actores*: En esta tarea el ingeniero de requisitos identifica cuales son los actores principales de la aplicación.
  - Entrada: Modelo conceptual del negocio, Proceso de negocio con responsabilidades.
  - Salida: Actores.
- *Identificar casos de uso*: Una vez encontrados los actores, el ingeniero de requisitos identifica cuales son los objetivos del usuario y cuales son las interacciones con el sistema. Realiza un diagrama de casos de uso en UML con los actores y los casos de uso identificados. Se pueden crear relaciones entre los casos de uso identificados, hay de dos tipos extend y uses. *Extend*: El caso de uso que extiende realiza una acción en un punto del caso de uso extendido, si se cumple una condición (en UML se denota <<extend>>). *Uses*: Se factorizan acciones que se utilizan en más de un caso de uso (en UML se denota como <<include>>).
  - Entrada: Actores del sistema, Proceso de negocio con responsabilidades.
  - Salida: Diagrama de casos de uso.
- *Describir casos de uso*: A partir de la plantilla dada por el método el ingeniero de requisitos rellena la descripción de cada caso de uso.
  - Entrada: Diagrama de casos de uso, Proceso de negocio con responsabilidades, Requisitos del negocio, Plantillas de casos de uso, Modelo de concepto de negocio.
  - Salida: Descripción de los casos de uso.

- **Actividad 1.4: Definir requisitos de calidad de servicio**

Se deben tener también en cuenta los requisitos de calidad de servicio, son muy importantes los referentes a la seguridad y el rendimiento, y necesarios para crear una aplicación de calidad que satisfaga al cliente.

La actividad se divide en dos tareas, una para definir los requisitos de seguridad y la otra para los requisitos de rendimiento.

- Definir de requisitos de seguridad: El ingeniero de requisitos define cuales son los requisitos de seguridad que debe cumplir la aplicación.
  - Entrada: Visión del sistema, Información del cliente.
  - Salida: Requisitos de seguridad.
- Definir de requisitos de rendimiento: El ingeniero de requisitos define cuales son los requisitos de rendimiento que debe cumplir la aplicación.
  - Entrada: Visión del sistema, Información del cliente.
  - Salida: Requisitos de rendimiento.

- **Actividad 1.5: Documentar la fase de requisitos (+)**

En la última actividad se crea un documento donde guardar todo el análisis de la aplicación que ha empezado a hacerse en esta fase y que servirá para tener una buena documentación del desarrollo. Además este documento en esta fase actuará de contrato

con los clientes, una vez aprobado se comenzará con las siguientes fases del proceso. Se proporciona una plantilla del esqueleto donde en el apartado correspondiente a la fase de requisitos se incluirán los requisitos (de negocio, rendimiento, seguridad…), el modelo de concepto, el proceso de negocio, y la información de los casos de uso. En esta plantilla además incluiremos información en lenguaje natural sobre la visión del sistema y una pequeña introducción sobre lo que se va a realizar.

La actividad se divide en tres tareas:

- *Crear un documento estándar para el desarrollo de la aplicación*: El documentalista hace una copia de la plantilla de documentos de proyecto. Se añade la fecha de creación y una introducción de la descripción del sistema.
  - Entrada: Plantilla del documento del proyecto.
  - Salidas: Documentación del proyecto provisional.
- *Rellenar la plantilla*: El documentalista rellena la plantilla con todos los datos recogidos en esta fase.
  - Entradas: Requisitos de negocio, Requisitos de seguridad, Requisitos de rendimiento, Diagrama de casos de uso, Descripción de los casos de uso, Modelo de concepto del negocio, Proceso de negocio con responsabilidades.
  - Salidas: Documentación del proyecto provisional.
- *Aprobación del documento*: Este documento se presenta a los clientes y si están de acuerdo se firma por ambas partes. En caso contrario se revisa.
  - Entradas: Documentación del proyecto provisional.
  - Salidas: Documentación del proyecto aprobada.

5.4.2 Fase de especificación de componentes

Al igual que en el proceso de Chessman y Daniels la fase de especificación de componentes la dividimos en tres subfases, de identificación, interacción y especificación de componentes.

5.4.2.1 Fase de identificación de componentes

En esta fase se crea un conjunto inicial de interfaces y especificaciones de componente, conectadas entre sí a una primera aproximación de la arquitectura de componente. Lo que se pretende averiguar es que información necesitamos manejar, que interfaces la manejan, que componentes necesitamos para proveer que funcionalidad y como va a encajar todo esto.

Los productos de trabajo creados en esta fase son:

- *Interfaces de sistema:* Conjunto de operaciones de los componentes del sistema.
- *Interfaces de negocio*: Conjunto de operaciones de los componentes del negocio.
- *Arquitectura inicial de componentes:* Es un modelo que une las especificaciones de los componentes del sistema y del negocio.
- *Modelo de tipo de negocio:* Es una copia exacta del modelo conceptual del negocio pero más refinado y que añade atributos y reglas de negocio.
- *Tipos principales del negocio:* Son entidades del negocio que tienen existencia independiente dentro del negocio.
- *Especificación de los componentes del sistema*: Está formado por los componentes del sistema y sus interfaces, incluye además las interfaces con los componentes de negocio e interfaces con otros componentes de sistema.

- *Especificación de los componentes del negocio*: Está formado por los componentes del negocio y sus interfaces propias.
- *Reglas de negocio:* Definen las asociaciones entre tipos del modelo de tipo de negocio. Pueden estar escritas en lenguaje natural o formal como OCL.
- *Diagrama de responsabilidad de interfaz del modelo de tipo de negocio:* Es un diagrama que muestra las interfaces de negocio, tipos principales y responsabilidades de las asociaciones en el modelo de tipo de negocio.

Esta fase la hemos dividido en cuatro actividades según puede verse en el siguiente diagrama.

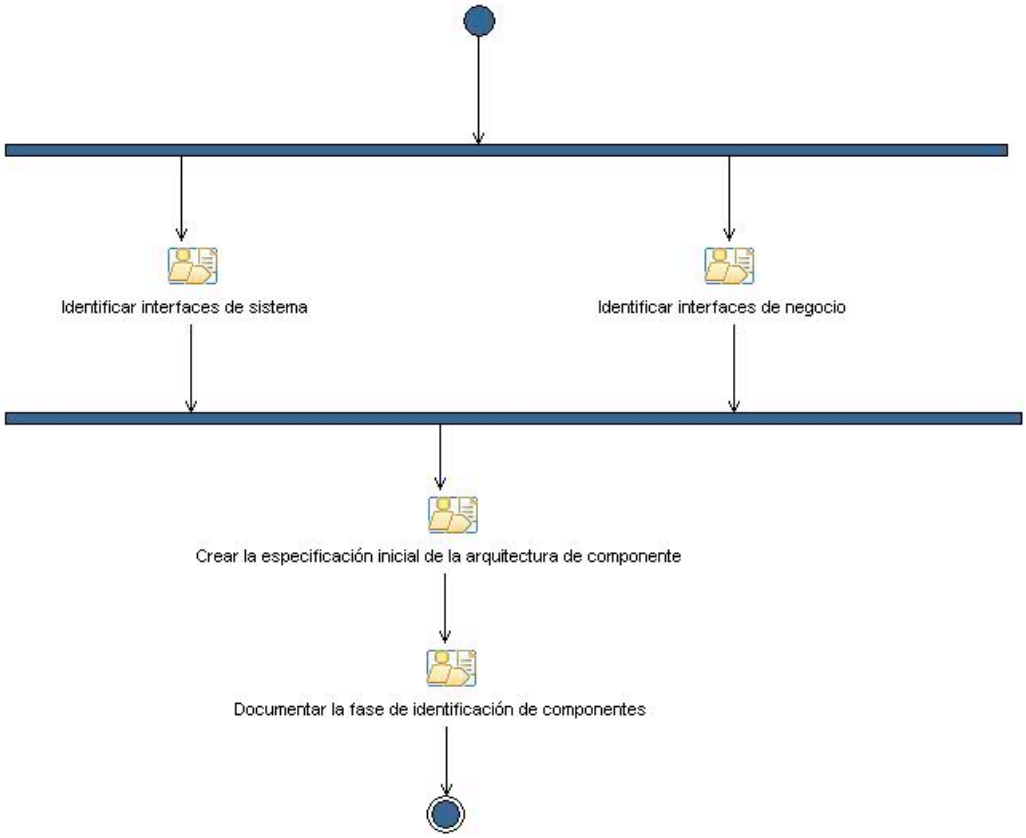


Figura 17. Diagrama de actividad de fase de identificación de componentes

- **Actividad 2.1.1: Identificar interfaces del sistema**

Para cada caso de uso el analista crea una interface de sistema, si el caso de uso es muy complejo deberá crear más. Cada paso de ejecución del caso de uso podría modelarse como una operación de la interfaz. En esta actividad solo se identifica el nombre de la operación.

Crear las interfaces de sistema es una actividad muy sencilla por lo que sólo se ha creado una única tarea.

- *Crear interface de sistema*
  - Entrada: Descripción de los casos de uso.
  - Salida: Interfaces de sistema.

- **Actividad 2.1.2: Identificar interfaces de negocio**

La identificación de interfaces de negocio es un poco más compleja que la de sistemas por lo que son necesarias varias tareas. Las interfaces de negocio se crean a partir del modelo de concepto de negocio y a partir de él se crea un modelo de tipo de negocio en el que se van identificando las interfaces y las responsabilidades de cada una.

- *Crear el modelo de tipo de negocio*: El analista crea una copia del modelo conceptual del negocio. Este modelo es representado por un diagrama de clase UML que contendrá información más específica que el anterior. Este modelo se refina añadiendo y eliminando elementos, de esta manera el modelo es mejorado con cualquier detalle que haya sido omitido anteriormente, en particular los detalles de los atributos en cada tipo, definiendo un conjunto de tipos de datos para usar en este modelo y definiendo las restricciones como asociaciones múltiples.
  - Entrada: Modelo conceptual del negocio.
  - Salida: Modelo de tipo de negocio.
- *Definir las reglas de negocio*: El analista añade al modelo cualquier regla de negocio que se requiera, escribiendo restricciones y añadiendo nuevos atributos. Las restricciones pueden ser escritas utilizando el lenguaje natural, pero también podemos utilizar lenguajes como OCL para hacer una restricción en un lenguaje formal escribiéndolas entre llaves {}.
  - Entrada: Modelo de tipo negocio.
  - Salida: Modelo de tipo negocio, reglas de negocio.
- *Crear las interfaces del negocio y asignar responsabilidades*: El analista identifica que tipos se consideran los principales. El propósito de identificarlos es tener conocimiento de que información será dependiente de que otra. Es un paso necesario para luego poder designar responsabilidades para las interfaces. Un tipo principal es aquel que tiene existencia independiente del sistema. Para identificar los tipos principales les damos el esterotipo <<core>> de UML. Todos los otros tipos proveen los detalles de los tipos principales. El analista crea una interfaz de negocio por cada tipo principal en el modelo de tipo de negocio. Cada interfaz de negocio maneja la información representada por el tipo principal y sus tipos detallados. Estas interfaces se llaman IxxxMgt, donde xxxx representa el nombre del tipo principal que manejan. Para las asociaciones se intenta reducir las dependencias. Cuando existe una relación entre dos tipos manejados por diferentes interfaces se produce una asociación inter-interface, y en estos casos debemos tomar la decisión de quien será el que guarde la información para que se produzca la mínima dependencia. Se asigna direcciones de referencia a las asociaciones para definir de forma más precisa que información tiene que mantener cada interfaz.
  - Entrada: Modelo de tipo de negocio.
  - Salida: Interfaces de negocio, Diagrama de responsabilidad de interfaz del modelo de tipo de negocio.

- **Actividad 2.1.3: Crear la especificación inicial de la arquitectura del componente:**

Se crea una arquitectura de componentes inicial, esta arquitectura está formada por los componentes e interfaces de sistema y negocio y las relaciones entre sus interfaces. Esta arquitectura se refinará en las siguientes fases.

Se ha divido en tres tareas, dos para hallar las especificaciones de sistema y negocio y por último una para hallar la arquitectura inicial.

- *Crear especificaciones de los componentes del sistema*: El analista crea un componente para cada interface de sistema y le añade interfaces a otros componentes de sistema y a los componentes de negocio que creamos que vamos a necesitar.

        - Entrada: Interfaces del sistema.
        - Salida: Especificación de componentes del sistema.
    - *Crear especificaciones de los componentes del negocio*: Los componentes son los tipos principales del modelo de tipo de negocio y a estos se les añaden las interfaces identificadas.
        - Entrada: Diagrama de responsabilidad de interfaz del modelo de tipo de negocio.
        - Salida: Especificación de los componentes del negocio.
    - *Crear arquitectura inicial*: El analista une las especificaciones de componentes de negocio y sistema a través de sus interfaces comunes creando una arquitectura inicial de componentes.
        - Entrada: Especificación de componentes del sistema, especificación de los componentes del negocio.
        - Salida: Arquitectura inicial.

- **Actividad 2.1.4: Documentar la fase de identificación de componentes (+)**

Por último, el documentalista añade la documentación del proyecto las interfaces de sistema y negocio, la arquitectura inicial de componentes y el diagrama de responsabilidad de interfaz del modelo de tipo de negocio, en el apartado correspondiente a la fase de identificación de componentes.

- *Insertar datos recogidos*
    - Entrada: interfaces del sistema, interfaces del negocio, arquitectura inicial de componentes, diagrama de responsabilidad de interfaz del modelo de tipo de negocio.
    - Salida: Documentación del proyecto.

5.4.2.2 Fase de interacción de componentes

En esta fase se decide como van a interactuar los componentes identificados en la fase anterior. Usamos diagramas de interacción para definir las interacciones que tienen lugar dentro del sistema, para refinar las definiciones de las interfaces existentes, para identificar cómo son usadas las interfaces y para descubrir nuevas interfaces y operaciones.

Los productos de trabajo que se crean en esta fase son:

- *Estructura de datos*: Estructura de datos con información de las entidades de negocio.
- *Operaciones de negocio*: Son las operaciones de las interfaces de negocio.
- *Diagramas de interacción*: Son diagramas de interacción entre interfaces en el que se muestra la secuencia de llamadas de una operación.
- *Arquitectura de componentes*: Es un refinamiento de la arquitectura de componentes inicial con restricciones entre interfaces.

Esta subfase se ha dividido en 3 actividades como puede verse en la siguiente figura.

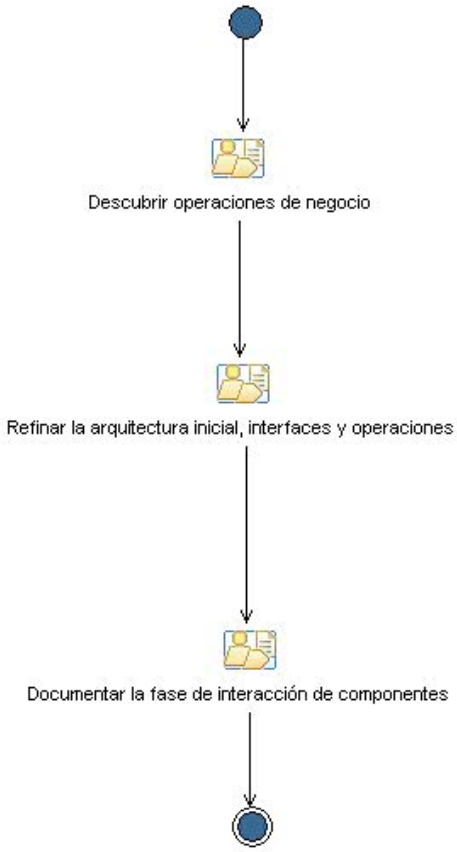


Figura 18. Diagrama de actividad fase de interacción de componentes

- **Actividad 2.2.1: Descubrir operaciones de negocio**

El analista descubre las operaciones de negocio a partir de las operaciones de sistema generando diagramas de interacción.

  - *Crear estructuras de datos:* El analista crea las estructuras de datos necesarias para las operaciones
    - Entrada: Arquitectura inicial, interfaces del sistema, modelo de casos de uso.
    - Salida: Estructura de datos.
  - *Hallar signatura de las operaciones de negocio*: El analista escribe la signatura de la operación (nombre, estructuras de datos y si devuelve algún valor).
    - Entrada: Estructura de datos, arquitectura inicial.
    - Salida: Operación de negocio.
  - *Crear diagrama de interacción para cada operación de sistema:* El analista crea un diagrama de interacción y añade la operación a la interfaz de negocio correspondiente.
    - Entradas: Arquitectura inicial, interfaces de sistema, operaciones de negocio.
    - Salidas: Diagrama de interacción, interfaces de negocio.

- **Actividad 2.2.2: Refinar la arquitectura inicial, interfaces y operaciones**

Por último, el analista refina la arquitectura inicial, las interfaces y sus operaciones.

  - *Añadir restricciones a la arquitectura de componentes*: El analista añade restricciones entre interfaces.

        - Entrada: Arquitectura inicial de componentes.
        - Salida: Arquitectura de componentes.
    - *Definir responsabilidades*: El analista define quién se encarga de borrar, crear, modificar… Existen 5 formas de hacerlo:
        1. Responsabilidad del componente que hace la llamada.
        2. Responsabilidad del componente al que se llama.
        3. Responsabilidad de un tercer componente más general.
        4. Permitir referencias inválidas.
        5. No permitir modificar/borrar datos.
        - Entrada: Arquitectura de componentes.
        - Salida: Diagrama de interacción.
    - *Añadir nuevas operaciones de negocio y sistemas*: Al definir las responsabilidades se descubren nuevas operaciones que deben añadirse a las interfaces correspondientes
        - Entrada: Diagrama de interacción.
        - Salida: Interfaces de sistema, Interfaces de negocio.
    - *Refinar operaciones e interfaces*: Se refinan y factorizan las operaciones y las interfaces, se normalizan, minimizan las llamadas, dependencias, se usan patrones de diseño…
        - Entrada: Interfaces de negocio, Interfaces de sistema.
        - Salidas: Interfaces de negocio, Interfaces de sistema.

- **Actividad 2.2.3: Documentar la fase de interacción de componentes (+)**

En la última actividad, el documentalista añade a la documentación del proyecto la arquitectura de componentes, y negocio y las estructuras de datos, en el apartado correspondiente a la fase de interacción de componentes. Los diagramas de interacción han sido creados como ayuda para hallar las responsabilidades de las operaciones así como nuevas operaciones por lo que no se consideran relevantes para su documentación.

- *Insertar datos recogidos en la fase de interacción*:
    - Entrada: Arquitectura de componentes, Diagramas de interacción, Interfaces de sistema, Interfaces de negocio, Estructuras de datos.
    - Salida: Documentación del proyecto.

5.4.2.3 Fase de especificación de componentes

El objetivo de esta fase es crear los contratos de uso y realización. Los contratos de uso vienen definidos por la especificación de las interfaces y los contratos de realización por la especificación de los componentes.

Los productos de trabajo que se crean en esta fase son:

- *Precondición:* Representa las suposiciones en las que se basa la operación para el correcto funcionamiento, y la postcondición representa las garantías contractuales que las operaciones marcan si esas suposiciones están bien fundadas. Las suposiciones son la responsabilidad del cliente de la operación mientras que las garantías son la responsabilidad del proveedor de la operación.
- *Postcondición:* Especifica el efecto de la operación si la condición previa o precondición es verdadera. La precondición no es una condición bajo la cual deba llamarse a la operación, la invocación de la operación es totalmente independiente de ella. Es la condición bajo la que la operación garantiza que la

postcondición es verdadera. Si la precondición es falsa cuando la operación es invocada la postcondición no se cumplirá y el resultado será incierto.

- *Modelo de información de interfaz*: Es un modelo en el que se detalla las relaciones entre una interfaz con los objetos de negocio que maneja.
- *Reglas de interfaces de sistema*: Son reglas para definir las relaciones entre las interfaces de sistema.
- *Reglas entre interfaces de negocio y sistema*: Son reglas para definir las relaciones entre las interfaces de negocio y sistema.

Esta subfase se ha dividido en 5 actividades según puede verse en la siguiente figura.

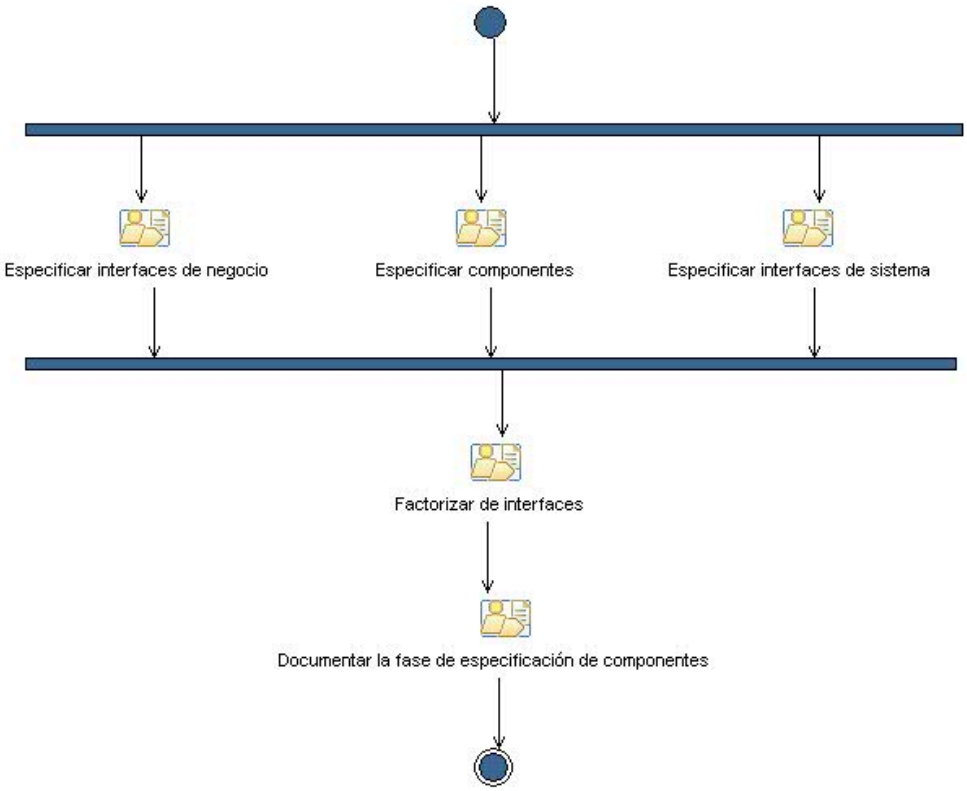


Figura 19. Diagrama de actividad de la fase de especificación de componentes

- **Actividad 2.3.1: Especificar interfaces de negocio**

El analista a partir de las operaciones del diagrama de responsabilidad de interfaz del modelo de tipo negocio crea una especificación completa para ellas. Halla sus pre y postcondiciones y escribe nuevas reglas de negocio.

- *Crear el modelo de información de interfaces*: A partir del modelo de tipo de negocio, el analista crea un modelo de información de interfaz para cada interfaz. En este modelo aparecen las relaciones de la interfaz con las entidades de negocio que maneja.
  - Entradas: Diagrama de responsabilidad de interfaz del modelo de tipo de negocio, Estructuras de datos, interfaces de negocio.
  - Salidas: Modelo de información de interfaces.
- *Añadir pre/postcondiciones*: Para cada operación se escribe su pre y poscondición. Para que sea más claro el proceso, el analista puede realizar snapshots, que son diagramas que muestran el antes y el después de los objetos al ejecutar una operación.
  - Entradas: Modelo de información de interfaces.
  - Salidas: Precondiciones, Postcondiciones.

- *Escribir reglas de negocio*: El analista añade las reglas de negocio encontradas en esta fase a las ya existentes. Serán escritas en lenguaje natural o en OCL.
  - Entradas: Modelo de información de interfaces.
  - Salidas: Reglas de negocio (Invariantes).
- *Completar el modelo de información de interfaces*: El analista añade las precondiciones, postcondiciones y reglas del negocio al modelo de información de interfaces.
  - Entradas: Precondiciones, PostCondiciones, reglas de negocio.
  - Salidas: Modelo de información de interfaces.

- **Actividad 2.3.2: Especificar interfaces de sistema**

Al igual que se ha hecho con las interfaces de negocio el analista crea un modelo de información de interfaces para cada interface de sistema. Será necesario el modelo de tipo de negocio para relacionar las interfaces con los objetos de negocio que maneja.

- *Crear el modelo de información de interfaces de sistema*
  - Entradas: Interfaces del sistema, Especificación de componentes de sistema, estructuras de datos, Diagrama de responsabilidad de interfaz del modelo de tipo de negocio.
  - Salidas: Modelo de información de interfaces.

- **Actividad 2.3.3: Especificar componentes**

Una vez especificadas las interfaces de sistema y negocio se especifican los componentes en esta actividad.

- *Hallar interacciones entre componentes:* Para los componentes de sistema, el analista halla las reglas entre sus interfaces de negocio para tener la seguridad de que las interfaces se han construido de manera consistente.
  - Entradas: Arquitectura de componentes, Especificación de componentes de sistema.
  - Salidas: Reglas de interfaces de sistema.
- *Añadir limitaciones entre interfaces del componente*: Para los componentes de sistema halla las reglas entre las interfaces de negocio con las interfaces de sistema.
  - Entradas: Arquitectura de componentes, Especificación de componentes de sistema, Especificación de componentes de negocio.
  - Salidas: Reglas entre interfaces de sistema y negocio.

- **Actividad 2.3.4: Factorizar interfaces**

El analista una vez hallado el modelo de información de interfaces procede a su factorización y a la creación de un nuevo modelo de información de interfaz factorizado. Posibles factorizaciones son crear supertipos para las interfaces o para los elementos del modelo, definir operaciones comunes entre modelos…

- *Factorización de interfaces*
  - Entradas: Modelo de información de interfaces.
  - Salidas: Modelo de información de interfaces.

- **Actividad 2.3.5: Documentar la fase de especificación de componentes (+)**

En el apartado correspondiente a la fase de especificación de componentes añadimos el modelo de información de interfaces, las reglas de interfaces de sistema y las reglas entre interfaces de sistema y negocio.

- *Insertar datos recogidos en la fase de especificación*
  - Entrada: Modelo de información de interfaces, Reglas de interfaces de sistema, Reglas entre interfaces de sistema y negocio.
  - Salida: Documentación del proyecto.

5.4.3 Fase de aprovisionamiento y desarrollo

El objetivo de esta fase es de especificar los componentes según la tecnología escogida y seleccionar aquellos componentes que necesitemos para la aplicación. Además se desarrollarán aquellos componentes para los que no se ha encontrado un componente reutilizable según su especificación y todos ellos tantos los implementados como los reutilizados serán probados individualmente.

Los productos de trabajo que se crean en esta fase son:

- *Requisitos*: Es el conjunto de todos los requisitos de la aplicación, tantos los requisitos de negocio, seguridad y rendimiento.
- *Tecnología:* Es la tecnología elegida para implementar la aplicación y sus componentes.
- *Componentes candidatos(+)*: Es un conjunto de componentes que cumplen con una serie de requisitos y casos de uso.
- *Características(+)*: Son características que se desea que un componente cumpla con cierto grado de satisfacción.
- *Componentes reutilizados*: Es el conjunto de componente seleccionados. Cada componente es un ejecutable que podría incluir su código fuente si este fuera adaptable.
- *Conjunto de componentes probados*: Es el conjunto de todos los componentes de la aplicación tanto los implementados como los reutilizados.
- *Especificación de la implementación de componentes:* Es una especificación del componente adaptado a una tecnología elegida a partir del modelo de información de la interfaz.
- *Componentes implementados:* Es la implementación de un componente según su especificación para la tecnología elegida.
- *Especificación de los componentes reutilizados*: Especificación y características de los componentes seleccionados.
- *Resultados pruebas unitarias(+)*: Informe con el resultado de las pruebas realizadas a cada componente.

Se ha dividido esta fase en 4 actividades según puede verse en la siguiente figura.

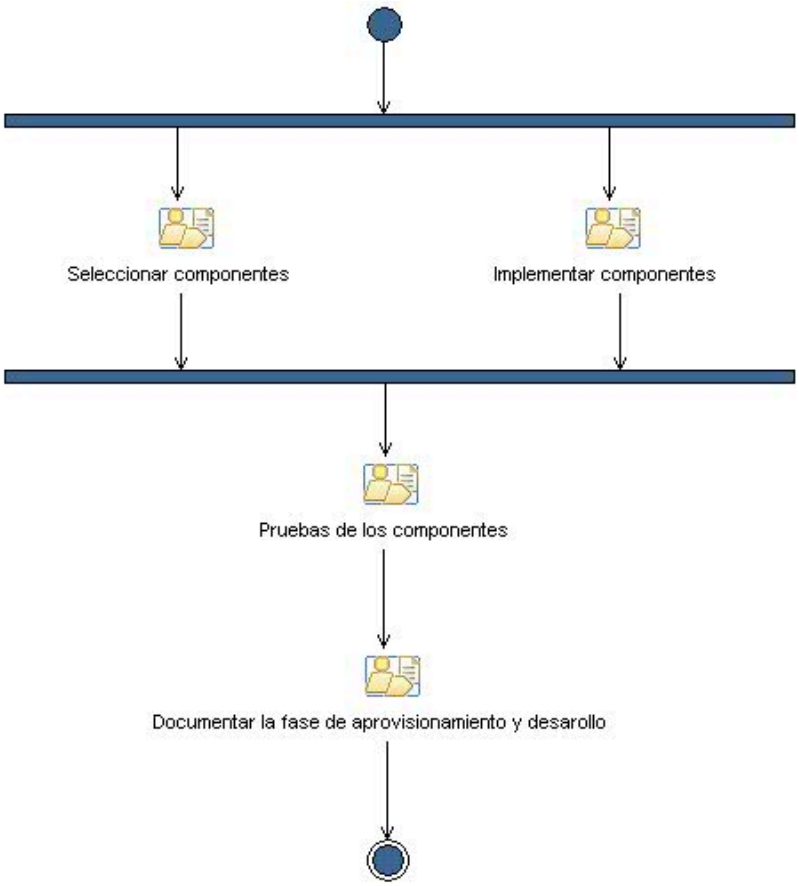


Figura 20. Diagrama de actividad de la fase de aprovisionamiento

- **Actividad 3.1: Seleccionar componentes (+)**

En esta actividad utilizamos una aplicación de escritorio llamada CotsreApp que nos permite seleccionar componentes a partir de sus requisitos y casos de uso. Al finalizar el proceso nos mostrará cual es el componente candidato.

- *Búsqueda de subconjunto de elementos candidatos:* En esta fase a partir de unos requisitos y unos casos de uso se crea una matriz de selección con los componentes candidatos a ser reutilizados, con la ayuda de la herramienta CotsreApp.
  1. El seleccionador introduce los requisitos que el componente debe cumplir. Puede buscar por catálogo los requisitos. A su vez puede filtrar por requisitos funcionales o no funcionales.
  2. Selecciona un caso de uso que debe cumplir.
  3. CostreApp genera la matriz de selección de componentes que cumplen con los requisitos y casos de uso introducidos.
  4. Si no existe un único componente candidato o sino existe ninguno la aplicación propondrá un conjunto de candidatos que cumpla con más de la mitad de requisitos y caso de uso.
     - Entradas: Requisitos, Descripción de casos de uso.
     - Salidas: Componentes candidatos.
- *Elección del componente a reutilizar*: En esta tarea el seleccionador introduce las características deseables del componente y su grado de satisfacción. Y la aplicación selecciona el mejor candidato según esas características. Si existe un empate entre candidatos el seleccionador puede incluir o quitar características para desempatar o elegir él mismo el componente entre los empatados.

        - Entradas: Características, Componentes Candidatos.
        - Salidas: Componentes reutilizados, Especificaciones del componente reutilizados.

- **Actividad 3.2: Implementar componentes**

Se mapea la especificación del modelo de información de interfaz para su implementación según la tecnología. Esta actividad se divide en tres tareas, primero se elige la tecnología, después se crea una especificación de cada componente según esta tecnología y por último se realiza la implementación.

  - *Elegir tecnología*: De todas las tecnologías existentes se realiza un estudio para averiguar cuál es la que mejor se adecua a los requisitos de la aplicación.
    - Entradas: Requisitos del negocio, Requisitos de seguridad, Requisitos de rendimiento.
    - Salidas: Tecnología.
  - *Mapeado de la especificación con la tecnología*: A la hora de realizar el mapeado se tienen en cuenta los siguientes puntos que tienen que ver con las características de la tecnología elegida.
    1. Tipos de parámetros de la operación, la clase y restricciones de referencias. Lo normal será encontrarnos con dos tipos de parámetros, cualquier dato que se pasará por valor y los objetos que se pasarán por referencia.
    2. Mecanismo de manejo de excepciones y errores.
    3. Herencia de interfaz y restricciones soportadas.
    4. Secuencia de operación.
    5. Propiedades de la interfaz.
    6. Mecanismo de creación de objetos.
    7. Manejo de eventos.
       - Entradas: Tecnología, Modelo información de interfaces, Reglas de interfaces de sistema, Reglas entre interfaces de sistema y negocio.
       - Salidas: Especificación de la implementación de componentes.
  - *Implementación del componente*: El programador realiza la implementación de los componentes para los que el seleccionador no haya encontrado un componente reutilizable. Se hará conforme a la especificación de la implementación realizada en la anterior tarea.
    - Entrada: Especificación de la implementación de componentes.
    - Salidas: Componentes implementados.

- **Actividad 3.3: Pruebas de los componentes (+)**

El testeador diseña y realiza pruebas unitarias a cada componente, tantos los reutilizados como los implementados.

  - *Pruebas de componente*
    - Entradas: Componentes reutilizados, Componentes implementados.
    - Salidas: Resultados pruebas unitarias.

- **Actividad 3.4: Documentar la fase de aprovisionamiento y desarrollo (+)**

El documentalista añade la tecnología, especificaciones de los componentes reutilizados, la especificación de la implementación de los componentes y los resultados de las pruebas unitarias, en el apartado correspondiente a la fase de aprovisionamiento de componentes.

  - *Insertar datos recogidos en la fase de aprovisionamiento:*

- Entradas: Tecnología, Especificaciones de los componentes reutilizados, Especificación de la implementación de los componentes, Resultados pruebas unitarias.
- Salida: Documentación del proyecto.

5.4.4 Fase de ensamblamiento y despliegue

En esta fase se realiza la implementación de la aplicación. Se irán realizando pruebas de integración de los componentes además de las pruebas que los usuarios finales realicen sobre el funcionamiento y la interfaz visual. El objetivo de esta fase es la entrega final de la aplicación junto con un manual de utilización a los clientes. Además al término de esta fase se catalogan los componentes y requisitos para una futura reutilización.

Los productos de trabajo que se crean en esta fase son:

- *Prototipo funcional(+):* Es la aplicación que implementa algunas funciones y a medida que se comprueba que son las apropiadas, se corrigen, se refinan y se van añadiendo otras.
- *Prototipo probado(+):* Es el prototipo que ha pasado con éxito las pruebas realizadas por los testeadores a partir de unos casos de prueba.
- *Prototipo validado(+):* Es un prototipo que ha sido probado y validado por los usuarios finales.
- *Aplicación*: Es la aplicación ejecutable producto de las especificaciones del usuario y el análisis realizado durante todo el proceso.
- *Manual de usuario(+):* Es un documento que explica el funcionamiento de la aplicación al usuario.
- *Catálogo de componentes(+):* Es un catálogo de componentes detallado para su futura reutilización.
- *Catálogo de requisitos(+):* Es un catálogo de requisitos que cumplen los componentes del catálogo de componentes. Cada requisito tiene un atributo "fuente" que apunta a un conjunto de componentes que lo cumple.
- *Casos de prueba:* Es un conjunto de pruebas que deben realizarse a la aplicación para comprobar su correcto funcionamiento y ayudar a detectar posibles errores.

Esta fase se ha divido en tres actividades según se muestra en la siguiente figura.

Figura 21. Diagrama de actividades de la fase de despliegue y pruebas

- **Actividad 4.1: Generar aplicación**

Se genera la aplicación final, para ello el programador realiza un prototipo que va refinando en el que añade la interfaz de usuario y la integración de los componentes desarrollados y reutilizados.

  - *Desarrollar interfaces de usuario:* El programador desarrolla la interface de usuario (ventanas y formularios).
    - Entradas: Modelo de información de interfaces.
    - Salidas: Prototipo funcional.
  - *Integración de componentes*: El programador complementa el prototipo integrando los componentes implementados a la interfaz de usuario.
    - Entradas: Componentes implementados, Componentes reutilizados, Reglas de interfaces de sistema, Reglas entre interfaces de negocio y sistema.
    - Salidas: Prototipo funcional.

- **Actividad 4.2: Pruebas de la aplicación (+)**

Los testeadores diseñan y realizan las pruebas del prototipo. Estas pruebas están dirigidas a encontrar fallos de integración de componentes. Los usuarios finales también realizan pruebas para comprobar que el funcionamiento es el esperado y si aceptan la interfaz gráfica creada.

  - *Diseño de las pruebas del prototipo:* A partir de los casos de uso del sistema y de los requisitos de la aplicación se diseñan una serie de casos de prueba.
    - Entradas: Descripción de los casos de uso.
    - Salidas: Casos de prueba.
  - *Realizar pruebas al prototipo:* Se realizan las pruebas siguiendo el guión de los casos de prueba.
    - Entradas: Prototipo funcional, Casos de prueba.
    - Salidas: Prototipo probado.

- *Realizar pruebas con los usuarios finales:* Se deja la aplicación a los usuarios finales para que realicen las pruebas que ellos crean convenientes. Sobre todo deberán realizar pruebas de validación del funcionamiento y aceptación de la interfaz de usuario creada.
  - Entradas: Prototipo probado.
  - Salidas: Prototipo validado.

– **Actividad 4.3: Finalizar el proceso (+)**

En cuanto el prototipo está totalmente probado y validado se crea la aplicación final y el manual de usuario de ella y se realiza el despliegue en los clientes. Por último, se catalogan los requisitos y componentes en la herramienta CotsreApp para su futura reutilización.

- *Generar aplicación final:* Una vez validado el prototipo el programador realiza el ejecutable de la aplicación final y lo que se necesite para poder realizar el despliegue de la aplicación.
  - Entradas: Prototipo validado.
  - Salidas: Aplicación.
- *Escribir manual de usuario:* Se escribe un manual de usuario sobre el funcionamiento de la aplicación.
  - Entradas: Aplicación.
  - Salidas: Manual de Usuario.
- *Entregar aplicación:* Se realiza el despliegue de la aplicación en el entorno de los usuarios finales y se les hace entrega además del manual de usuario.
  - Entradas: Aplicación, Manual de Usuario.
  - Salidas: En esta tarea no se produce ninguna salida.
- *Catalogación del proceso:* Se catalogan los componentes y requisitos de la aplicación para una futura reutilización.
  1. Introducción de requisitos: Se detallan los requisitos en caso de que no se encuentren ya en el catálogo de requisitos mediante la herramienta CotsreApp.
  2. Introducción de componentes: Se introducen los nuevos componentes junto con información detallada de ellos, ámbito de aplicación, autores, año, descripción, localización, casos de uso y características y requisitos que cumplen.
     - Entradas: Modelo de información de interfaz, Requisitos de rendimientos, Requisitos de seguridad, Descripción de casos de uso.
     - Salidas: Catálogo de componentes, Catálogo de requisitos.

5.5 Aportaciones de COTSRE+ sobre el proceso de Chessman y Daniels

En el anterior apartado hemos realizado la especificación de COTSRE+ usando la notación de SPEM, el cual como hemos comentado se basa en el proceso definido por Chessman y Daniels. Hemos marcado con un "+" aquellas actividades, tareas y productos de trabajo nuevos que se han introducido en este modelo.

El cambio más evidente ha sido en las fases, en Chessman y Daniels se definían 6 fases, de requisitos, especificación, aprovisionamiento, ensamblaje, pruebas y despliegue. En COTSRE+ se han respetado las dos primeras pero se han unido las fases de ensamblaje y despliegue de la aplicación. Además la fase de pruebas ha sido eliminada para poder realizar pruebas unitarias de los componentes en la fase de

aprovisionamiento y las pruebas de la aplicación completa en la fase de ensamblaje y despliegue. Las actividades correspondientes a las pruebas son, la actividad "Pruebas de los componentes" (3.3) en la fase de aprovisionamiento y "Pruebas de la aplicación" (4.2) en la fase de ensamblamiento y despliegue.

En la fase de aprovisionamiento se ha realizado una de las principales aportaciones de COTSRE+. Esta ha sido la selección de componentes basándonos en el ya existente método de selección de COTSRE, pero ampliado para encontrar componentes también a partir de los casos de uso. Chessman y Daniels aconsejaban intentar reutilizar componentes del mercado pero no indicaban cómo hacerlo ni que método seguir para encontrar el más adecuado. La actividad donde se ha realizado esto es en "Seleccionar componentes" (3.1).

La fase de ensamblamiento propuesta por Chessman y Daniels no aportaba mucha información de cómo debía realizarse. Sólo se especificaba que debía desarrollarse la interfaz de usuario y la integración de componentes. Así que para COTSRE+ se ha propuesto la realización de un prototipo para ir generando la aplicación así como un manual de usuario tal como vemos en la actividad "Finalizar el proceso" (3.4). Además en esta actividad se incluye una tarea para catalogar los componentes y requisitos en la herramienta CotsreApp.

Otra aportación interesante ha sido la definición de los roles que actúan a lo largo de todo el método (usuarios finales, expertos del dominio, ingeniero de requisitos, analista, seleccionador, testeador, programador, documentalista).

Se ha creído conveniente hacer una buena documentación al termino de cada fase para conseguir una buena documentación de todo el proceso de desarrollo de la aplicación. Las actividades creadas son las actividades "Documentar la fase de requisitos" (1.5), "Documentar de la fase de identificación de componentes" (2.1.4), "Documentar la fase de interacción de componentes" (2.2.3), "Documentar la fase de especificación de componentes" (2.3.5) y los productos de trabajo manejados son "Documentación del proyecto provisional" y "Documentación del proyecto aprobada".

La última aportación ha sido la de proporcionar unas plantillas de casos de uso y otra para generar el documento de desarrollo que se corresponden con los productos de trabajo "Plantilla de documento del proyecto" y "Plantilla de caso de uso".

6. Edición de COTSRE+ con EPF Composer

Para editar nuestro proceso especificado con SPEM vamos a utilizar una herramienta gratuita desarrollada en el entorno de Eclipse llamada *Eclipse Process Framework Composer* (EPF) [31]. Con ella crearemos una biblioteca y generaremos una página web con los datos del proceso ordenados de manera que estén accesibles para cualquier persona que desea utilizar el método para realizar una aplicación basada en componentes. La biblioteca podrá ampliarse en un futuro según se amplíe COTSRE+.

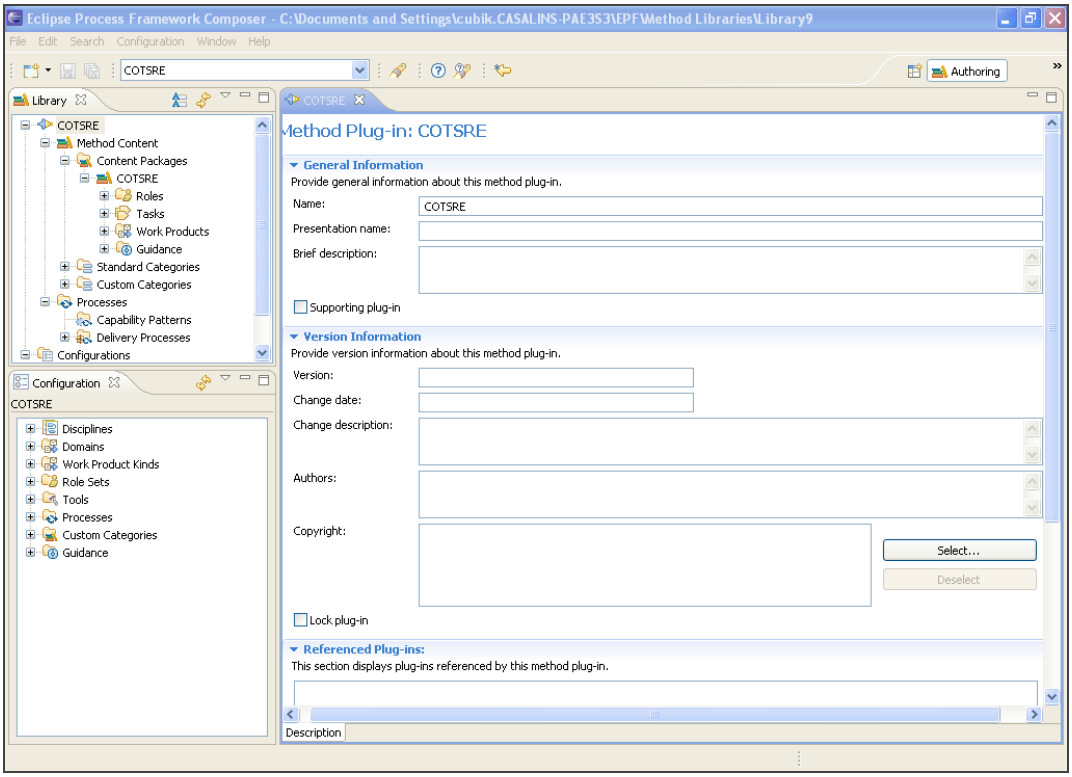


Figura 22. Entorno de Eclipse Process Framework Composer (EPF)

Una biblioteca (*Method Library*) es una colección que está formada por uno o más plugins y configuraciones. Cada plugin contiene un paquete de contenido de método (*Method Content*) que define los elementos del proceso (roles, tareas, productos de trabajo y guías) y el paquete con la estructura de procesos (*Processes*) que define el ciclo de vida del proceso que se está editando utilizando SPEM. El contenido de método a su vez está formado por uno o más paquetes de contenido (*Content Package*) los cuales son organizados por el usuario, categorías estándar (*Standard Category*) y categorías personalizadas (*Custom Category*). Los procesos están formados por patrones de proceso (*Capability Pattern*) y procesos para despliegue (*Delivery Process*). Además a la biblioteca se añaden una o más configuraciones (*Configurations*) de vistas para su publicación. Todo esto es explicado en detalle en los siguientes subapartados.

Esta estructura podemos verla en la figura 23. COTSRE sería el nombre que le hemos puesto a la biblioteca donde se definirán todos los elementos (roles, tareas y productos de trabajo) necesarios para definir su ciclo de vida del método COTSRE+. Como se observa está compuesta de un paquete contenido de método y un paquete de procesos. Se ha creado una configuración llamada también COTSRE para las vistas de la publicación web.

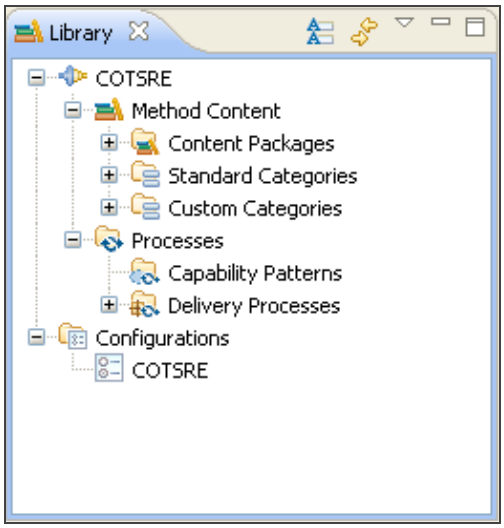


Figura 23. Estructura de la librería COTSRE

6.1 Contenido de método (Method Content)

El primer paso ha sido crear un contenido de método dentro de la biblioteca COTSRE, en el que se han creado todas las tareas, roles, productos de trabajo y guías del proceso. Además se ha creado una categoría personalizada para ordenar todos los elementos de contenido.

6.1.1 Categorías

Una categoría es un elemento de contenido que se usa para clasificar o agrupar el resto de elementos de contenido.

Para cada categoría existen 5 tipos de categoría que son:

- Conjunto de roles (*Rol Set*): Encargada de agrupar los roles que tienen características en común. Hemos creado dos conjuntos de roles, el de cliente y técnico para clasificar todos los roles creados.
- Disciplina *(Discipline*): En ellas se agrupan las tareas relacionadas con un área del proceso. En nuestro caso hemos creado cuatro disciplinas correspondientes a las cuatro fases del proceso llamadas requisitos, especificación, aprovisionamiento, y ensamblaje para clasificar las distintas tareas.
- Dominio (*Domain*): Para hacer una clasificación de los productos de trabajo, al igual que con las tareas, se ha optado por crear cuatro dominios correspondientes con las cuatro fases del proceso también llamados requisitos, especificación, aprovisionamiento y ensamblaje.
- Herramienta (*Tool*): Categoriza las guías.
- Clase de producto de trabajo (*Work Product Kind*): No ha sido utilizada, simplemente aparece en EPF por compatibilidad con la versión 1 de SPEM.

Además se han creado una serie de categorías personalizadas:

- Introducción: Para ofrecer información general sobre el método COTSRE+. Esto sería la página de inicio de la publicación web del proceso.
- Guías: Para agrupar las plantillas de documentos (casos de uso y documento de desarrollo).
- Acerca de: Información sobre la autora del proceso.

En la siguiente imagen podemos ver todas las categorías.

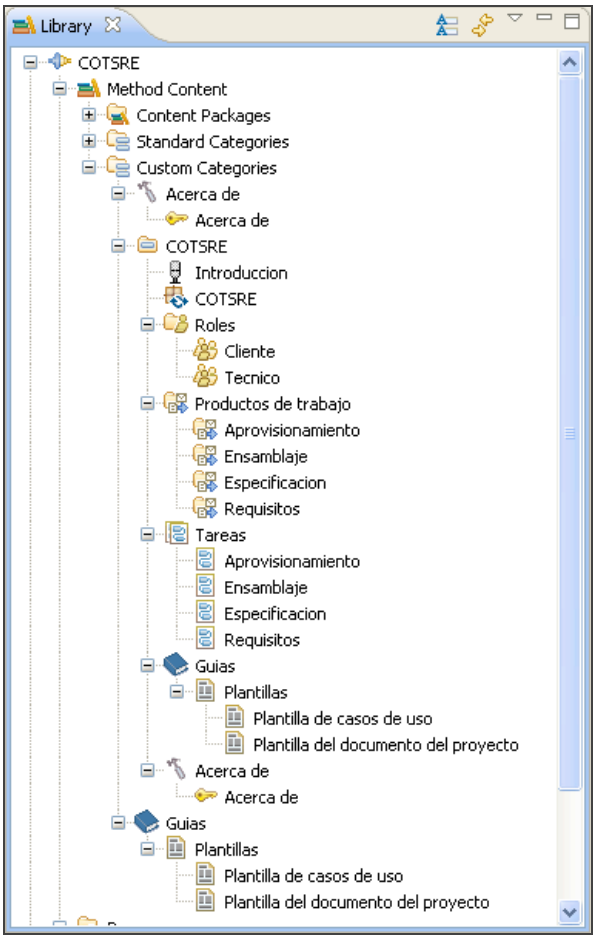


Figura 24. Estructura de las categorías de COTSRE

6.1.2 Roles

Los roles definen un conjunto de habilidades, competencias y responsabilidades dentro del proceso. Se han introducido los roles descritos en el método COTSRE+ ordenados por categorías (usuario y técnico), para cada uno además se ha añadido una descripción.

En la siguiente figura vemos como dentro del paquete contenido COTSRE en la carpeta de Roles se han creado los roles del proceso ("Analista", "Documentalista", "Expertos en el dominio", "Ingeniero de requisitos", "Programador", "Seleccionador", "Testeador", "Usuario final"). Al crear un rol aparece una plantilla con 5 pestañas:

- *Description*: Se rellenan los campos de nombre y descripción como vemos en la figura.
- *Work Products*: Se seleccionan todos los productos de trabajo en los que participa.
- *Guidance*: Provee información adicional externa si fuera necesario.
- *Categories*: Se selecciona el conjunto de rol al que pertenecen. En la figura, el rol analista habrá sido añadido al conjunto Técnico.
- *Preview*: Se previsualiza a información del rol en HTML.

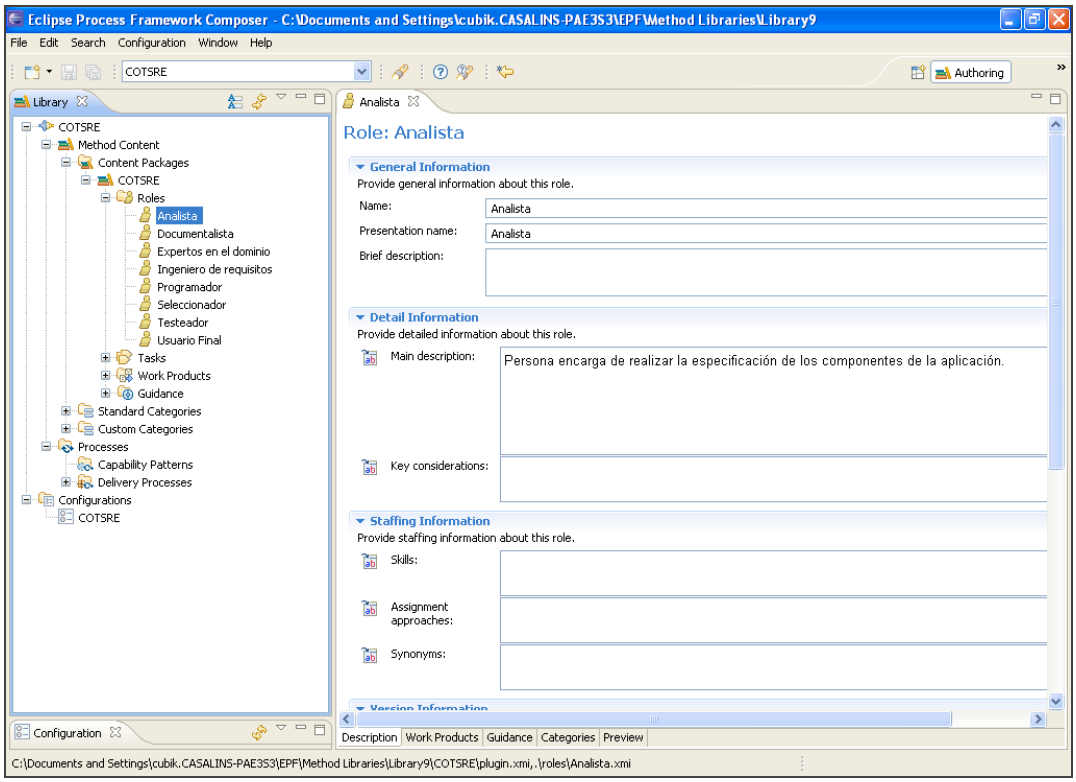


Figura 25. Pestaña creación de un rol

6.1.3 Productos de trabajo

Un producto de trabajo es consumido, producido o modificado por las tareas. Se han añadido todos los productos de trabajo identificados en el proceso ordenados por dominios (requisitos, especificación, aprovisionamiento y ensamblamiento).

En la siguiente figura dentro de *Work Products* podemos ver el listado de todos los procesos de trabajo creados para COTSRE+. Los productos de trabajo se han creado como artefactos que son los que tienen una naturaleza tangible. Para crearlos aparece una plantilla con cuatro pestañas:

- *Description*: Se rellenan los campos de nombre y descripción del producto de trabajo. En la figura podemos ver el producto de trabajo "Diagrama de Casos de Uso".
- *Guidance*: Provee información adicional externa si fuera necesario.
- *Categories*: Se selecciona el dominio al que pertenece, para el producto de trabajo de la figura se habrá seleccionado el dominio "Requisitos".
- *Preview*: Se previsualiza la información introducida del producto de trabajo en HTML.

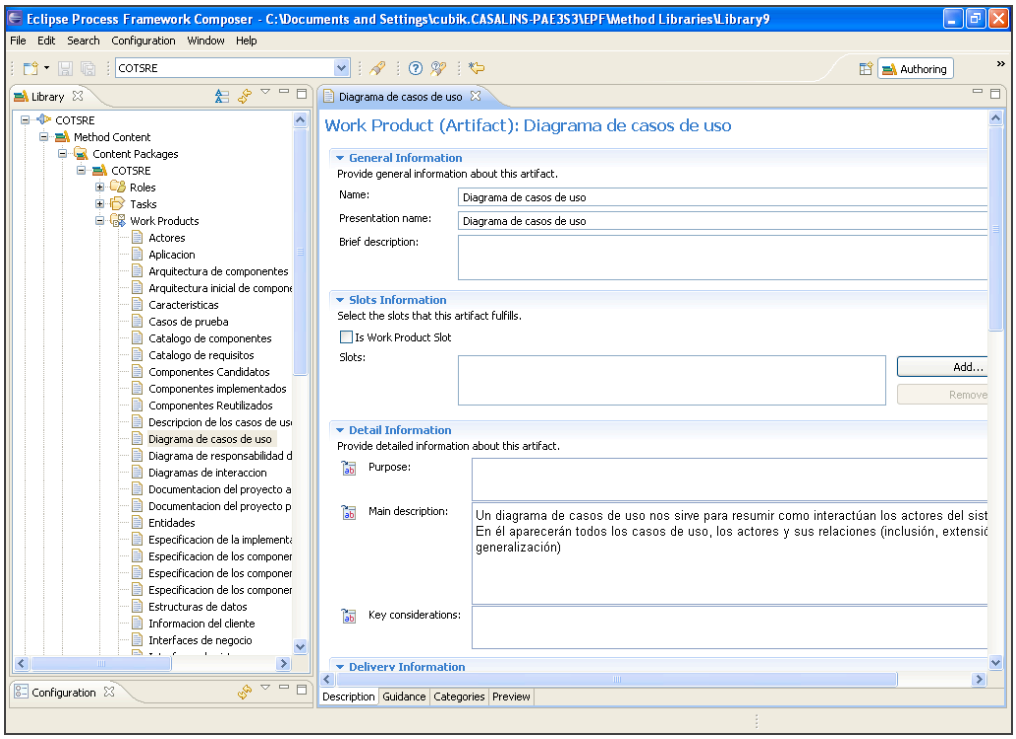


Figura 26. Pestaña creación de un producto de trabajo

6.1.4 Tareas

Las tareas describen una unidad de trabajo asignable y gestionable. Se han creado todas las tareas del método de COTSRE+ ordenadas por la disciplina a la que correspondan (requisitos, especificación, aprovisionamiento y ensamblamiento).

Al crear una tarea aparece una plantilla con 7 pestañas con información a rellenar, estas pestañas son:

- *Description*: Se ha introducido por cada tarea nombre y descripción.
- *Steps*: Son los pasos que se realizan para completar cada tarea. La mayoría de las tareas creadas son muy simples y no han requerido su desglose en steps.
- *Roles*: Se seleccionan los roles que han intervenido en la tarea.
- *Work Products*: Se seleccionan los productos de trabajo que se han utilizado en la tarea.
- *Guidance*: Provee información adicional externa si fuera necesario.
- *Categories*: Se selecciona la disciplina a la que pertenece. Por ejemplo, la tarea de la figura "Búsqueda de un subconjunto de componentes candidato" pertenece a la disciplina "Aprovisionamiento".
- *Preview*: Se previsualiza la información introducida de la tarea que se está editando en HTML.

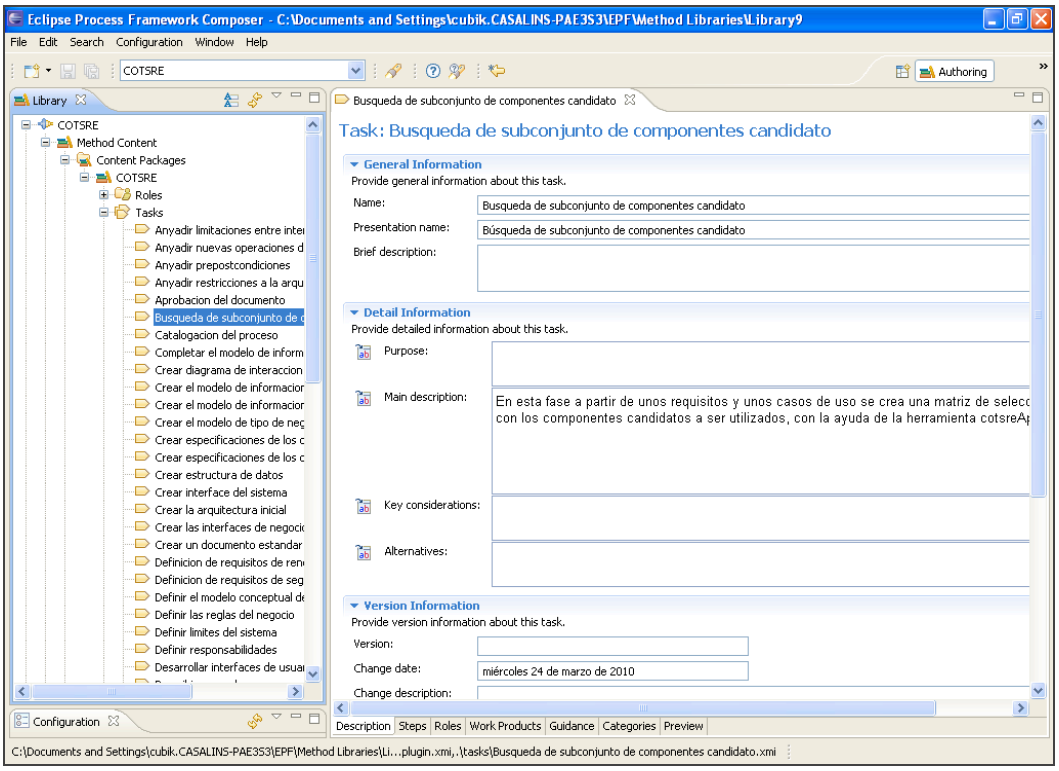


Figura 27. Pestaña creación de una tarea

6.1.5 Guías

Una guía es un elemento de método que provee información adicional relacionada con otros elementos de método. Existen diferentes tipos de guías, para esté proceso se han creado las siguientes:

- Plantillas: Hemos creado plantillas para la descripción de los casos de uso (anexo II) y la documentación del proyecto.
- Material de soporte: Una introducción que explica lo que es el método COTSRE+ y con accesos directos a la página web de los roles, productos de trabajo, tareas, ciclo de vida y guías.
- Conceptos: Información sobre la autora del método.

En la figura vemos un ejemplo de la edición de la guía de introducción, en la primera pestaña se muestra la introducción de la descripción. Para una visualización en HTML se accede a la pestaña de “Preview”.

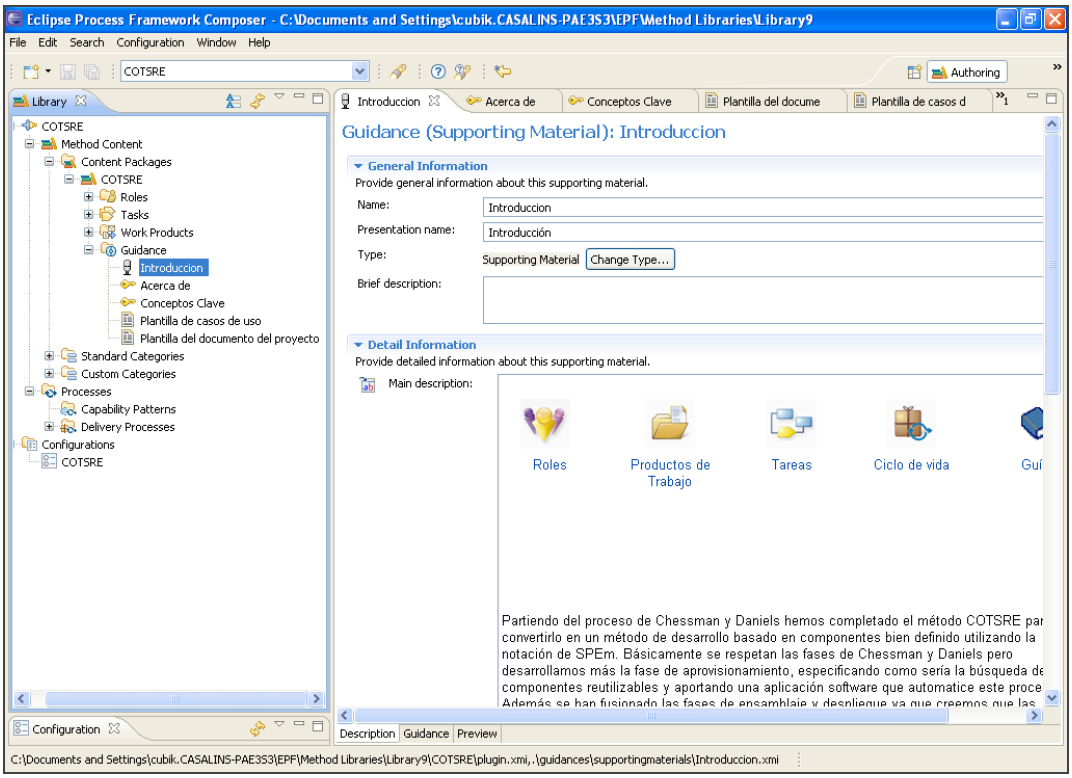


Figura 28. Pestaña creación de una guía

6.2 Procesos

Una vez definidos los elementos de contenido del método (tareas, roles y productos de trabajo) se reutilizan para crear procesos. Aunque estos elementos se pueden definir directamente al crear un proceso esto dificulta mucho la posterior reutilización y actualización de dichos elementos. Por lo que, se ha optado por crear primero los elementos (roles, tareas y productos de trabajo) y reutilizarlos después para crear los procesos.

Con la herramienta EPF se pueden crear los procesos de SPEM de dos formas distintas:

- Patrones de proceso: Describen una agrupación reutilizable de tareas o actividades.
- Proceso para despliegue: Describen un enfoque completo para una metodología completa compuesta por un ciclo de vida. Se ha optado por crear este tipo de proceso para COTSRE+ ya que es una metodología completa. En la siguiente imagen podemos ver la plantilla para rellenar con la información de este proceso.

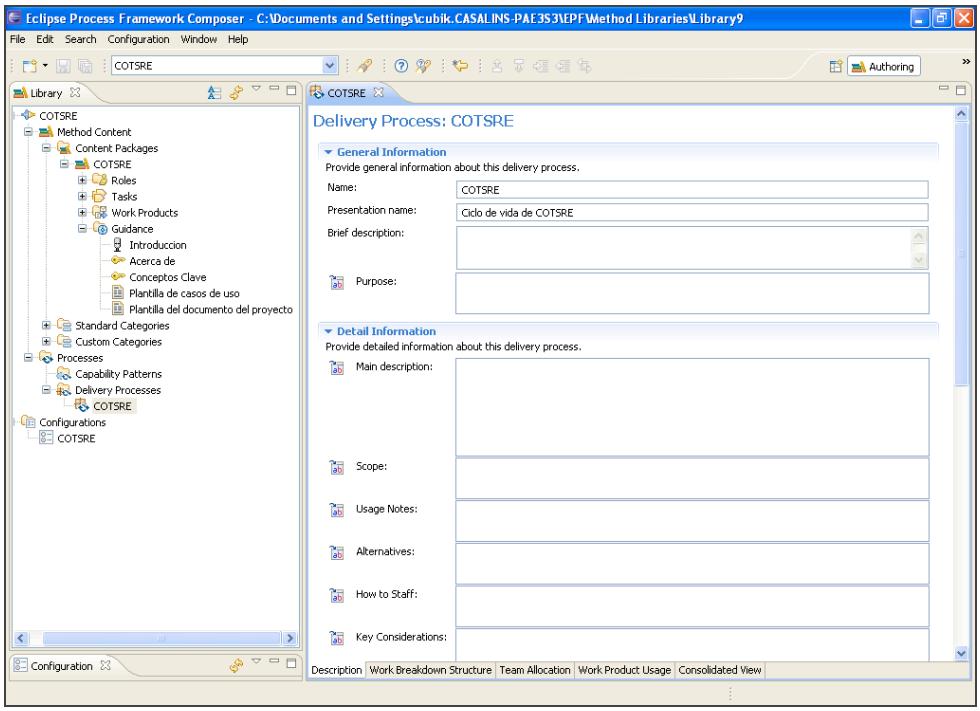


Figura 29. Pestaña creación de un proceso

6.2.1 Ciclo de vida

Se ha creado un ciclo de vida para COTSRE+ el cual está formado por fases. Una fase representa un periodo de tiempo significativo para un proyecto que acaba en un hito. Se han insertado las 4 fases identificadas en COTSRE+ (requisitos, especificación, aprovisionamiento y desarrollo, y ensamblaje y despliegue), además de las subfases de la fase de especificación (identificación, interacción y especificación). Las fases en EPF tienen la propiedad iteración a falso por lo que ha sido preciso incluir un elemento iteración que representa un conjunto de actividades anidadas que se repiten más de una vez. Además a las fases se le ha añadido un hito que hemos llamado *Objetivo* que representa un evento significativo para el desarrollo de un proyecto.

En la siguiente figura se visualiza la pestaña *Work Breakdown Structure* de la plantilla de creación de un proceso. En ella podemos ver la estructura del ciclo de vida de COTSRE+ desglosado en fases cada una con una iteración y un objetivo.

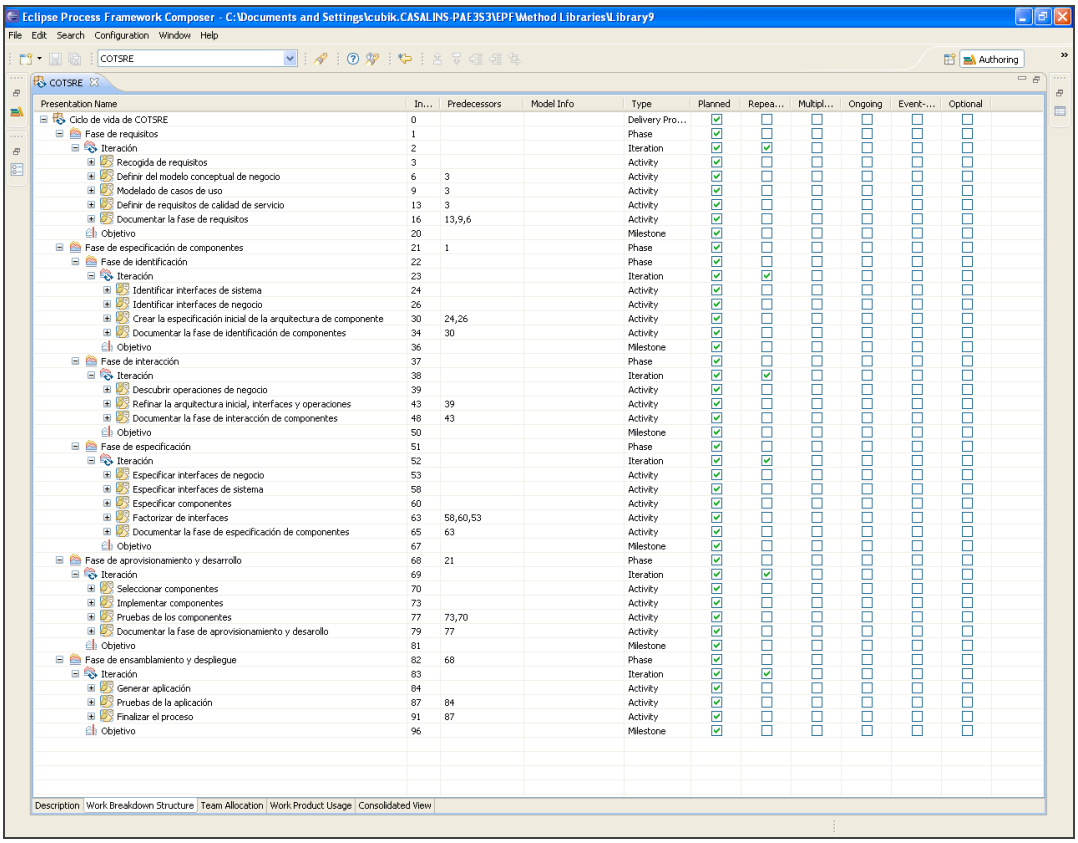


Figura 30. Pestaña Ciclo de Vida COTSRE+. Fases, Iteraciones y Objetivos

A cada iteración se le han añadido las actividades correspondientes a cada fase. Estas actividades están compuestas por las tareas del contenido de método creadas anteriormente. Estas actividades crean una copia de estas tareas.

En la siguiente figura vemos un ejemplo de desglose de una fase, la de requisitos, en actividades y estas a su vez en tareas.

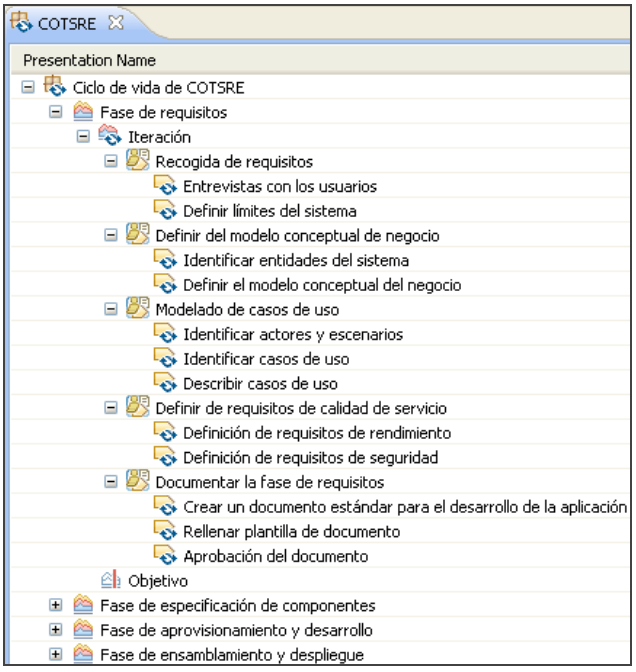


Figura 31. Pestaña Ciclo de Vida COTSRE+. Actividades y Fases.

6.2.2 Diagramas

Existen tres tipos de diagramas SPEM que EPF genera automáticamente y que pueden modificarse mediante un editor gráfico. Estos diagramas son diagramas de actividad, de detalle de actividad y de dependencias de productos de trabajo de los cuales ha parecido interesante utilizar los dos primeros.

- Diagrama de actividad: Estos diagramas muestran el flujo de control entre las actividades de la iteración de la fase seleccionada. EPF genera en el editor gráfico todas las actividades pertenecientes a la fase seleccionada y mediante las herramientas que ofrece podemos ordenar las actividades, crear los nodos de inicio y fin, nodos de bifurcación y unión… para crear el diagrama. En la figura podemos ver el editor de diagramas mostrando el diagrama de actividad de la fase de requisitos.

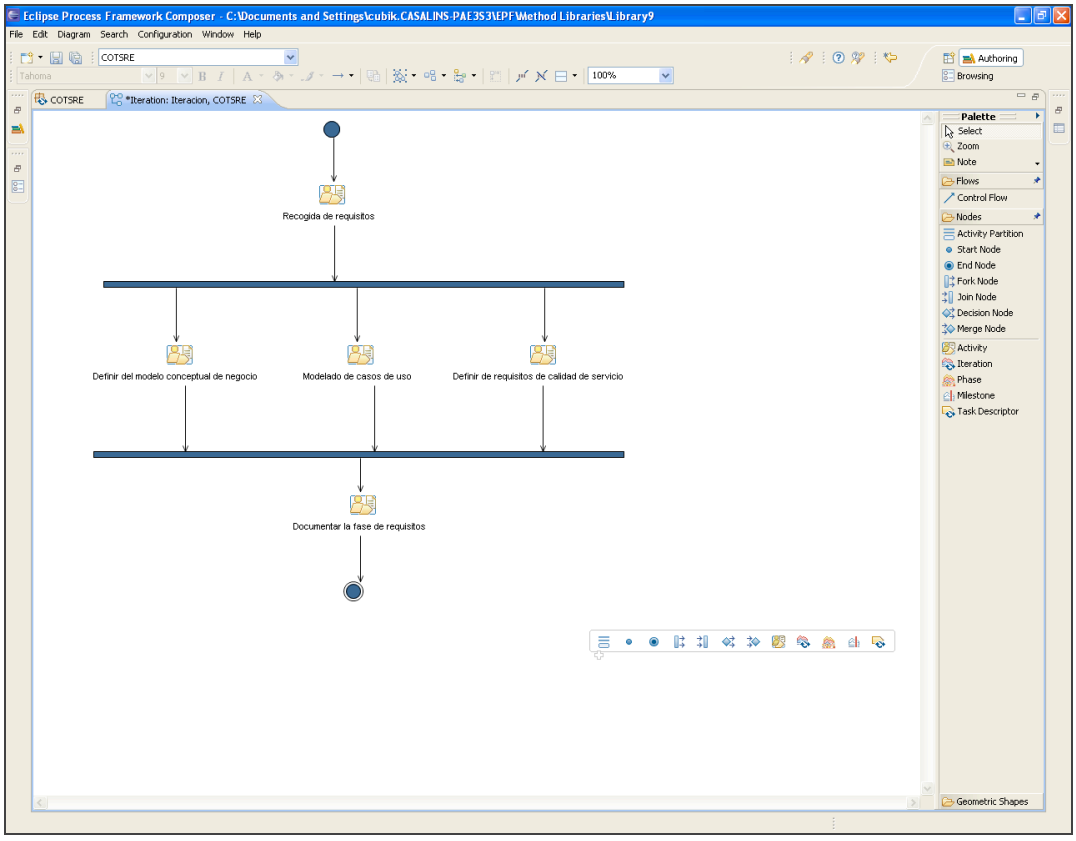


Figura 32. Editor de Diagramas de Actividad

- Diagrama de detalle de actividad: La información que detallan estos diagramas son las tareas, roles y productos de trabajo de cada actividad. Estos diagramas los genera automáticamente EPF y no se puede hacer ninguna modificación en ellos, salvo incluir texto. En la siguiente figura podemos ver el editor de este tipo de diagramas con la edición de la actividad "Definición del modelo conceptual de negocio".

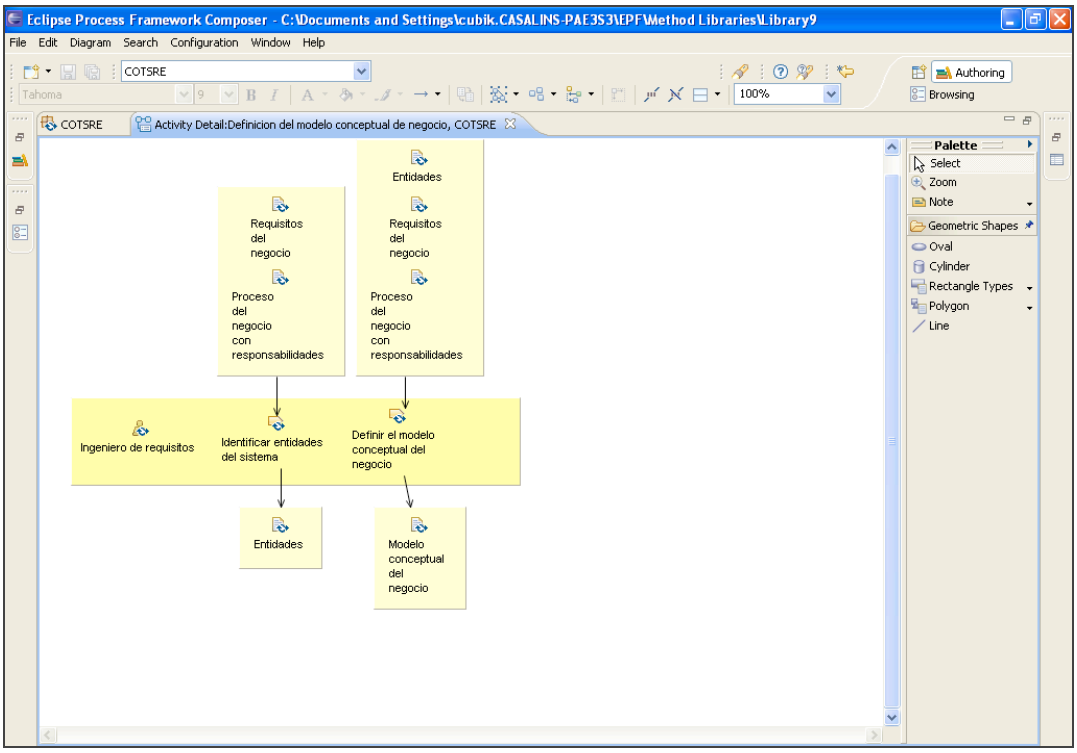


Figura 33. Editor Diagrama de Detalle de Actividad

6.3 Configuración de método

Una configuración de método es una selección de elementos y procesos que queremos incluir para su visualización en la web. Para esto resultan muy útiles las categorías para que la visualización sea estéticamente más ordenada. Para nuestro proyecto de EPF se ha creado una configuración llamada COTSRE en la cual se incluyen las categorías Introducción, Ciclo de Vida, Roles, Tareas, Productos de Trabajo, Guías y Acerca de. A continuación, vemos en la imagen como hemos incluido en la configuración el *method plugin* de COTSRE+.

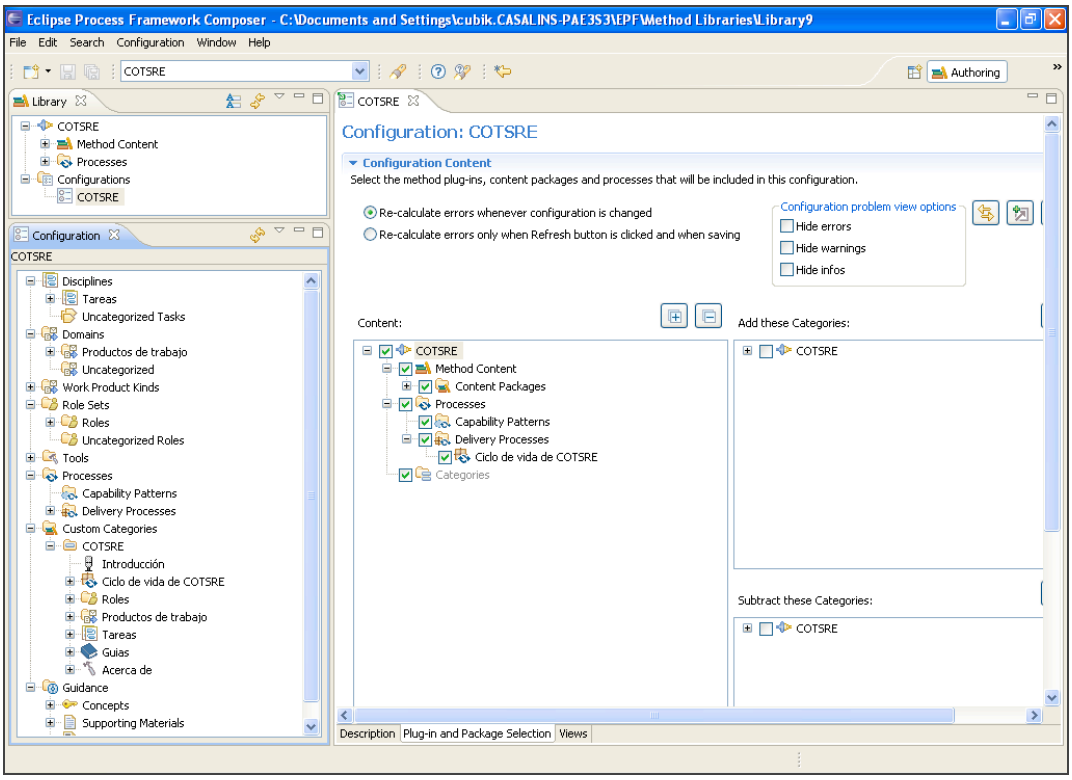


Figura 34. Pestaña configuración

Aquí vemos que hemos incluido todas las categorías creadas y las hemos ordenado según queremos que aparezca en la publicación web.

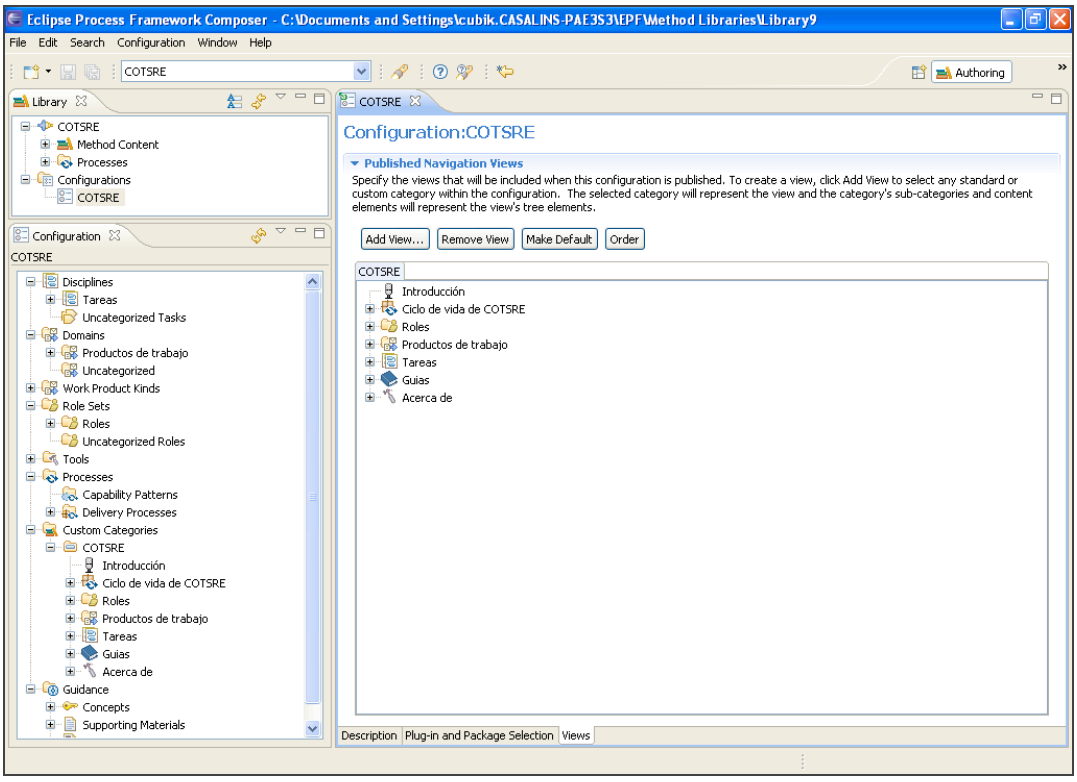


Figura 35. Configuración de las vistas

6.4 Publicación

El resultado de crear una biblioteca con la información del método COTSRE+ en EPF Composer es un sitio web que permite un rápido despliegue de los procesos y una navegación ágil para los usuarios del método. Este sitio web que genera está basado en la configuración creada, por lo tanto se pueden generar distintas vistas a partir de diferentes configuraciones dando como resultados varios sitios web distintos. Por ejemplo, se podría diseñar una vista especial para programadores en la que solo aparezcan las actividades, tareas, productos… que tengan relación directa con el desarrollo del código. En nuestro caso se ha decidido crear una web que muestre todo el proceso de COTSRE+ y que sea consultable por cualquier tipo de usuario.

A la hora de realizar la publicación web, EPF Composer permite crear una web html o J2EE (generando un archivo *.war*) con toda la información del proceso desarrollado. Se ha optado por una publicación *html* para COTSRE+.

Los pasos seguidos para crear nuestra publicación web son:

- Seleccionar la configuración creada para COTSRE+.
- Seleccionar la biblioteca creada.
- Especificamos las opciones concretas como el título de la web, información que queremos incluir, publicación de todos los diagramas…
- Finalmente elegir sitio web generándose todas las páginas html en una carpeta específica.

En el siguiente capítulo explicamos con más detalle la estructura del sitio web creado.

7. Publicación Web de COTSRE+

A la izquierda tenemos un menú con los distintos elementos mostrados para una mejor navegación.

7.1 Introducción

Es la página principal que puede verse cuando se entra en la publicación. En ella vemos un resumen del método COTSRE+ y accesos a los distintos elementos de la web.

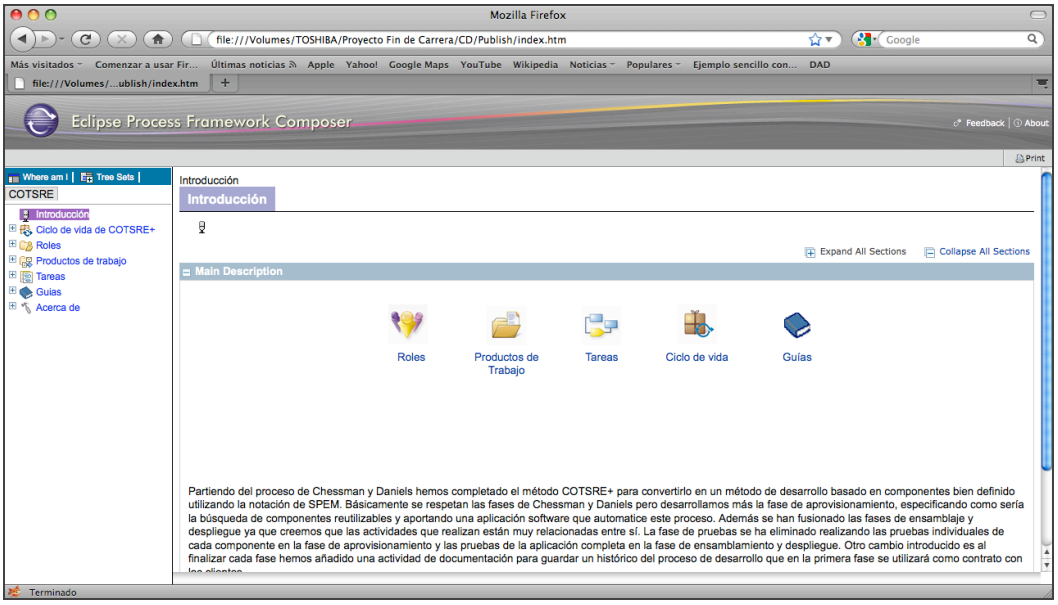


Figura 36. Página de inicio

7.2 Ciclo de vida de COTSRE+

En el Ciclo de vida de COTSRE muestra toda la información relativa al proceso dividida en fases.

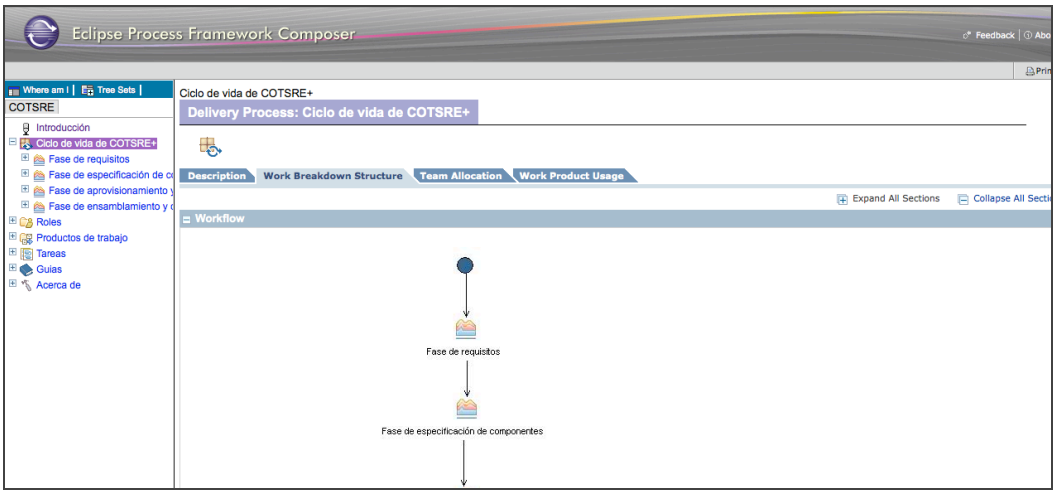


Figura 37. Vista del Ciclo de Vida COTSRE+

Por cada fase tenemos unas iteración y un objetivo. En la iteración nos muestra su descripción (*Description*), su estructura (*Work Breakdown Structure*), roles (*Team*

Allocation) y productos de trabajo (*Work Product Usage*). En la pestaña estructura aparecen el diagrama de actividad y un desglose con las actividades de la fase a la que pertenezca la iteración como vemos las dos siguientes figuras.

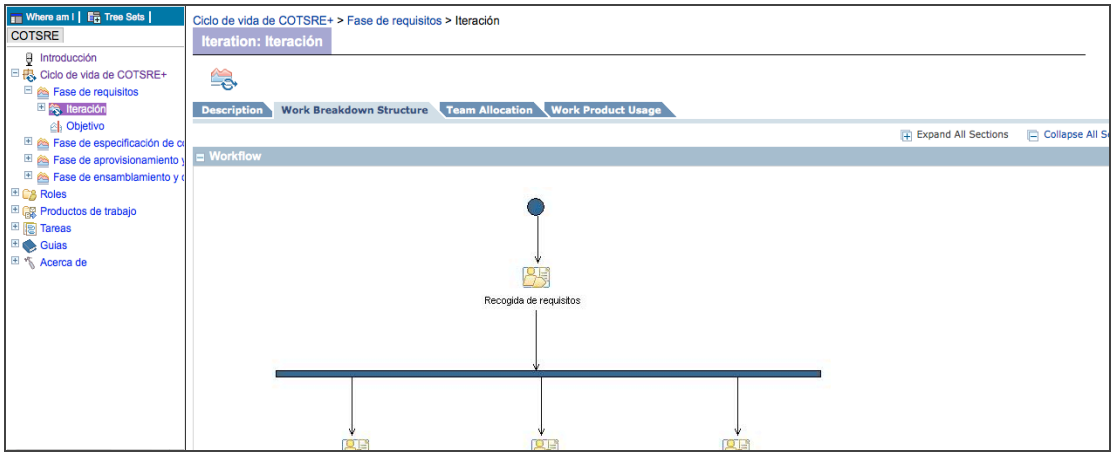


Figura 38. Vista Iteración. Pestaña Work Breakdown Structure 1

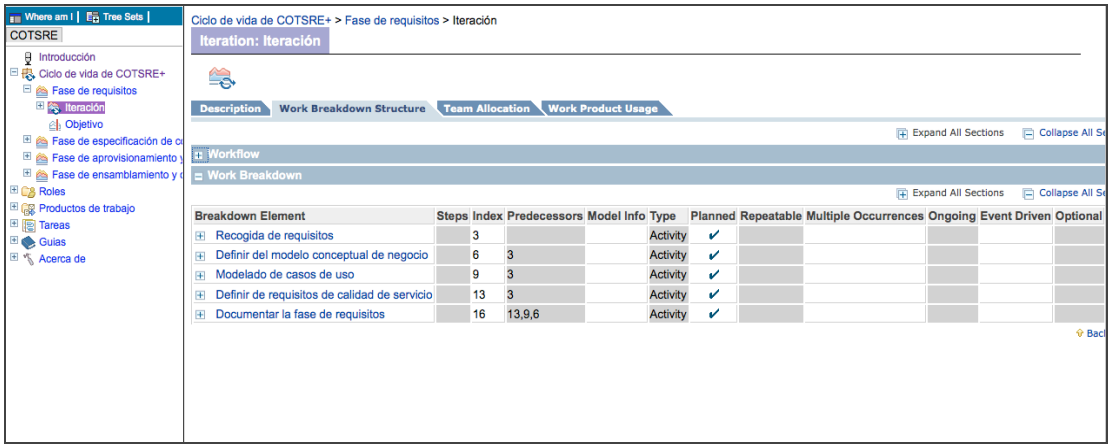


Figura 39. Vista Iteración. Pestaña Work Breakdown Structure 2

Para cada actividad aparece el diagrama de actividad además de la descripción, productos de trabajo y roles que interactúan con ella. En las dos siguientes figuras podemos ver el diagrama de actividad y un desglose con las tareas y roles y productos de trabajo que participan en cada tarea.

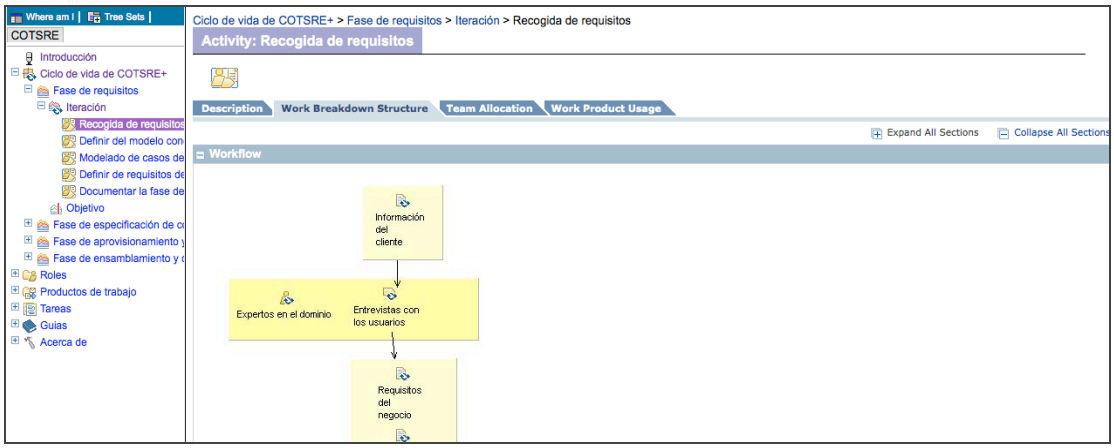


Figura 40. Vista Fase. Pestaña Work Breakdown Structure 1

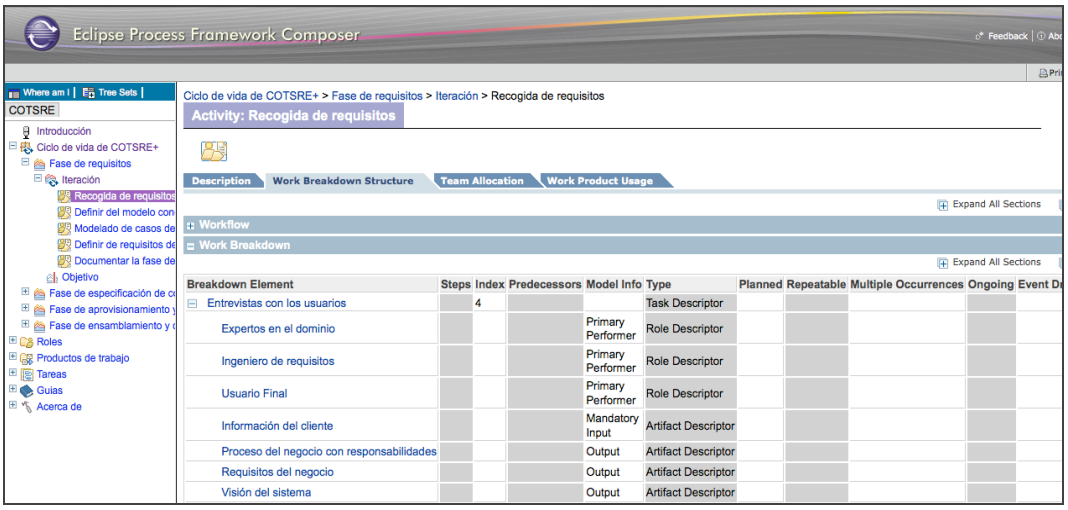


Figura 41. Vista Fase. Pestaña Work Breakdown Structure 2

En el Objetivo se muestra la descripción del objetivo de la fase a la cual pertenezca.

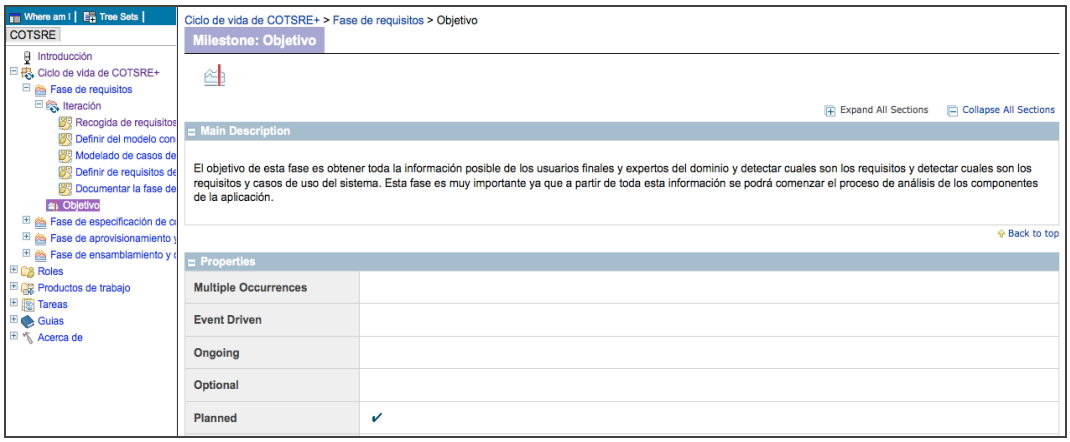


Figura 42. Vista Objetivo

7.3 Roles

En *Roles* aparece el conjunto de roles (*Role Sets*) creados.

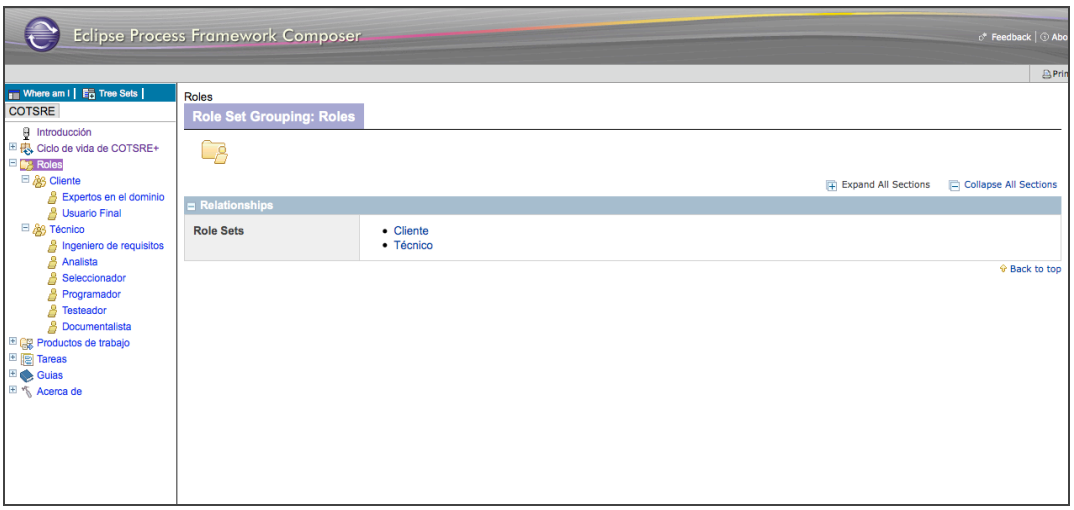


Figura 43. Vista conjunto de roles

Por cada conjunto se muestra el listado de roles que pertenecen a él.

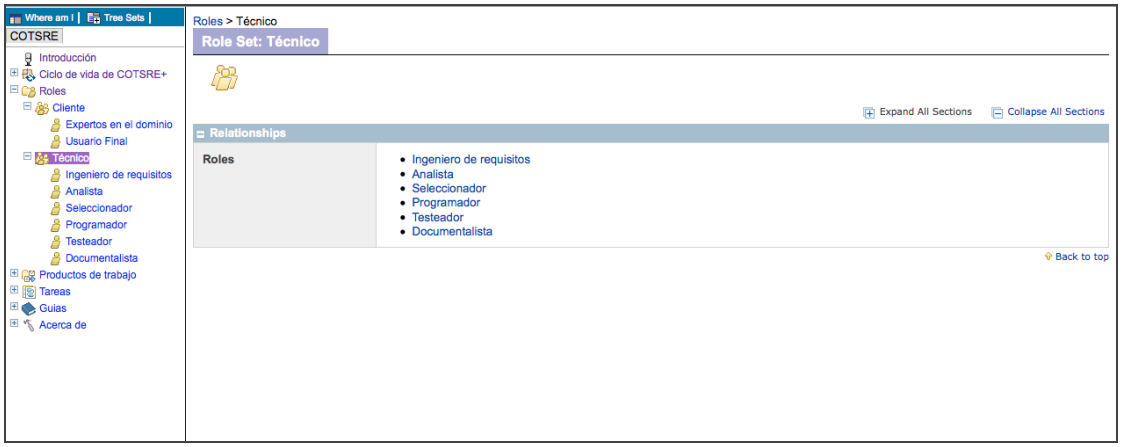


Figura 44. Vista listado de roles

Por cada rol muestra un esquema con las tareas con las que interactúa además de un listado con los productos de trabajo que modifica y una descripción.

Figura 45. Vista rol

7.4 Productos de trabajo

En *Productos de trabajo* podemos ver los dominios creados.

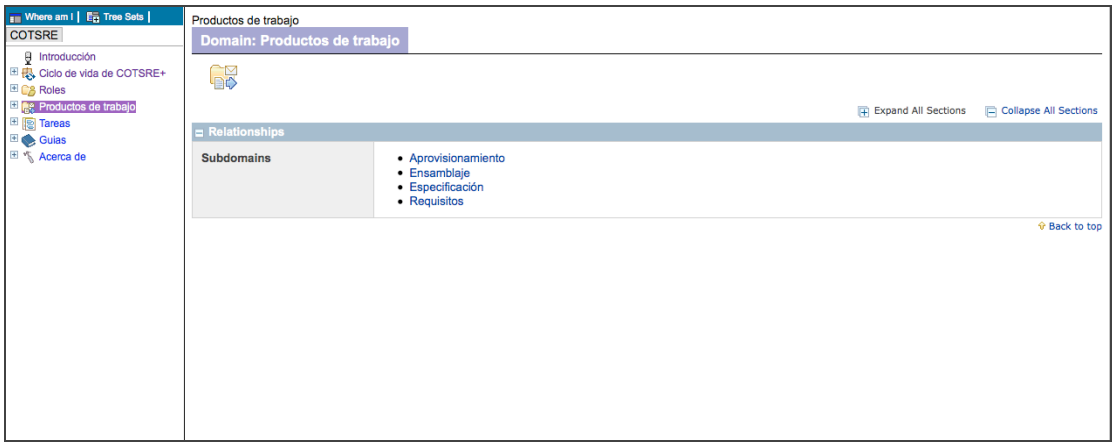


Figura 46. Vista dominio de productos de trabajo

En cada dominio vemos el listado de productos de trabajo que pertenecen a él.

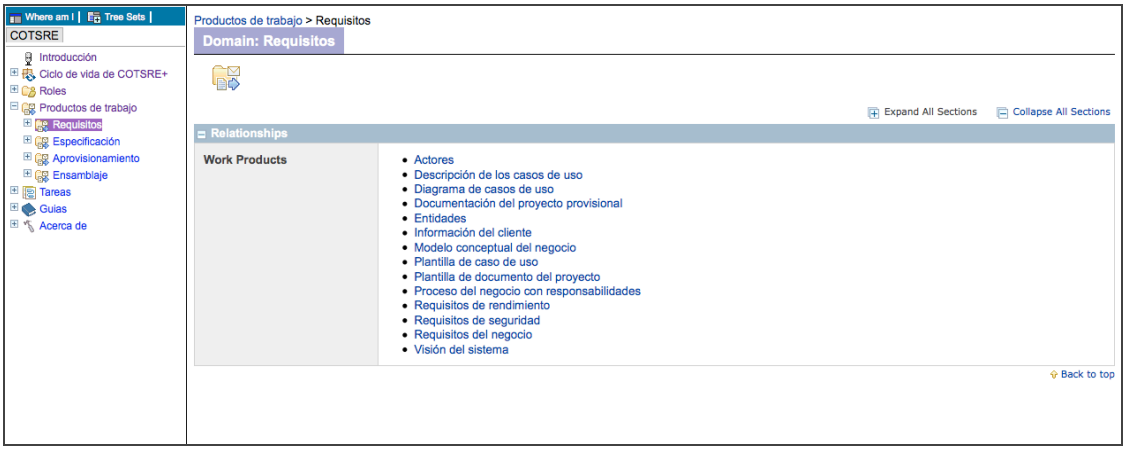


Figura 47. Vista listado de productos de trabajo de un dominio

Por cada producto de trabajo podemos ver las tareas (*Task*) en las que interviene como entrada (*Input to*) o como salida (*Output from*).

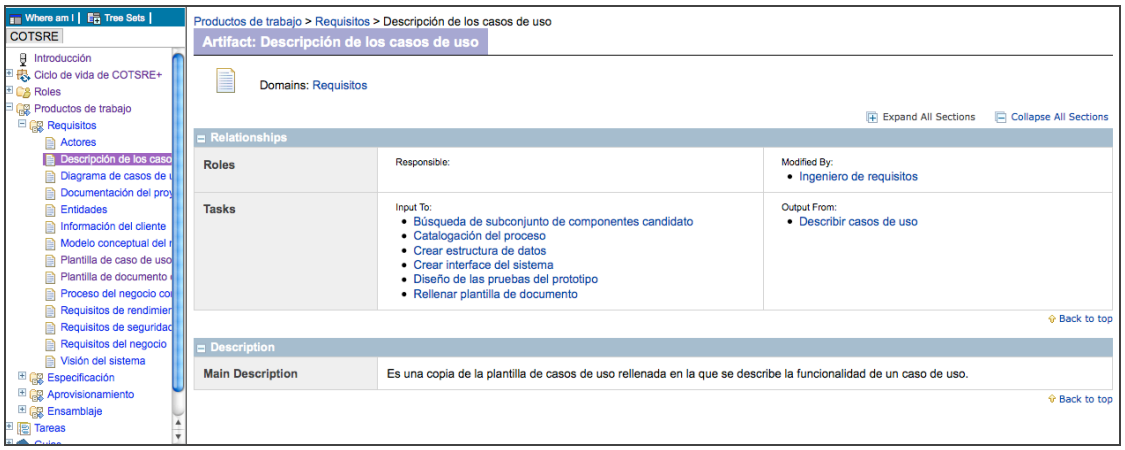


Figura 48. Vista Producto de Trabajo

7.5 Tareas

En *Tareas* tenemos las tareas ordenadas por categorías. Aquí se nos muestra el índice de disciplinas.

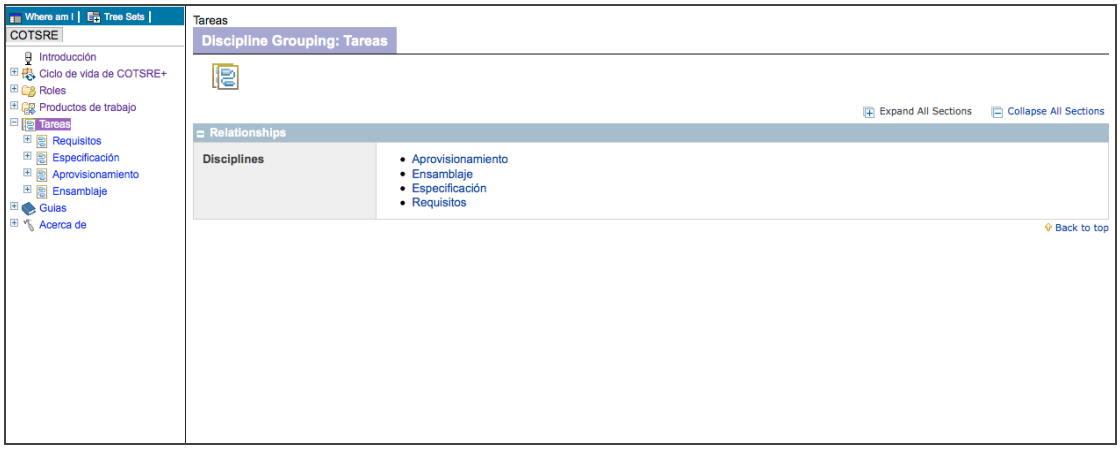


Figura 49. Vista Disciplinas

Dentro de cada disciplina podemos ver el listado con todas las tareas pertenecientes a ella.

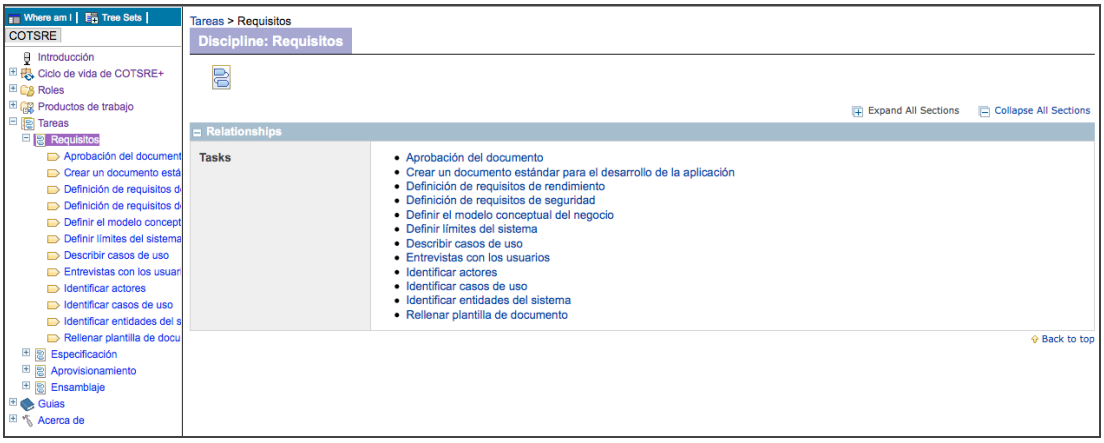


Figura 50. Vista tareas de una disciplina

Por cada tarea se nos muestra información sobre qué roles interaccionan con ella, los productos de trabajo de entrada (*Input*) y salida (*Outputs*) y su descripción (*Main Description*).

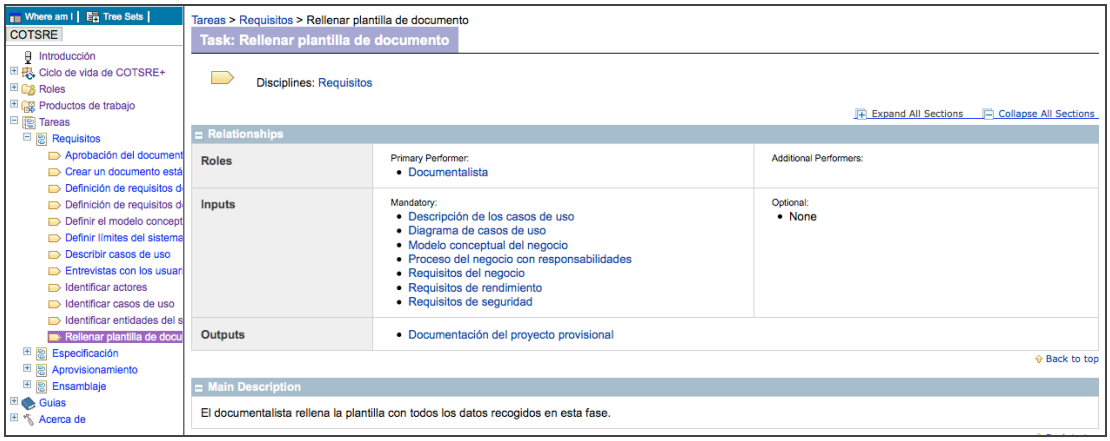


Figura 51. Vista Tarea 1

Además cuando se tratan de tareas más complejas su ejecución puede estar dividida en pasos (*Steps*).

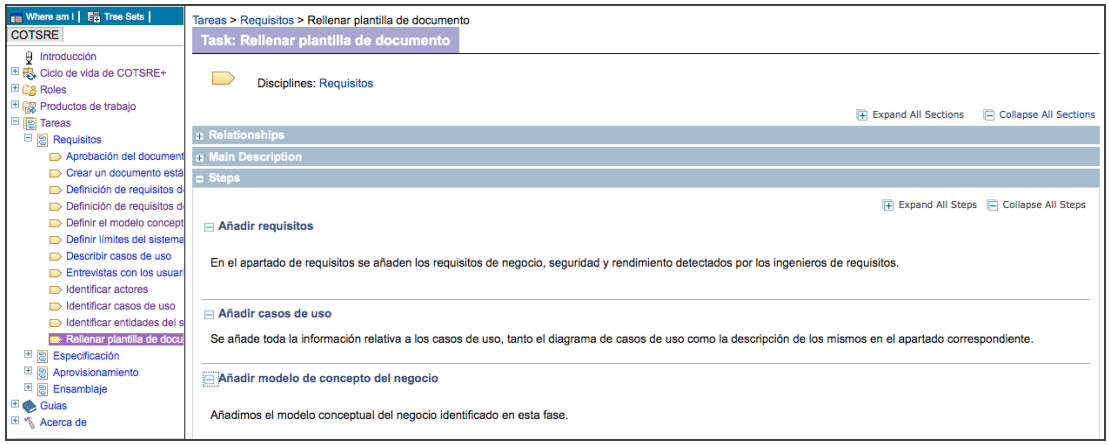


Figura 52. Vista Tarea 2

7.6 Guías

En nuestro caso las únicas guías que tenemos son las plantillas. En el menú de plantillas nos encontramos su índice de contenidos que son las dos plantillas que se han creado para COTSRE+.

Figura 53. Vista Listado Plantillas

Dentro de cada plantilla podemos ver el producto de trabajo al cual se asocia y el fichero adjunto de la plantilla para descargar.

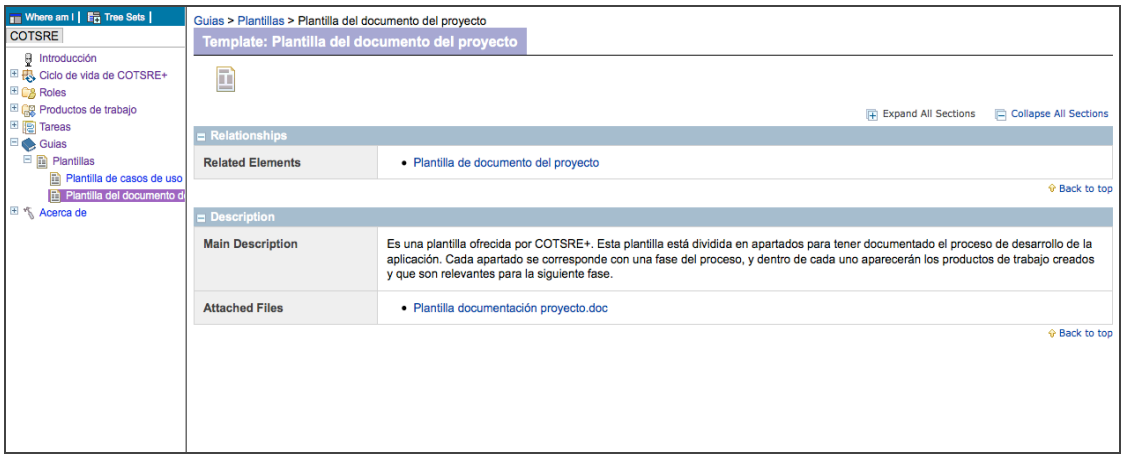


Figura 54. Vista Plantilla

7.7 Acerca de

Muestra la información de la autora del proceso.

Figura 55. Vista Acerca De

8. CotsreApp: Selección de componentes de COTSRE+

Se ha desarrollado una aplicación de escritorio que automatiza la selección de componentes explicada en el método COTSRE+, llamada CotsreApp. Esta aplicación nos permite introducir requisitos, características, casos de uso relacionados con una serie de componentes. Al insertar un componente se introduce una información detallada de él y se da una puntuación a las características que cumple según su grado de cumplimiento, además se le añadirán sus requisitos y el caso de uso que cumple. A partir de esto la aplicación ayudará en la selección de un componente adecuado a partir de requisitos y casos de uso según hemos explicado en el método.

Antes de ejecutar la aplicación se debe crear una carpeta llamada "COTSRE_BBDD" donde se guardarán los ficheros de base de datos. Si se ejecuta sobre un sistema operativo *Mac OS X Leopard* el directorio deberá crearse en "/Users/nombre_usuario", si es *Windows XP* o superior en "C:/Documents and Settings/nombre_usuario". Para ejecutar la aplicación se utilizará el fichero COTSREAPP.jar. Al abrir la aplicación se comprueba si la base de datos está vacía y en ese caso se generan unos datos de prueba para comenzar a probar la aplicación.

8.1 Manual de Usuario

Cuando entramos en la aplicación tenemos un menú para ir realizando las operaciones.

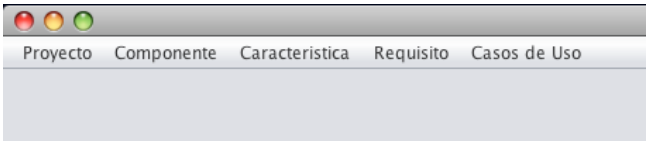


Figura 56. Menú CotsreApp

8.1.1 Casos de Uso

En el menú de "Casos de Uso" al hacer clic se nos despliega un submenú con 3 opciones, añadir un actor, un caso de uso o hacer búsquedas de casos de uso.

Figura 57. Submenú de Casos de Uso

Nuevo Actor

En este formulario se añade el nombre y la descripción del actor que se desea guardar.

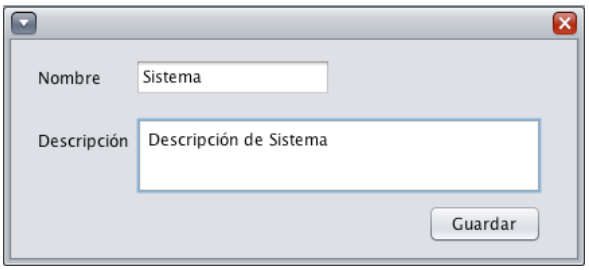


Figura 58. Nuevo Actor

Una vez introducida se hace clic en el botón "Guardar" y aparecerá un mensaje informando que se ha guardado o si por el contrario el actor ya existe en la base de datos.

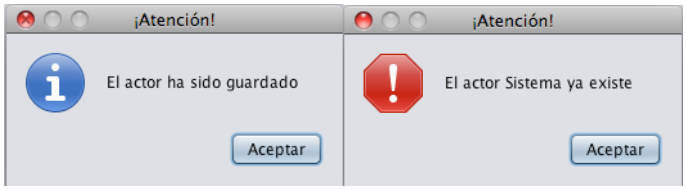


Figura 59. Mensajes Nuevo Actor

Nuevo Caso de Uso

En el formulario se da la opción de añadir el nombre, objetivo, actores que intervienen y los pasos necesarios.

Para introducir los actores, abrimos el desplegable con los actores introducidos en la base de datos, seleccionando los que queramos haciendo clic en el botón "Insertar". Debajo aparecerá la lista de los actores seleccionados.

Para introducir los pasos del guión del caso de uso escribiremos la descripción de cada uno y haremos clic en el botón "Añadir", debajo también aparecerá el listado de todos los pasos introducidos.

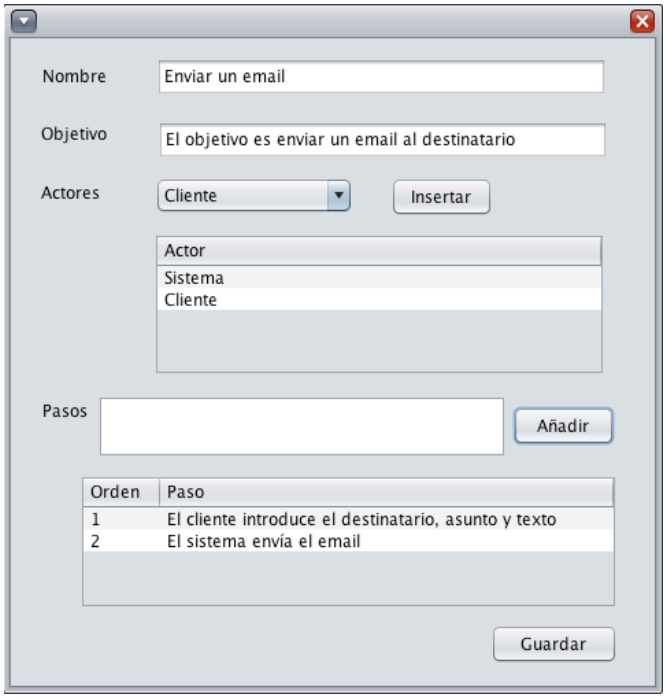


Figura 60. Nuevo Caso de Uso

Una vez terminado de introducir toda la información del caso de uso que se desea insertar se hace clic en "Guardar" apareciendo un mensaje que informa de que se ha guardado correctamente.

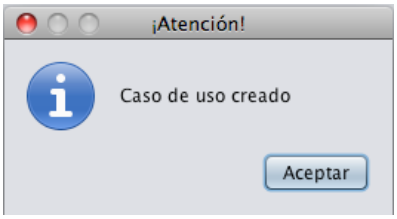


Figura 61. Mensaje Caso de Uso

Buscar Caso de Uso

Este formulario es utilizado para visualizar todos los casos de uso introducidos y realizar operaciones de borrado, modificación y vista de su información. Podemos filtrar la lista de casos de uso por objetivo y/o actores. Se selecciona un objetivo y se hace clic en filtrar, apareciendo en la lista los casos de uso que cumplen dicho objetivo. Con los actores se procede de forma parecida, se seleccionan los actores del despegable y se hace clic en "Añadir" o en "Quitar" para filtrar la búsqueda.

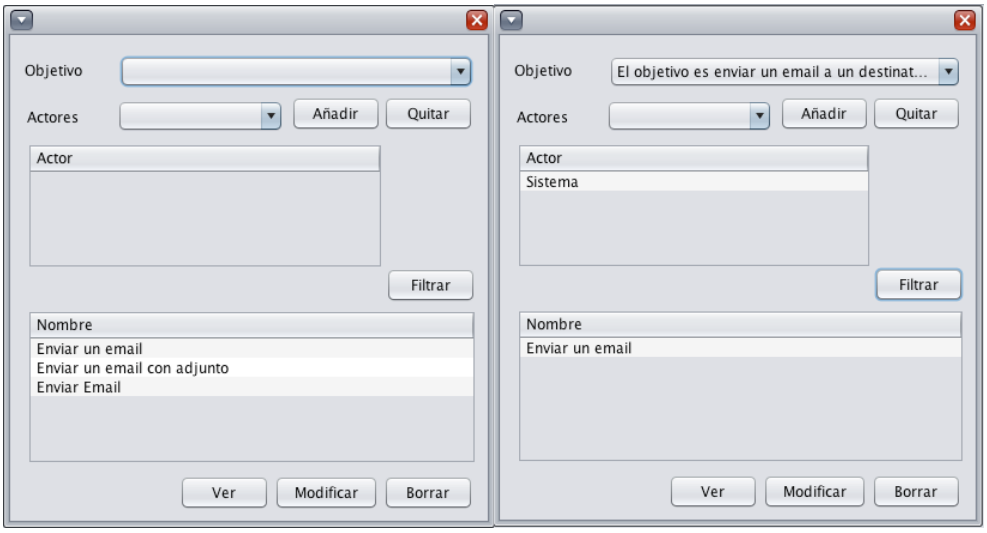


Figura 62. Buscar Caso de Uso

Si selecciona un requisito de la lista podemos ver su información, modificarlo o borrarlo. Si hacemos clic en "Ver", "Modificar" o "Borrar" sin haber seleccionado ninguno nos aparece un mensaje informativo.

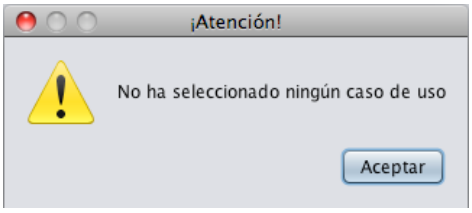


Figura 63. Mensaje Buscar Caso de Uso

Al hacer clic en "Ver" aparece una ventana con la información no editable del caso de uso.

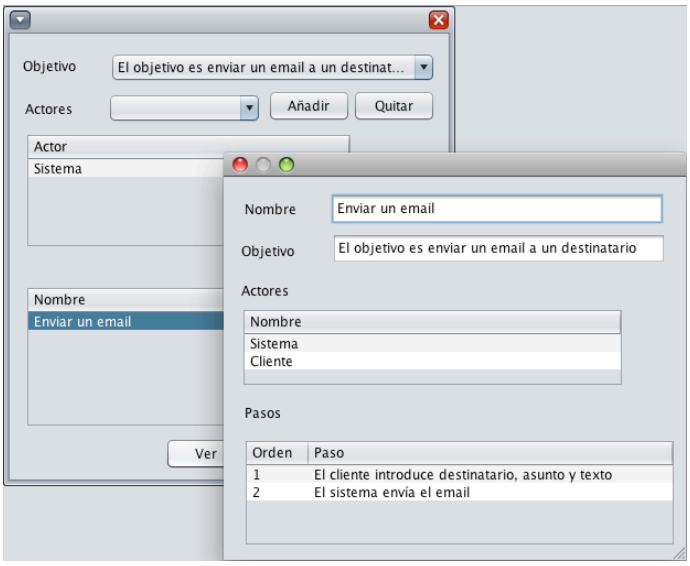


Figura 64. Ver caso de uso

Al borrar aparecerá un mensaje informativo informando del éxito o no de la operación. Un caso de uso podrá borrarse cuando no esté asociado a ningún componente o no haya sido utilizado en un proyecto COTSRE+.

Figura 65. Mensaje de borrado de caso de uso

Al modificar volverá a aparecer la ventana de inserción de casos de uso explicada antes con todos los campos rellenos del caso de uso seleccionado.

8.1.2 Requisitos

El menú "Requisito" ofrece dos opciones, añadir uno nuevo o buscar requisitos.

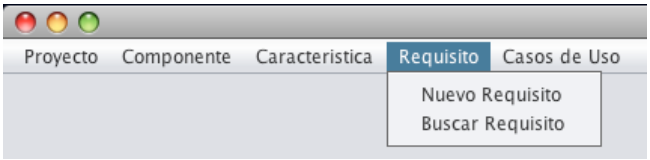


Figura 66. Submenú Requisito

Nuevo Requisito

La información que podemos guardar del requisito es su nombre, catálogo al que pertenece y el tipo (funcional o no funcional). Rellenados los datos se hace clic en "Guardar".

Figura 67. Nuevo Requisito

Una vez guardado el requisito aparecerá un mensaje informativo.

Figura 68. Requisito guardado

Buscar Requisito

En esta ventana aparece un formulario con todos los requisitos de la base de datos, esta lista puede filtrarse por el tipo de requisito. Los requisitos pueden ser seleccionados para visualizar su información, modificarlos y borrarlos.

Figura 69. Buscar Requisito

Si hacemos clic en “Ver”, “Modificar” o “Borrar” sin seleccionar un requisito nos aparece un mensaje.

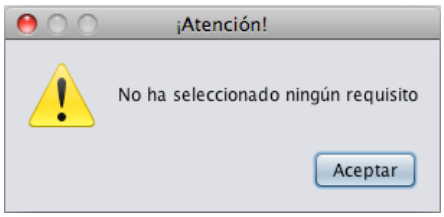


Figura 70. Mensaje de requisito no seleccionado

Al hacer clic en "Ver" aparece una ventana con los datos no editables del requisito seleccionado. Además de su nombre, catálogo al que pertenece y tipo se puede visualizar un listado de componentes que cumplen con este requisito.

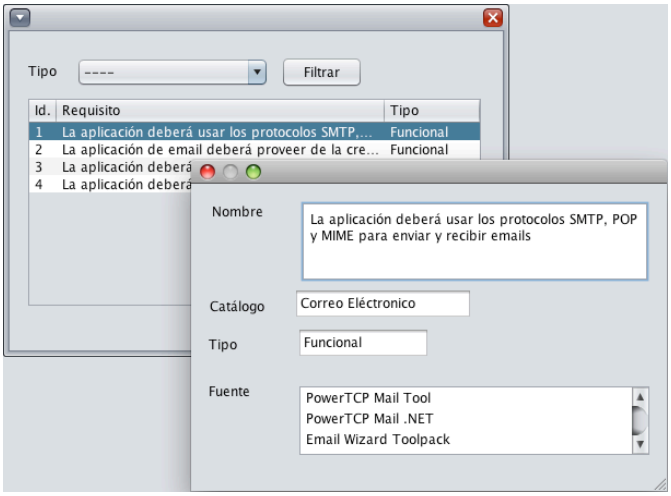


Figura 71. Ver Requisito

Al hacer clic en modificar aparecerá una pantalla igual que la de insertar requisitos con los campos rellenos con la información del requisito seleccionado.

Al hacer clic en borrar se borrará el requisito siempre que no esté asociado a un componente o haya sido utilizado por un proyecto COTSRE+ y aparecerá un mensaje informativo en caso de éxito.

Figura 72. Mensaje informativo requisito guardado

8.1.3 Características

En el menú "Característica" tenemos la opción de insertar una nueva.

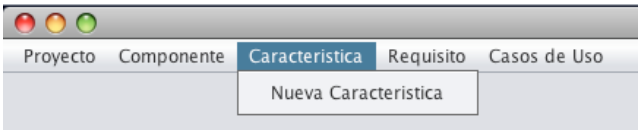


Figura 73. Submenú Característica

Nueva Característica

Se abre un formulario muy simple en el que vemos el listado de características y un cuadro donde insertar el nombre.

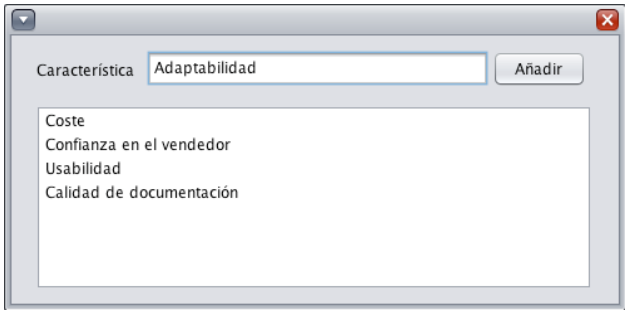


Figura 74. Nueva Característica 1

Escribimos su nombre y le damos a "Añadir", debajo podremos ver la característica insertada más el listado total de características.

Figura 75. Nueva Característica 2

8.1.4 Componentes

En el menú "Componente" tenemos las opciones de insertar y buscar componentes.

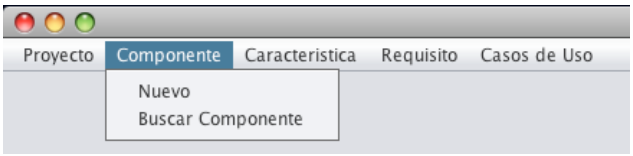


Figura 76. Submenú Componente

Nuevo

Al darla a Nuevo nos aparecerá un formulario con tres pestañas "Componente", "Características" y "Requisitos". En la pestaña "Componente" introducimos la información del componente, tal como su nombre, una pequeña descripción, autor, fecha de creación, versión, el lenguaje en el que se haya desarrollado, la localización del mismo y el caso de uso que cumple. Para seleccionar un caso de uso de manera más cómoda se hace clic en "Buscar" y aparecerá el buscador de casos de uso explicado anteriormente.

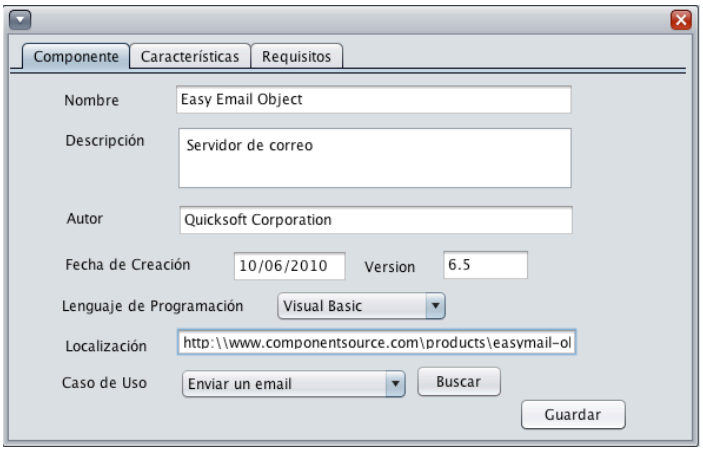


Figura 77. Nuevo Componente. Pestaña Componente

Si hacemos clic en "Guardar" se creará el componente en la base de datos y podremos insertar sus características y requisitos.

Figura 78. Mensaje informativo componente guardado

En la pestaña "Características" añadimos las características del componente junto con su grado de cumplimiento Conforme añadamos una característica veremos que nos aparece arriba en el listado de características.

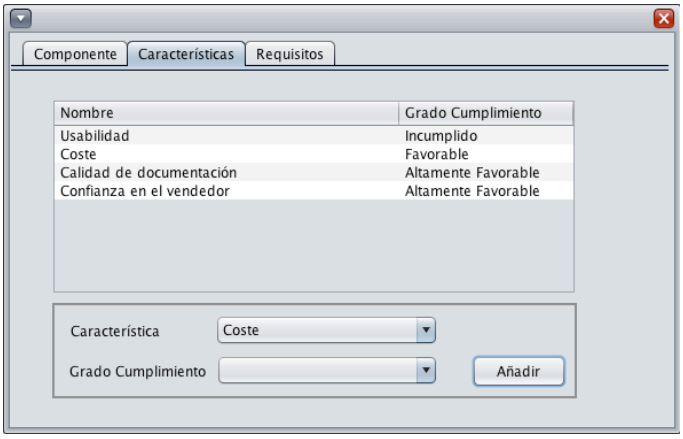


Figura 79. Nuevo Componente. Pestaña Características

En la pestaña "Requisitos" añadimos los requisitos que cumpla el componente que hayamos añadido. Podemos insertar uno nuevo o buscar dentro de nuestro catalogo de requisitos.

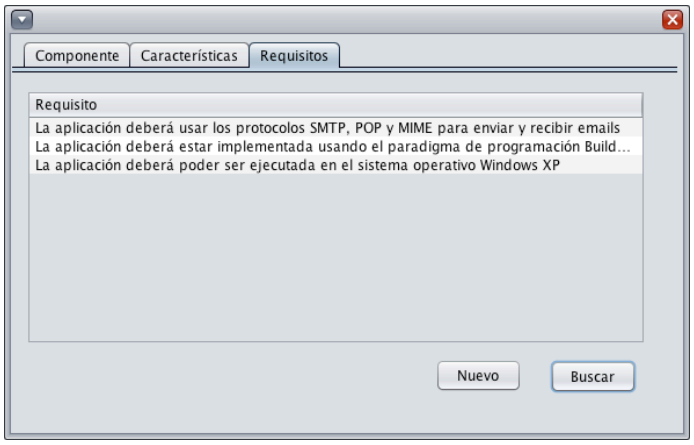


Figura 80. Nuevo Componente. Pestaña Requisitos

Si decidimos darle a nuevo veremos un formulario igual al de insertar requisito explicado antes.

Si seleccionamos buscar, veremos una pantalla con los requisitos introducidos. Podemos filtrar por requisitos funcionales y no funcionales para ayudar en la búsqueda.

Buscar Componentes

En esta ventana aparece una ventana con los componentes existentes en la base de datos. Se puede visualizar, modificar y borrar los componentes que se seleccionen de la lista.

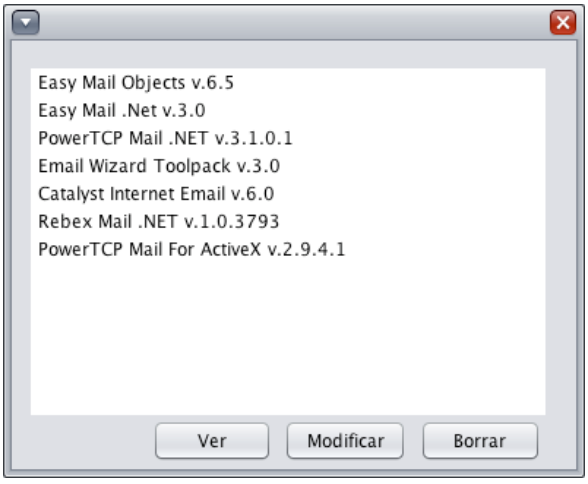


Figura 81. Buscar Componentes

Si se hace clic en algunos botones sin haber seleccionado un componente aparecerá el siguiente aviso.

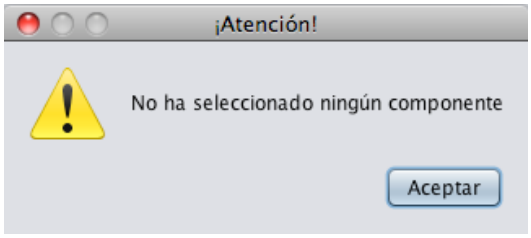


Figura 82. Mensaje no selección

Al hacer clic en “Ver” aparece una ventana con la información no editable del componente dividida en 3 pestañas: “Componente”, “Características” y “Requisitos”.

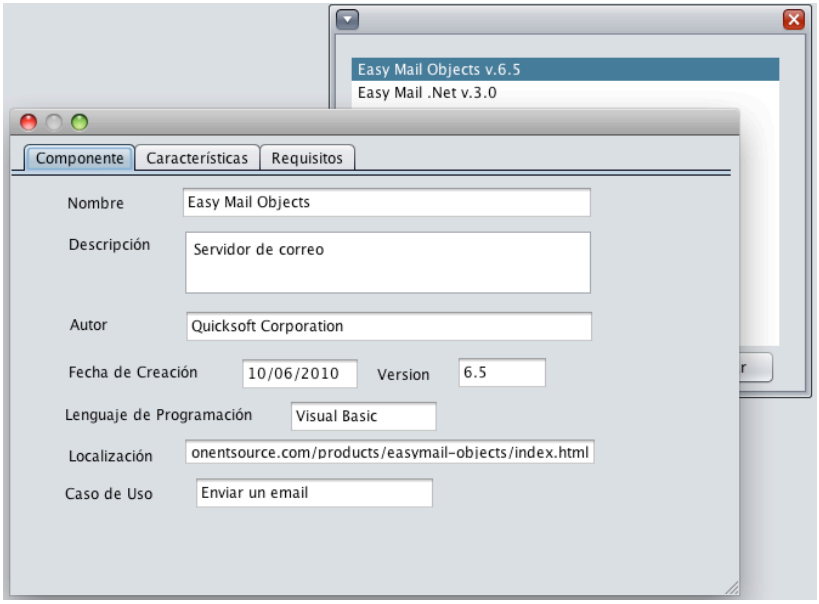


Figura 83. Ver Componente

Al hacer clic en “Modificar” aparece el formulario de inserción de componentes con los datos del componente seleccionado para modificarlos.

Al hacer clic en “Borrar” se borrará el componente si este no está asociado a ningún proyecto COTSRE+. Al borrar el componente se borrarán los valores de sus características pero no sus requisitos y casos de uso, ya que estos permanecen en catalogados en la base de datos para su reutilización en otros componente.

8.1.5 COTSRE+

En el menú “Proyecto” tenemos los submenús de “COTSRE+”, “Buscar Proyecto” y “Salir”. Para explicar el submenú de COTSRE+ que es el más importante de la aplicación lo haremos a través de 4 ejemplos de uso en el siguiente apartado.

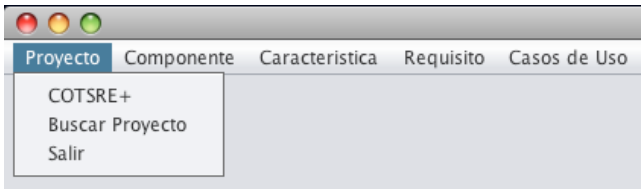


Figura 84. Submenú Proyecto

COTSRE+

En este submenú se automatiza la selección de componentes del método COTSRE+. Se seleccionan una serie de requisitos y/o casos de uso y en caso de no encontrar el componente adecuado se le añaden también unas características deseables que debería tener el componente. Al final del proceso se detalla información del componente ganador y se guarda la información del proceso con un nombre.

En la primera ventana se pueden elegir los requisitos y/o caso de uso deseables que sean cumplidos. Al hacer clic en botón “Añadir” nos visualiza el buscador de

requisitos explicado antes y se mostrarán en la tabla de "Requisitos seleccionados". Seleccionando alguno y haciendo clic en "Quitar" podremos quitarlos de la selección.

Para seleccionar el caso de uso simplemente desplegaremos el combo y seleccionaremos el deseado. También haciendo clic en "Buscar" se mostrará el buscador de casos de uso para seleccionar el caso de uso de una manera más cómoda.

En la "Matriz de selección" se mostrarán todos los componentes que cumplen con alguno de los requisitos o el caso de uso.

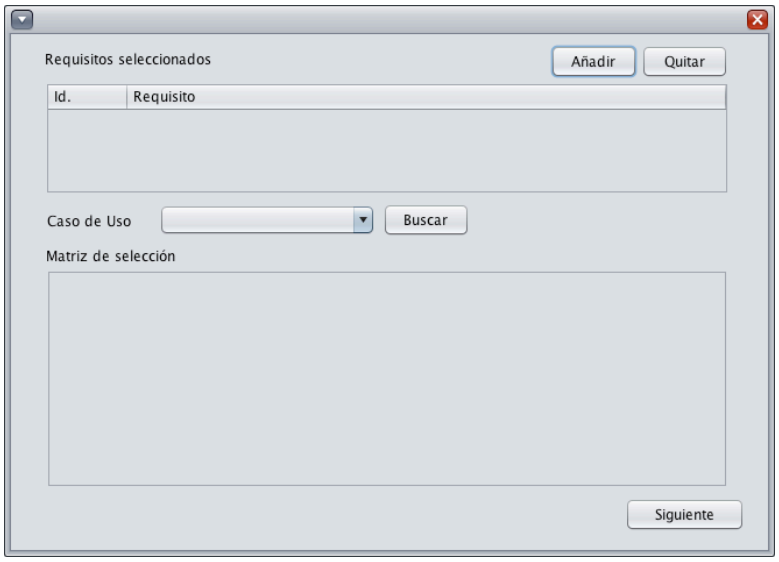


Figura 85. COTSRE+ Vacío

Haciendo clic en siguiente nos informará de si se ha encontrado un componente adecuado o si por el contrario existe empate o ninguno cumple con todos los parámetros introducidos por lo que se continuará con el proceso. Si hiciéramos clic en "Siguiente" sin haber seleccionado ningún requisito o caso de uso no pasaríamos a la siguiente etapa y nos informaría de ello.

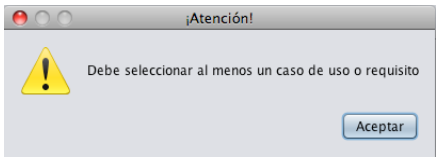


Figura 86. Mensaje no selección de requisito y/o caso de uso

En el siguiente apartado se explica su funcionamiento con una serie de ejemplos significativos.

Buscar Proyecto

En esta ventana aparece un listado el nombre y fecha de todos los proyectos creados hasta la fecha.

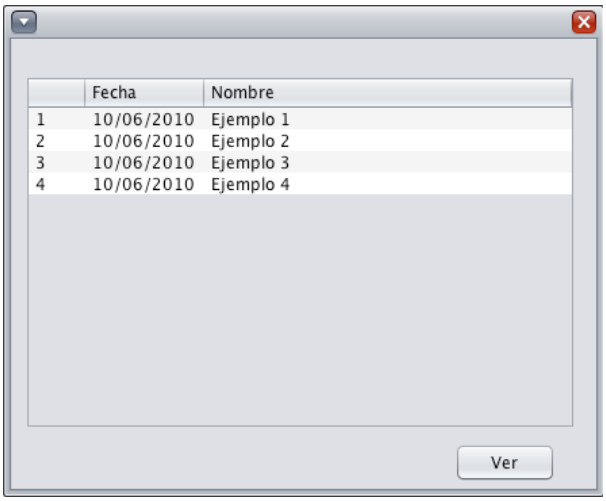


Figura 87. Buscar Proyecto

Si seleccionamos un proyecto podemos hacer clic en "Ver" para visualizar su descripción. En la ventana que aparece podemos ver la fecha y nombre del proyecto, los parámetros introducidos para seleccionar el componente como son el listado de requisitos y/o el caso de uso que debería cumplir y finalmente el componente seleccionado es el componente más adecuado según los parámetros introducidos.

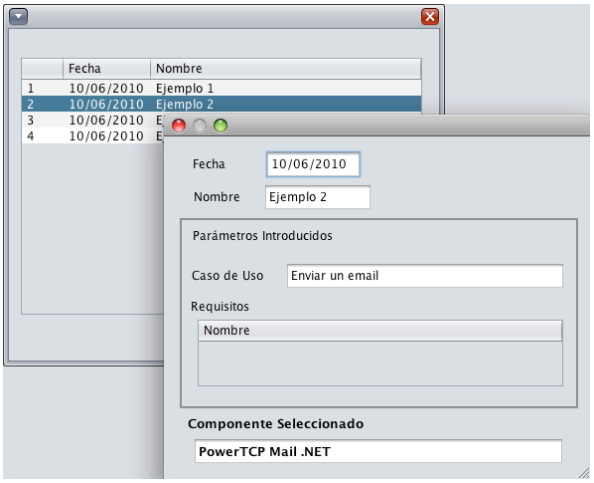


Figura 88. Ver Proyecto

Salir

Finalmente haciendo clic en el submenú "Salir" se cerrará la aplicación.

8.2 Ejemplos de selección de componentes

A continuación se detallan una serie de ejemplos de ejecución del método de selección de COTSRE+ que intentan ser lo más representativos posibles.

8.2.1 Ejemplo 1

Seleccionamos 4 requisitos de la base de datos y ningún caso de uso. En la matriz de selección se produce un empate entre dos componentes que cumplen con

todos los requisitos introducidos, estos son *Catalyst Internet Email v6.0* y *Email Wizard Toolpack v3.0*.

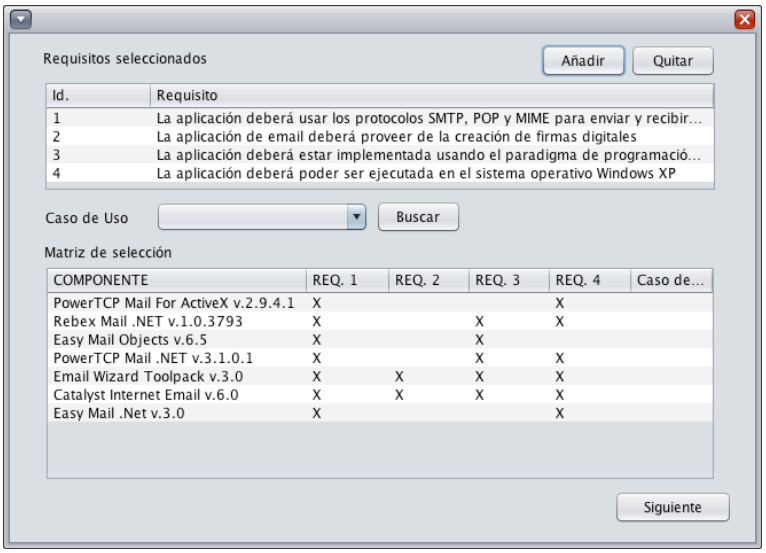


Figura 89. Ejemplo 1. Matriz de Selección

Al hacer clic en "Siguiente" aparece un mensaje avisando que se ha producido un empate entre varios componentes.

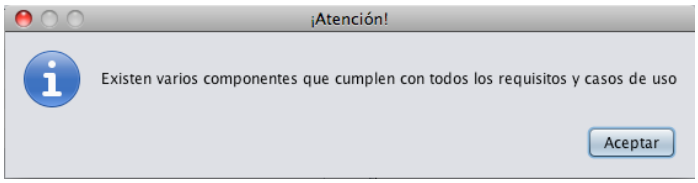


Figura 90. Mensaje varios candidatos

Para desempatar en la siguiente fase se introducen características y una puntuación para cada una. Cada vez que se introduzca una característica y su grado de satisfacción, se actualiza la matriz de decisión y su visualiza la puntuación total de cada uno seleccionándose el de mayor puntuación.

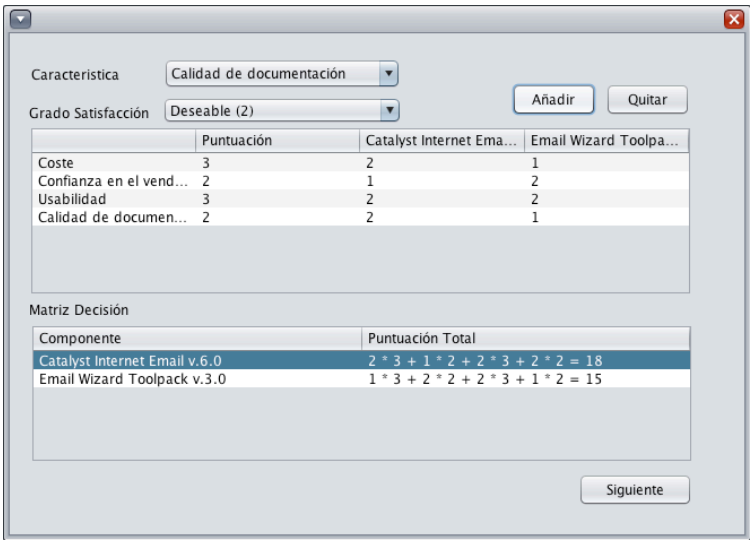


Figura 91. Ejemplo 1. Matriz de Decisión

Una vez se introducen las características con su puntuación deseable se hace clic en “Siguiente” para que se muestre la información del componente seleccionado.

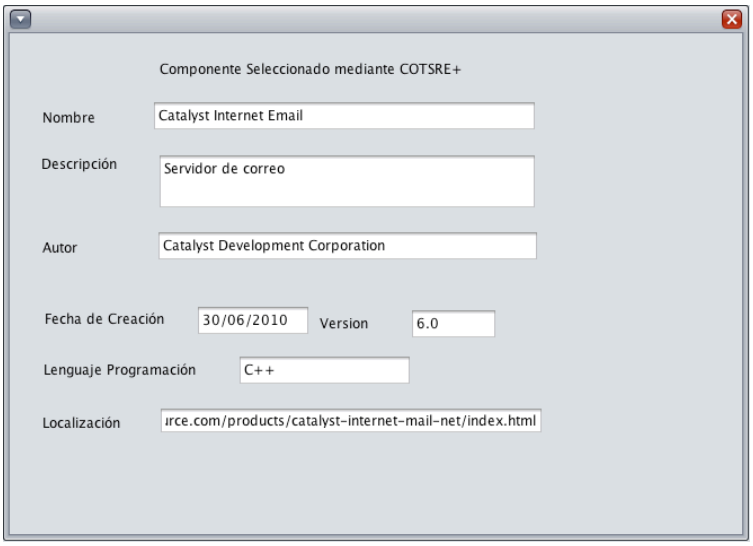


Figura 92. Ejemplo 1. Componente Seleccionado

Al cerrar esta ventana se pide introducir el nombre con el que el proyecto COTSRE+ será guardado en la base de datos.

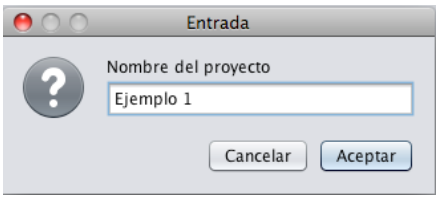


Figura 93. Guardar Proyecto

8.2.2 Ejemplo 2

Para el ejemplo no seleccionaremos ningún requisito pero sí un caso de uso. En la matriz de selección aparecerán todos los componentes que cumplan con este caso de uso.

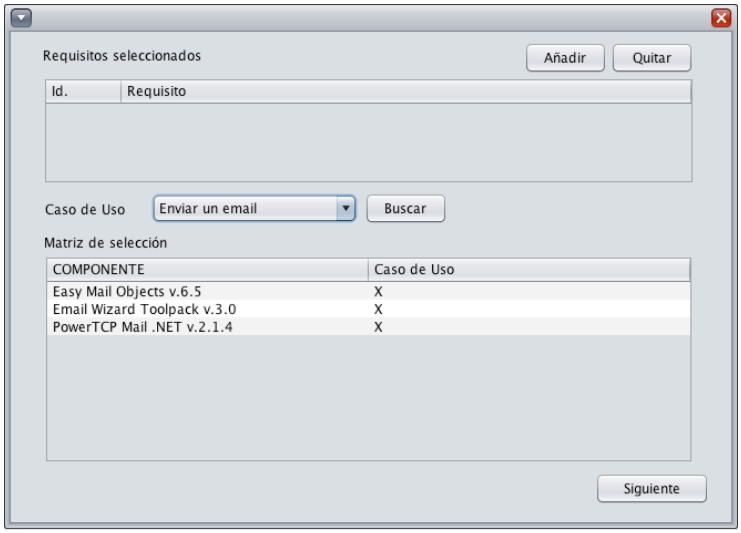


Figura 94. Ejemplo 2. Matriz de Selección

Como se observa son tres los componentes que cumplen con el caso de uso elegido por lo que se mostrará un mensaje informativo y se procederá a generar la matriz de decisión a partir de las características introducidas.

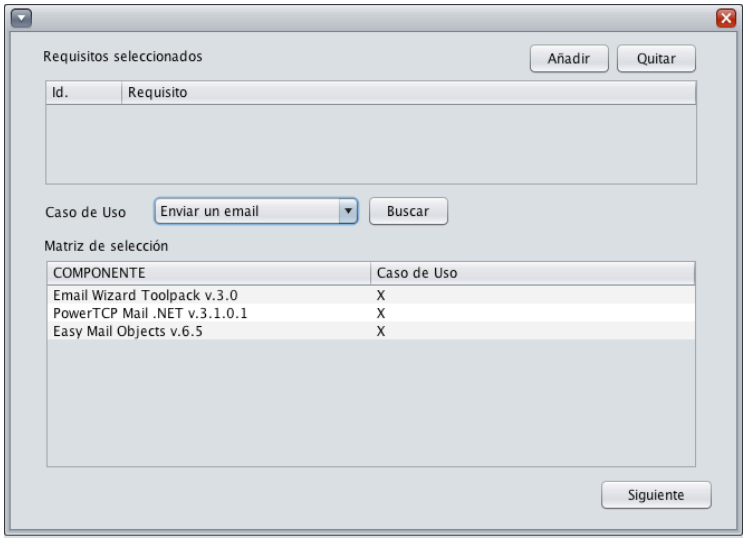


Figura 95. Ejemplo 2. Matriz de Decisión con empate de componentes

Al introducir las características y sus puntuaciones se produce un empate entre *Email Wizard Toolpack v3.0* y *PowerTCP Mail .NET v.3.1.0.1* en la matriz de decisión. Por lo que al hacer clic en "Siguiente" nos lo advertirá con un mensaje informativo y nos ofrecerá la opción de añadir o quitar características para desempatar o seleccionar directamente el componente deseado.

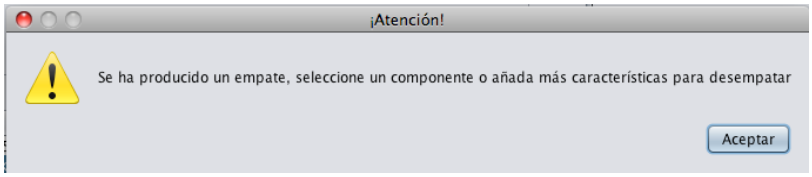


Figura 96. Mensaje empate en Matriz de Decisión

Se opta por seleccionar el primero de ellos, *PowerTCP Mail NET v3.1.0.1.*

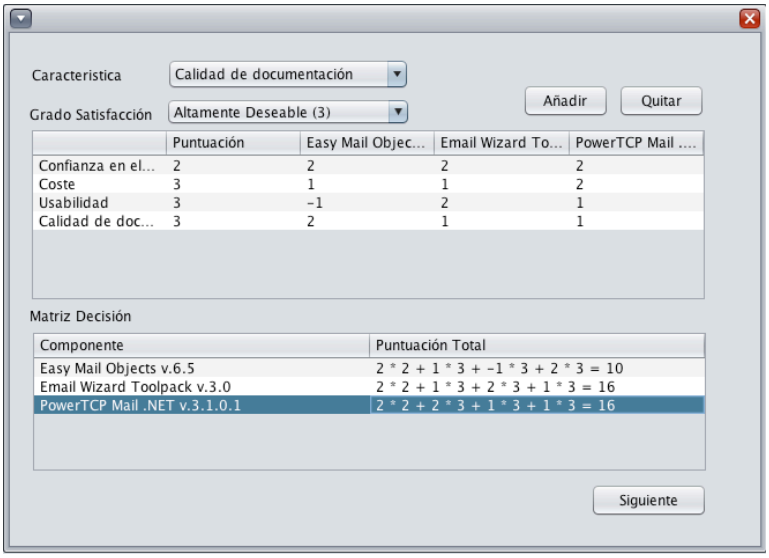


Figura 97. Ejemplo 2. Matriz de Decisión

Y la información de dicho componente será mostrado. Al cerrar la ventana se deberá introducir el nombre del proyecto COTSRE+ con el que se quiere guardar.

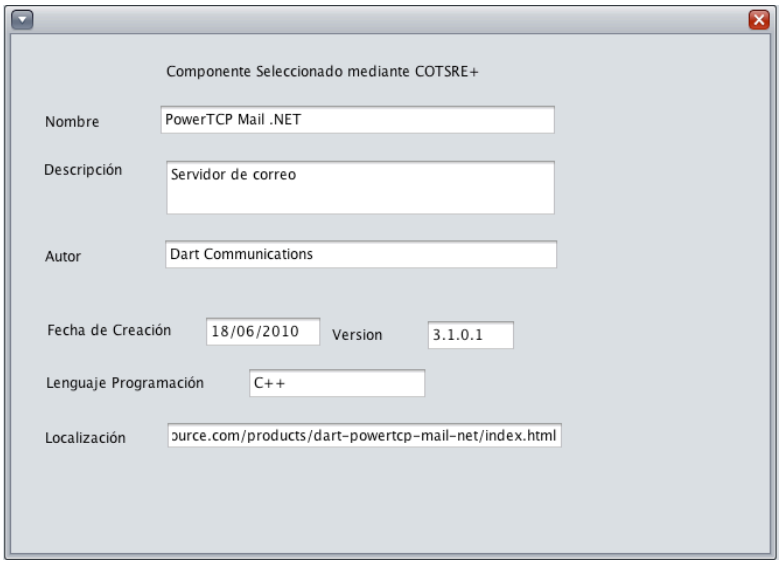


Figura 98. Ejemplo 2. Componente Seleccionado

8.2.3 Ejemplo 3

En este ejemplo se introducen cuatro requisitos y un caso de uso para seleccionar un componente.

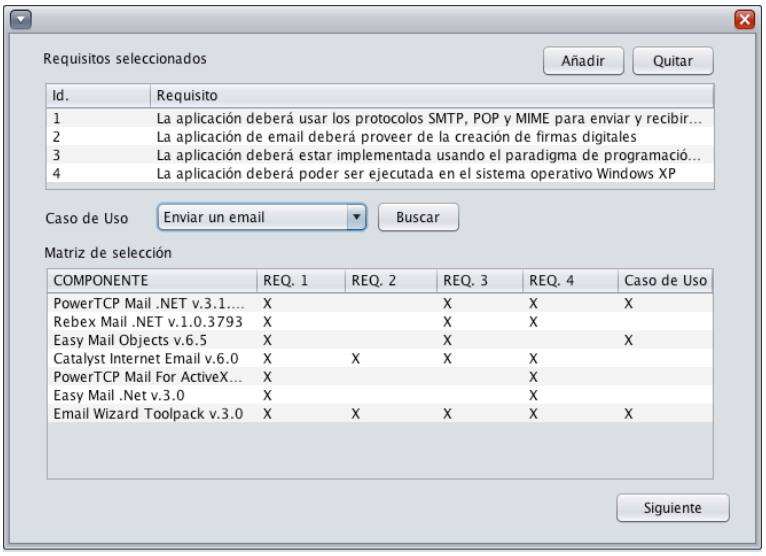


Figura 99. Ejemplo 3. Matriz de Selección

De esta manera se ha encuentra un único candidato que cumple con los requisitos y caso de uso introducidos, *Email Wizard Toolpack v.3*, y se finaliza el proceso sin que sea necesario generar una matriz de decisión. Se mostrará un mensaje informando del hecho al hacer clic en “Siguiente”.

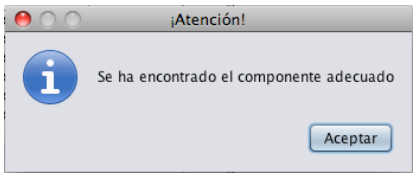


Figura 100. Mensaje componente encontrado

Y se mostrará la información relativa al componente. Al finalizar se guardará el proyecto con el nombre que se introduzca.

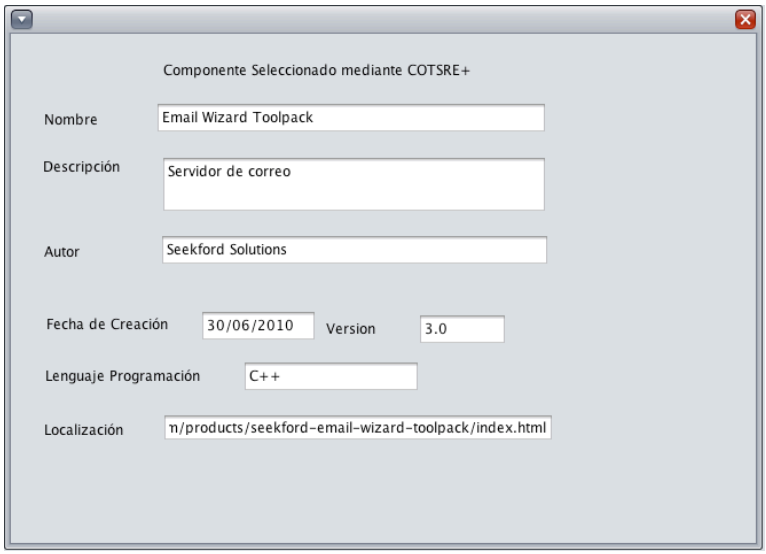


Figura 101. Ejemplo 3. Componente Seleccionado

8.2.4 Ejemplo 4

Por último en este último ejemplo seleccionaremos dos requisitos y un caso de uso, con esta configuración no existe ningún componente que cumpla esto.

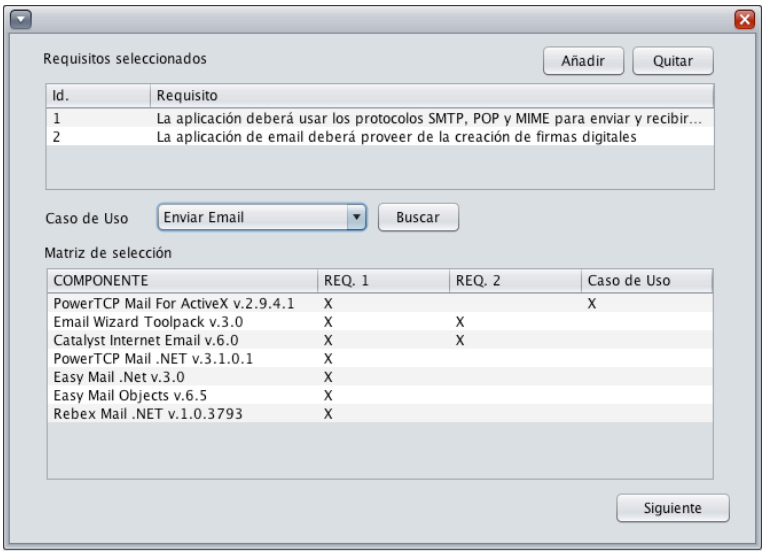


Figura 102. Ejemplo 4. Matriz de Selección

Al hacer clic en siguiente nos informa de que no hay ningún componente que cumpla con los requisitos y caso de uso introducidos. Podremos cancelar y cambiar el caso de uso o añadir o quitar requisitos para intentar encontrar un componente que

cumpla con todo. Por el contrario podremos también continuar con el proceso a partir de un conjunto de componentes que cumplan con más de la mitad de parámetros introducidos sumando requisitos y casos de uso. Además también se dará la opción de no continuar con el proceso.

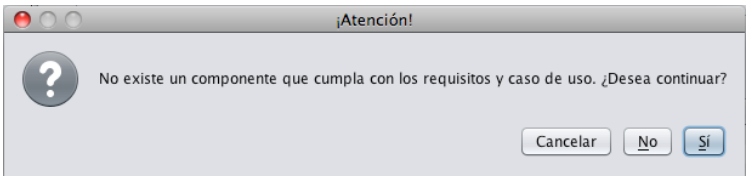


Figura 103. Mensaje no componentes.

Se elige continuar con un conjunto de componentes que cumple la mayoría de requisitos y caso de uso, quedándonos tres componentes *PowerTCP Mail For ActiveX v. 2.9.4.1* que cumple con un requisito y el caso de uso, y *Catalyst Internet Mail v.6.0* y *Email Wizard Toolpack v.3.0* que cumplen con los dos requisitos.

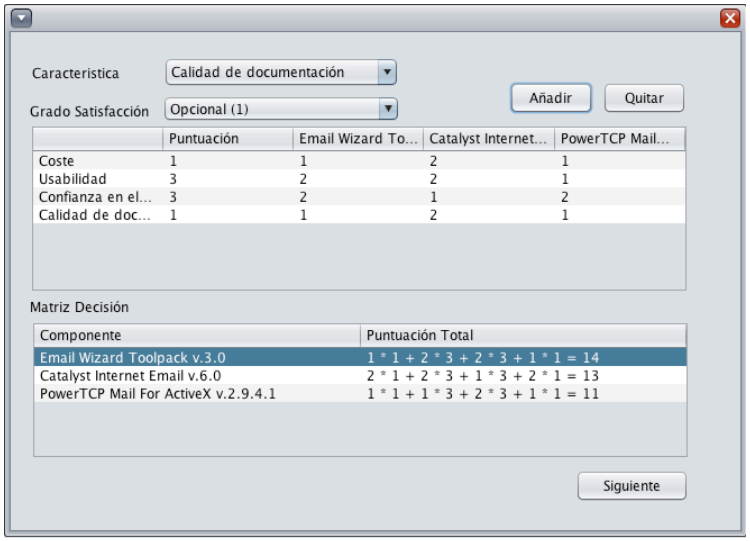


Figura 104. Ejemplo 4. Matriz de Decisión

Una vez introducidas las características y su puntuación la matriz de decisión indica que el componente más adecuado es *Email Wizard Toolpack* que además de cumplir con los dos requisitos es el que se acerca más a las características que se desea que posea.

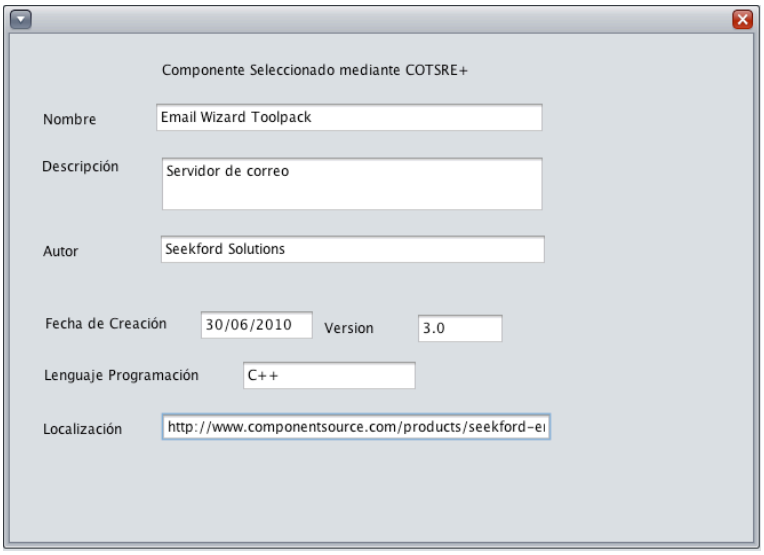


Figura 105. Ejemplo 4. Componente Seleccionado

8.3 Calidad del método

- *Calidad de la interface de usuario:* La interface para manejar el programa basado en el método de COTSRE+ es sencilla. El usuario puede introducir un componente en el repositorio añadiéndole su especificación o bien consultar de una forma sencilla que componente es más adecuado para su aplicación. El aprendizaje es sencillo ya que se ha realizado esta interface de la manera más simple posible para que esta técnica sea una ayuda al usuario y que le ayude a mejorar los tiempos de desarrollo de su aplicación.
- *Calidad del esquema de clasificación:* Los componentes están debidamente especificados por lo que su clasificación es sencilla dentro del repositorio.
- *Calidad del proceso de recuperación:* La consulta se hace mediante la elección de requisitos y casos de uso, estos dan un primer conjunto de componentes candidatos, este listado a su vez puede refinarse añadiendo características que se desea que el componente satisfaga.

9. Conclusiones y vías futuras

9.1 Conclusiones

En este proyecto hemos definido un método que creemos muy completo y que hace hincapié en la importancia de la reutilización de componentes. Partiendo de una especificación muy inicial de un método de selección de componentes, COTSRE, se ha generado un proceso que aborda todo el ciclo de vida de los componentes. Para ello nos hemos basado en la propuesta de Chessman y Daniels, el cual no especificaba claramente cómo se debía llevar a cabo la selección de componentes y hemos renombrado el método como COTSRE+.

El proceso de selección se ha completado teniendo en cuenta los casos de uso para seleccionar un componente a parte de los requisitos. Incluso tiene en cuenta que no se encuentre ningún componente que cumpla con toda la especificación, en cuyo caso es capaz de relajar la condición para tener en cuenta aquellos candidatos que no cumplen exactamente con toda la especificación.

El método de desarrollo basado en componentes y requisitos creado, además de basarse en Chessman y Daniels, se amplía especificando mejor las fases de prueba de componentes y generando una documentación que se va completando al finalizar cada fase, para obtener un histórico del proceso.

El resultado final ha sido la definición de un método que, aunque desde luego puede ser mejorable tal como veremos en el siguiente apartado, sienta unas bases de cómo debería ser un proceso completo de el ciclo de vida de un componente pensando tanto en la futura reutilización de los mismos como la posible utilización de componentes ya creados.

Somos conscientes que nuestra propuesta no deja de ser una propuesta teórica, que necesita ser aplicada y validada en situaciones reales para acabar obteniendo un método realmente práctico y útil. No obstante, es un primer paso, al que se ha dotado de la infraestructura de herramientas necesaria para mostrar también su viabilidad en este sentido.

Personalmente, este proyecto me ha servido para profundizar en temas estudiados en algunas asignaturas como Ingeniería de Requisitos, Técnicas Formales en Ingeniería del Software o Fundamentos de Ingeniería del Software. Además de tener la oportunidad de conocer la especificación SPEM. También me ha sido útil para ampliar mis conocimientos de Java y bases de datos para la realización de la herramienta CotsreApp.

9.2 Vías futuras

En relación con este proyecto existen varias vías futuras por explorar, algunas de ellas por las que se podría empezar para completar COTSRE+ serían las que se detallan a continuación.

El primer trabajo pendiente, después de la definición de COTSRE+, sería su validación con uno o varios casos de estudio (que puede ser objeto de otro PFC), como se ha dicho antes.

También vemos necesario mejorar el método de selección, actualmente éste se basa en búsquedas por requisitos y casos de uso pero esta podría tener en cuenta algún tipo de parámetro o característica. Incluso dar más prioridades a los requisitos entre sí o frente a los casos de uso, es decir, al igual que se hace para generar la matriz de decisión a partir de la introducción de unas serie de características con sus valores de satisfacción

deseable, hacer lo mismo con requisitos y casos de uso y añadirles un valor de satisfacción. Otra forma de mejorar el método de selección es mediante el uso herramientas de búsqueda y análisis de similitudes entre texto y otras formas de representación, existen ya algunas propuestas encaminadas en estas líneas de trabajo [37, 38, 39].

En tercer lugar, ampliar los catálogos de componentes reutilizables haciéndolos públicos y que se encuentren de forma distribuida de manera que cuantos más componentes se introduzcan mayores serán las posibilidades de encontrar el más adecuado.

También podría añadirse una nueva fase al finalizar el proceso de desarrollo de COTSRE+ dedicada al mantenimiento de los componentes desarrollados para complementar el ciclo de vida de un componente. Estos cambios que pudiera sufrir el componente debido a su mantenimiento deberían verse reflejados también en el catálogo de componentes a reutilizar y sus relaciones con los requisitos, casos de uso y características.

Por último, puede plantearse la integración de parte de los resultados de este proyecto en la definición de SIRENgsd un método de Ingeniería de Requisitos para Desarrollo Global, basado en SIREN, en el marco del actual proyecto PANGEA del GIS del Departamento de Informática y Sistemas.

Bibliografía y referencias

[1] John Cheesman y John Daniels. "UML Components. A Simple Process for Specifying Component-Based Software" Ed. Addison- Wesley, 2000.

[2] Miguel A. Martinez, Ambrosio Toval, Manuel F. Bertoa y Antonio Vallecillo. "Towards a Component Based and Requirements Driven Development". Handbook on Computer Software Engineering Research. Nova Science Publishers (ISBN: 978-1-60021-774-6).

[3] Miguel A. Martinez, Ambrosio Toval, "COTSRE: a ComponenTs Selection method based on Requirements Engineering", Universidad de Murcia. The 7th. IEEE Int. Conf. on Composition-Based Software. ICCBSS 2008 Student Session. February 25, 27, 2008, Madrid, Spain.

[4] Proyecto Fin de Carrera de Alfonso Olmos Marín, Septiembre de 2000,Facultad de Informática, Universidad de Murcia.

[5] Toval, J. Nicolas y B.Moros. "SIREN: Un proceso de Ingeniería de Requisitos Basado en Reutilización" I Jornadas de Ingeniería de Requisitos Aplicada. Sevilla, 2001 (JIRA 2001) 11-12 Junio 2001.

[6] A. Toval, A.Olmos, M.Piattini "Legal Requirements Reuse: A Critical Success Factor for Requirements Quality and Personal Data Protection" Proceedings of the IEEE Joint International Conference on Requirements Engineering (ICRE'02 and RE'02), pp: 9-13, Septiembre 2002, Essen Germany, IEEE Computer Press. Pp. 95-103. ISBN 0-7695-1465-0; ISSN 1090-705X.

[7] A.Toval, J.Nicolás y B.Moros, F. García. "Requirements Reuse for Improving Information Systems Security: A Practitioner's Approach". Requirements Engineering Journal vol. 6, n.4, pp. 205-219, 2001.

[8] Joaquín Lasheras, Ambrosio Toval, Joaquín Nicolás, Begoña Moros,"Soporte Automatizado a la Reutilización de Requisitos", Grupo de Investigación de Ingeniería del Software. Departamento de Informática y Sistemas. Universidad de Murcia. VIII Jornadas de Ingeniería del Software y Bases de Datos (JISBD'03), Alicante.

[9] Vanessa Hamar Tesis "Aspectos metodológicos del desarrollo y reutilización de componentes de software", Universidad de los Andes, Mérida, Venezuela, Noviembre 2003.

[10] Ian Sommerville, Ingeniería del software, Addison Wesley, 7ªEdición, 2004.

[11] Luis F. Iribarne Martínez, Tesis "Un modelo de mediación para el desarrollo de software basado en componentes COTS", Universidad de Almería, España, 14 de Julio de 2003.

[12] Bill Flowers, "Modifications to COTS", http://www.requirementsnetwork.com/. Consultado el 4 de Junio de 2010.

[13] Fredy Navarrete, Pere Botella, Xavier Franch, "Análisis de los métodos de selección de componentes COTS desde una perspectiva ágil", Universitat Politécnica de Catalunya. X Jornadas de Ingeniería del Software y Bases de Datos (JISBD 2005), Septiembre 14-16, 2005, Granada, España.

[14] Navarrete, F.; Botella, P.; Franch, X., "How Agile COTS selections methods are (and can be)?", Actas de la 30th IEEE Euromicro Conference 160-167, Septiembre 2005.

[15] Navarrete, F.; Botella, P.; Franch, X., "Reconciling Agility and Discipline in COTS Selection Processes", Proceedings 6th IEEE International Conference on COTS-based Software Systems 103-113, Alberta, Canada, Febrero 2007.

[16] Navarrete, F.; Botella, P.; Franch, X., "Defining a Scope for COTS Definition", Situational method engineering, fundamentals and experiences 298-312, Septiembre 2007.
[17] Kontio, J. "A Case Study in Applying a Systematic Method for COTS Selection", 18th Internacional Conference on Software Engineering, Springer (1996) 291-209.
[18] Kontio, J., Chen, S., Limperos, K., Tesoreiro, R., Calderia, G., Deutsh, M., "A COTS Selection Method and Experience of Its Use". Proceeding of the 20th annual Software Engineering Workshop. NASA. Greenbelt, Maryland (1995).
[19] Ncube, C., Maiden, N. "PORE: Procurement-Oriented Requirements Engineering Method for the Component-Based Systems Engineering Development Paradigm". Proceedings of the 2nd Internacional Workshop on Component Based Software Engineering (CBSE), Los Angeles, USA (1999).
[20] Alves, C. Castro, J. "CRE: A systematic method for COTS component selection". Proceeding of the XV Brazilian Symposium of Software Engineering (SBES), Rio de Janeiro, Brazil (2001).
[21] Maiden, N.A.M., Croce, V., Kim, H., Sajeva, G., Topuzidou, S. "SCARLET: Integrated Process and Tool Support for Selecting Software Components". Cechich, A. Piattini, M., Vallecillo, A., eds. "Component-Based Software Quality. Methods and Techniques", Volume 2693 de LNCS, Springer(2003) 85-98.
[22] Cheng, L., Cooper, K., "COTS-Aware Requirements Engineering and Software Architecting", Software Engineering Research and Practice (2004) 57-63.
[23] José Luis Barros Justo, "Técnicas de para la clasificación/recuperación de componentes software reutilizables y su impacto en la calidad", Universidad de Vigo, Sistema de Información nº9 / 1998 (37-52).
[24] Manuel F. Bertoa, José M. Troya, Antonio Vallecillo, "Aspectos de calidad en el desarrollo de software basado en componentes", Capítulo 8 del libro "Calidad en el desarrollo y mantenimiento del software", RA-MA, 2002. ISBN: 84-7897-544-6.Mary Gorman, "Savvy Shopping for COTS Software", EBG Consulting (http://www.ebgconsulting.com/articles.php), 2008.
[26] M.R.J Qureshi, S.A. Hussain, "A reusable software component-based development process model", Advances in Engineering Software 39 (2008) 88-94. Departament of Computer Science, COMSATS Institute of Information Technology, Lahore, Pakistan.
[27] Jose Antonio Pow Sang Portillo, "La Especificación de Requisitos con Casos de Uso: Buenas y Malas Prácticas", II Simposio Internacional de Sistemas de Información e Ing. de Software en la Sociedad del Conocimiento-SISOFT 2003, Pontificia Universidad Católica del Perú, Lima-Perú, 2003.
[28] Información y documentación de SPEM http://www.omg.org/spec/SPEM/2.0/ Consultado 15 de Mayo 2010.
[29] UML, http://es.wikipedia.org/wiki/Lenguaje_Unificado_de_Modelado Consultado 15 de Mayo 2010.
[30] Documentación y aplicación Eclipse, http://www.eclipse.org/ Consultado 15 de Mayo 2010.
[31] Documentación y aplicación Eclipse Process Framework Project, http://www.eclipse.org/epf/ Consultado 15 de Mayo 2010.
[32] Documentación de H2, http://www.h2database.com/html/main.html Consultado 15 de Mayo 2010.
[33] Documentación OCL, http://www.omg.org/technology/documents/formal/ocl.htm Consultado 15 de Mayo 2010.

[34] Microsoft Word para MAC, http://www.microsoft.com/mac/downloads.mspx, Consultado el 4 de Junio de 2010.
[35] MAC OS X, http://www.apple.com/es/macosx/ Consultado el 4 de Junio de 2010.
[36] Mozilla Firefox 3.6, http://www.mozilla-europe.org/es/firefox/. Consultado el 8 de Junio de 2010.
[37] Maria Pierez, Sven Casteleyn, Ismael Sanz, Maria Jose Aramburu. "Requirements gathering in a model-based approach for the design of multi-similarity systems." In MoSE+DQS'09: Proceeding of the 1st international workshop on model driven service engineering and data quality and security, pages 45-52, New York, NY, USA, 2009. ACM.
[38] The Apache Software Foundation, "Apache lucene," http:// lucene.apache.org/java/2 Consultado el 2 de Febrero de 2009.
[39] K. Indukuri, A. Ambekar, A. Sureka, "Similarity analysis of patent claims using natural language processing techniques." International Conference on Computational Intelligence and Multimedia Applications, 2007, vol.4, pp. 169-175, Dec. 2007.
[40] Anil Jadhav, Rajendra Sonar, "Analytic Hierarchy Process(AHP), Weigthed Scoring Method(WSM), and Hybrid Knowledge Based System (HKBS) for Software Selection: A Comparative Study", Second International Conference on Emerging Trends in Engineering and Technology, ICETET-09.

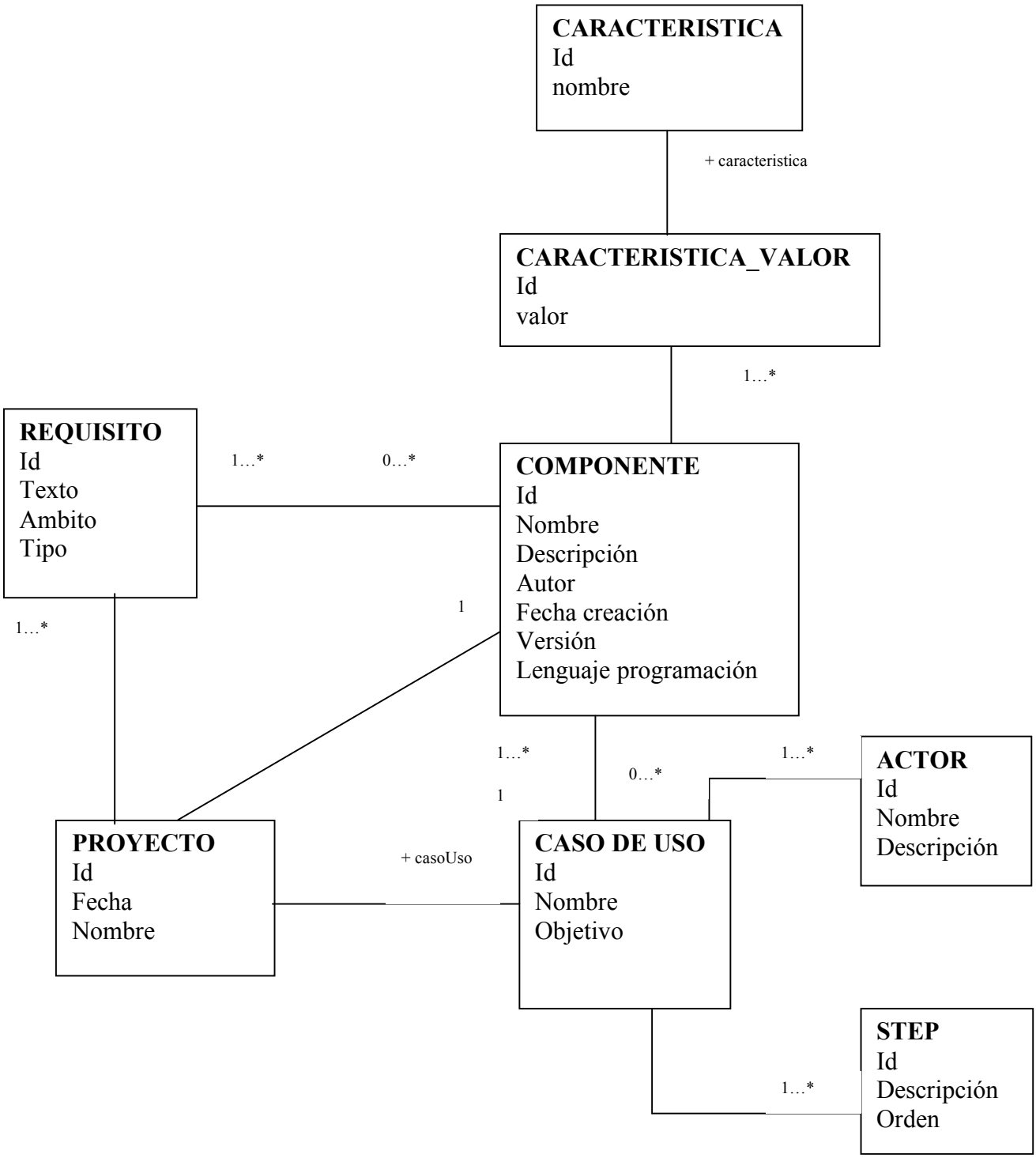
Anexo I: Plantilla de casos de uso

Esta es la plantilla proporcionada en el proceso COTSRE para la creación de casos de uso.

Nombre:
Iniciador:
Objetivo:
Escenario principal: 1. 2. 3. … Extensiones: X. a. b. … …

Anexo II: Modelo conceptual para COTSRE APP

Este es el modelo conceptual con el que se ha modelado la base de datos de H2 para COTSRE APP.



Anexo III: Términos de SPEM 2.0 y EPF Composer

Término Inglés	Término Español	Icono
Activity	Actividad	
Activity Detail Diagram	Diagrama de Detalle de Actividad	
Activity Diagram	Diagrama de Actividad	
Activity Use Kind	Tipo de Reutilización de Actividad	
Artifact	Artefacto	
Breakdown Element	Elemento de Desglose (de Descomposición)	
Capability pattern	Patrón de Proceso (patrón de capacidad)	
Category	Categoría	
Category Package	Paquete de Categorías	
Checklist	Lista de Comprobación	
Composite Role	Rol Compuesto (Equipo)	
Concept	Concepto	
Configuration Method	Configuración de Método	
Content Description	Descripción de Contenido	
Content Element	Elemento de Contenido	
Content Package	Paquete de Contenido	
Custom Category	Categoría Personalizada	
Deliverable	Entregable	
Deliverable Component	Componente de Entregable	
Delivery Process	Proceso para Despliegue (de entrega)	
Describable Element	Elemento Describible	
Discipline	Disciplina	
Discipline Group	Grupo de Disciplinas	
Domain	Dominio	
Estimation Considerations	Consideraciones para el Cálculo	
Example	Ejemplo	
Guidance	Guía (Instrucción)	
Guideline	Directriz	
Iteration	Iteración	

Library Method	Biblioteca de Métodos	
Method Content	Contenido de Método	
Method Element	Elemento de Método	
Method Package	Paquete de Métodos	
Milestone	Hito (Objetivo)	
Outcome	Resultado	
Participant Process	Participante de Proceso	
Performer Process	Realizador de Proceso	
Phase	Fase	
Plugin Method	Plugin de Método	
Practice	Práctica	
Process	Proceso	
Process Component	Componente de Proceso	
Process Element	Elemento de Proceso	
Process Package	Paquete de Proceso	
Process Parameter	Parámetro de Proceso	
Process Planning Template	Plantilla para Planificación de Proceso	
Report	Informe	
Reusable Asset	Activo Reutilizable	
Roadmap	Hoja de Ruta (Mapa)	
Role	Rol	
Role Set	Conjunto de Roles	
Role Use (Role Descriptor)	Rol en Uso (Descriptor de Rol)	
Section	Sección	
Standard Category	Categoría Estándar	
Step	Paso	
Supporting Material	Material de Soporte	
Task	Tarea	
Task Use (Task Descriptor)	Tarea en Uso (Descriptor de Tarea)	
Team Allocation Structure	Estructura de Asignación de Equipos (de Personal)	
Template	Plantilla	

Term Definition	Definición de Término	
Tool	Herramienta	
Tool Mentor	Guía de Herramienta	
Variability Element	Elemento de Variabilidad	
Whitepaper	Documentación	
Work Breakdown Element	Elemento de Desglose de Trabajo	
Work Breakdown Structure	Estructura de Desglose de Trabajo	
Work Definition	Definición de Trabajo	
Work Product	Producto de Trabajo	
Work Product Dependency Diagram	Diagrama de Dependencias de Producto de Trabajo	
Work Product Kind	Clase de Producto de Trabajo	
Work Product Usage Structure	Estructura de Utilización de Productos de Trabajo	
Work Product Use (Work Product Descriptor)	Producto de Trabajo en Uso (Descriptor de Producto de Trabajo)	
Work Sequence	Secuencia de Trabajo	