PROYECTO INFORMÁTICO

MEMORIA

HERRAMIENTA DE ADQUISICIÓN DE GUÍAS CLÍNICAS Y TRANSFORMACIÓN EN WORKFLOW

AUTOR

Beatriz García González
b.garciagonzalez@alu.um.es

DIRECTORES

Manuel Campos Martínez
manuelcampos@um.es

Jose Manuel Juárez Herrero
jmjuarez@um.es

Septiembre 2009

RESUMEN

Una Guía de Práctica Clínica (GPC) es conjunto de recomendaciones desarrolladas de forma sistemática para ayudar a las profesionales en la toma de decisiones sobre la atención sanitaria que se proporciona a los pacientes bajo circunstancias específicas. La idea de que una GPC pueda ser interpretada por un ordenador de manera que se facilite la aplicación y ejecución de los procedimientos de esta es un tema de investigación actualmente. Para que una GPC pueda ser interpretada por un ordenador se debe conseguir una representación formal de esta mediante la formalización de los procedimientos médicos que aparecen en ella. En los últimos años han surgido diferentes lenguajes y motores de deducción capaces de procesar las GPC y facilitar así su aplicación.

En este proyecto se presenta una herramienta que permite pasar mediante un procedimiento estructurado en pasos desde una GPC textual en formato HTML a una representación formal, obteniendo representaciones intermedias en cada uno de los pasos.

Como modelo de representación formal se ha elegido un lenguaje de workflow o flujo de trabajo. Un workflow es una representación de flujos de tareas que permite el guiado automático para que tanto personal humano como sistemas informáticos estén coordinados durante la ejecución. Los workflow están siendo recientemente utilizados para modelar formalmente las GPC.
En este proceso constructivo se va realizando un etiquetado de la GPC por medio de estándares clínicos y terminologías obteniendo representaciones intermedias. En todo momento debe mantener la trazabilidad entre las distintas representaciones y el texto original de la GPC.

En particular este proyecto se centra en la integración del proceso con los sistemas de información clínica.

Índice

Índice de figuras

1 INTRODUCCIÓN

En las últimas décadas, la práctica clínica está sufriendo un profundo cambio debido a la necesidad de introducir una asistencia de calidad en sus pacientes. En particular, se han realizado grandes esfuerzos para estandarizar los procedimientos de diagnóstico, tratamiento y seguimiento del paciente. El resultado de este esfuerzo se refleja en los distintos repositorios de protocolos, recomendaciones y planes de actuación disponibles en Internet. Así, el documento que recoge esta información para cada patología concreta se denomina Guía de Práctica Clínica (GPC).

La irrupción de las tecnologías de la información en el campo sanitario ha revolucionado la recogida, almacenamiento y tratamiento de la información del paciente. Los sistemas de información clínica (SIC) son los sistemas específicos para este fin, estructurando la información del paciente en la denominada historia clínica electrónica (HCE).

De esta manera, existe un gran interés en la aplicación informatizada de GPC a los pacientes con el objetivo de garantizar un tratamiento estandarizado y de calidad. Por ejemplo, siendo capaces de comprobar de manera automática si la HCE refleja la aplicación de una GPC. Por un lado, los SIC tienen como objetivo primordial el registro y organización de toda la información relativa al paciente, sin hacer especial hincapié en la metodología ni en herramientas que ayuden al clínico a interpretar los datos. Por otro lado, existen lenguajes de representación computacional de GPC pero tienen una expresividad y características que difícilmente pueden encontrarse en los SIC implantados en la actualidad.

Por tanto, surge la necesidad de reconciliar las perspectivas del modelado de GPC y los SIC puesto que la primera es habitualmente representada en lenguaje natural y los SIC utilizan una representación estructurada y estandarizada.

Este acercamiento entre ambas perspectivas puede realizarse de manera gradual mediante un proceso que se pueda descomponer en una secuencia de pasos independientes. En particular, distinguimos tres grandes tareas a resolver. La primera, la estructuración del documento de GPC (habitualmente en formato HTML) con el fin de identificar las diferentes secciones constituyentes. En estas secciones deberemos diferenciar entre aquellas que contienen información sobre: la propia GPC (fecha creación, autores, revisión, etc.), condiciones de aplicación, recomendaciones generales y aspectos procedimentales (tratamientos, diagnósticos, etc.). La segunda tarea a resolver es formalizar todos aquellos aspectos susceptibles de ser tratados de forma automática por un ordenador. Por ejemplo, el conjunto de acciones que pueden llevar a un diagnóstico, como sería obtener el resultado de una prueba de laboratorio y realizar un examen físico, se puede representar mediante un flujo ordenado de acciones.

La tercera tarea consiste en el establecimiento de un canal de comunicación que permita la interoperabilidad entre la GPC informatizada y la información de cualquier paciente, almacenada en su HCE. Esta comunicación debe proporcionar una interoperabilidad sintáctica y semántica. Para facilitar esta comunicación se deberá cumplir con los estándares clínicos definidos para este fin. Dentro de esta comunicación distinguimos entre los estándares de transporte, con los que se consigue la interoperabilidad sintáctica, y estándares de codificación de la información clínica, con los que se consigue la interoperabilidad semántica.

En la Figura 1.1 mostramos un ejemplo concreto del proceso descrito mediante estas tareas. En primer lugar, disponemos del documento de GPC en formato HTML que, mediante un proceso asistido, puede organizarse mediante un estándar como GEM II. Para la formalización de los

aspectos procedimentales extraídos del documento estructurado se ha utilizado el lenguaje de workflow YAWL. Finalmente, para la integración del proceso con un SIC se utiliza HL7 como estándar de transporte clínico y una codificación específica de un SIC concreto (CH4-UCI) instalado en una Unidad de Cuidados Intensivos de un Hospital de la Comunidad de Madrid.

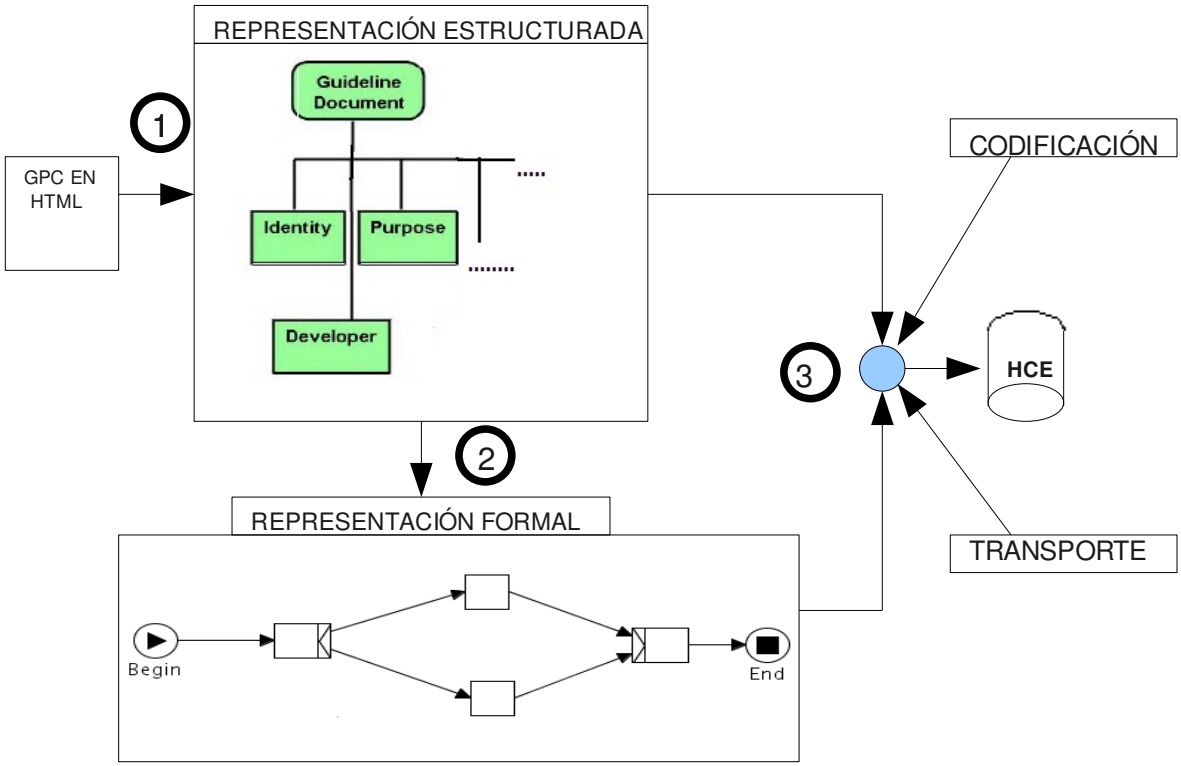

Figura 1.1: Comunicación entre GPC y SIC.

Además, para facilitar la legibilidad del proceso es fundamental facilitar la trazabilidad entre los diferentes pasos. Por ejemplo, a partir de una acción definida en el workflow, poder ver a qué apartado del documento original del GPC pertenece.

En el siguiente apartado explicaremos más detalladamente el concepto de GPC y la importancia de estas dentro del la calidad asistencial sanitaria.

1.1 GUÍA DE PRÁCTICA CLÍNICA

Una Guía de Práctica Clínica (GPC) es un conjunto de recomendaciones desarrolladas de forma sistemática para ayudar a los profesionales y a los pacientes en la toma de decisiones sobre la atención sanitaria más apropiada, seleccionando las opciones diagnósticas y/o terapéuticas más adecuadas en el abordaje de un problema de salud o una condición clínica específica [Field and Lohr, 1992]. De esta forma, una GPC puede ser considerada como una potente herramienta de decisión que ayuda a reducir la diferencia entre la práctica clínica desarrollada y la óptima [Handley et al, 1994].

Sin embargo, en la aplicación práctica de las GPC a un paciente concreto, hay que extraer el conocimiento recogido en ellas para crear una secuenciación de actividades, contextualizadas y personalizadas, que conforman lo que se denomina Planes de Cuidado (PC) (Clinical Pathways), que representan una estrategia global para la gestión de la asistencia sanitaria para un paciente concreto, mostrando dicha gestión del problema médico desde la perspectiva de un flujo de trabajo [HISA, 2006; Carrasco and Ferrer, 2001]. Para el diseño de un PC, es necesaria la contextualización y adaptación de un GPC a un entorno y pacientes concretos.

Ambas herramientas (GPC y PC) se encuentran en la base de la mayoría de los procesos para la estandarización y mejora de la calidad de los servicios sanitarios [Field and Lohr, 1992]. El objetivo principal en la definición de GPC y PC radica en la necesidad de mejorar la calidad de asistencia sanitaria y la utilización de recursos económicos y humanos, en la atención de pacientes, fomentando la utilización de buenas prácticas [Grimshaw and Russel, 1993; Kohn et al., 1999; Micieli et al, 2002; Qualigini et al, 2004; Wang et al., 2004]. Con el uso de estas herramientas se consigue una mejora de la calidad asistencial ya que pueden producir una reducción de los recursos sanitarios utilizados e incrementar la eficiencia de los estos reduciendo así los costes. En definitiva, las GPC y PC son un instrumento que permite la toma de decisiones informadas por parte de los profesionales de la salud, y sirven de apoyo a los gestores en la planificación de lo recursos sanitarios facilitando la priorización de necesidades.

Sin embargo, la creación y publicación de GPC no ha producido el cambio esperado en la práctica clínica diaria [Wolff et al, 1998; Zielstorff RD, 1998], debido principalmente a que la difusión de las GPC ha estado limitada a documentos textuales y que, en la mayoría de los casos, la aplicación de un GPC a un paciente concreto, para diseñar el correspondiente PC requiere una profunda lectura e interpretación de la misma. La utilización de las nuevas tecnologías de la información y las comunicaciones ha permitido la publicación de GPCs en internet o en intranets hospitalarias, mejorando en gran medida la difusión de las mismas [Zielstorff et al., 1998; SIGN, 2005; NGPC, 2005; NZGG, 2005; SIGN, 2005; NICE, 2005]. Esto ha supuesto una mejora para el mundo de la calidad asistencial sanitaria, pero el verdadero reto está en la integración de GPC en sistemas inteligentes de ayuda a la decisión (SAD) que apoyen la puesta en práctica de las GPC, y su contextualización y personalización en PC, además de permitir la fusión de diferentes PC aplicables a un mismo paciente. La utilización de este tipo de sistemas permite una mejora mayor de la calidad asistencial, especialmente cuando están integrados junto con Sistemas de Información Clínica (SIC) que hacen posible la implementación de la Historia Clínica Electrónica (HCE) [Field and Lohr, 1992; East et al., 1991].

1.2 CONTEXTO DEL TRABAJO

Este proyecto fin de carrera se enmarca dentro de una serie de proyectos más generales que desarrolla el grupo de investigación AIKE (Artificial Inteligence and Knowledge Engineering) de la Universidad de Murcia para el desarrollo de herramientas y sistemas inteligentes en el dominio de una Unidad de Cuidados Intensivos (UCI). El principal proyecto dentro del cual se enmarca es el Proyecto PETRI “Sistemas Inteligentes para la gestión de Información un una UCI: Herramientas de soporte a la investigación y asistencia médica” (PET2006-0406).

El proyecto PETRI tiene como objetivo principal el descubrimiento de conocimiento médico para el desarrollo de un sistema de monitorización inteligente de pacientes capaz de detectar y diagnosticar

la evolución del estado del paciente ingresado en la UCI. Otro de los objetivos de este proyecto es la gestión de la calidad asistencial, que es contemplada desde una aproximación basada en guías de práctica clínica.

Este trabajo supone una continuación de trabajos anteriores enmarcados en estos proyectos y en los cuales se ha definido un sistema de información clínica (SIC) para UCI así como un conjunto de módulos inteligentes. Estos módulos han sido definidos para distintas tareas, tanto intermedias como de alto nivel.

En estos proyectos se plantea, entre otros, como objetivo el proporcionar una infraestructura tecnológica avanzada para la gestión electrónica de GPC y PC. Ello incluye soluciones para: 1) la representación computacional de GPC, 2) la edición de PC adaptadas al contexto organizacional, y 3) el análisis, la crítica y la mejora de los PC aplicados en el servicio. Este proyecto surge como una solución parcial a este objetivo general.

1.3 RESULTADOS PREVIOS

Como documentación de este proyecto vamos a partir de unos resultados iniciales:

- Tesis de máster de Ángel Cifuentes: uso de workflow para modelado de GPC .
- Tesis de máster de Patricia: medidas de similitud entre workflows [2].

En estos dos trabajos se explica como los workflow pueden ser utilizados para representar formalmente acciones de una GPC y a su vez integrarse con SIC que implementan la HCE consiguiendo así una mejora de la calidad asistencial sanitaria.

1.4 REFERENCIAS COMENTADAS

Diferentes formalismos han sido desarrollados para implementar las GPC en un formato que pueda ser interpretado por el ordenador, algunos de estos son Asbru [14], Prodigy [13] o Proforma [6]. Cada formalismo soporta específicos lenguajes de representación de GPC y varias herramientas han sido desarrolladas para facilitar el modelado y el proceso de visualización. Estas herramientas se pueden dividir en dos categorías, las que siguen un enfoque centrado en el modelo o un enfoque centrado en el documento.

El objetivo del enfoque centrado en el modelo es la creación de un modelo conceptual de la GPC original consiguiendo así una interpretación visual, pero sin guardar conexión directa entre el documento y el modelo. En el enfoque centrado en el documento el objetivo es marcar partes del documento original de la GPC y generar un modelo semiformal de la GPC mediante una representación estructurada de esta. De esta manera, se guarda una relación directa con el documento original de la GPC y es posible la consulta del texto literal de la GPC.

A continuación vamos a nombrar algunas de las herramientas cuyo enfoque es centrado en el documento como será la realizada en este proyecto y destacar su principales características.

1.4.1 Delta

Delta [9] es una extensión del proyecto GMT (Guideline Markup Tool). Es una herramita enfocada

al documento cuyo objetivo es pasar de una representación en HTML de la GPC a una representación semiformal en XML mediante el marcado de texto de las GPC en HTML. En DELTA se distinguen dos elementos principales:

- **Links:** Permiten conectar partes del fichero HTML con partes del fichero XML
- **Macros:** Combinan múltiples elementos XML con sus atributos y permite construir nuevos documentos XML bien formados introduciendo las macros adecuadamente.

Con esta herramienta se trabaja en dos pasos:

- Un primer paso donde se transforma la GPC en HTML a una representación intermedia en XML. Las etiquetas XML utilizadas dependerán del fichero Macro que se use. Algunas de estas macros deberán tener enlaces al parte del texto. Uno de los ficheros de matros que se ofrece es el que contiene elementos que forman el lenguaje Asbru.
- Una vez hecha la primera transformación se procede al segundo paso donde se pasa de la transformación intermedia en XML a una segunda transformación en XML. Al igual que anteriormente se elige un fichero de Macros donde se indicarán las etiquetas XML utilizadas para marcar.

Otras funcionalidades a destacar de DELTA son que permite señalizar algunas etiquetas elegidas por el usuario, permite ocultar u mostrar elementos XML, permite colapsar o expandir elementos XML y genera informes XSLT scripts.

1.4.2 Steeper

Steeper [1], al igual que Delta, es una herramienta que sigue un enfoque centrado en el documento. El objetivo de Steeper es la formalización de la GPC mediante el marcado y consta de múltiples entornos de interconexión entre usuarios:

- para marcado de texto libre (delimitación de bloques de conocimiento inicial).
- para transformación interactiva paso a paso de estructura XML y contenido.
- para enlaces que permiten facil navegación a través de niveles de transformación.

Steeper contiene un procesador embebido de XSLT el cual lleva la parte no interactiva de la transformación, mientras que el marcado y la transformación interactiva son llevadas por un lenguaje llamado XKBT. Se utiliza XML y XHTML para enlazar con el documento.

La principal caracteristica de Steeper es la capacidad para descomponer explicitamente en diferentes fases la formalización del documento. Se definen cinco pasos de transformación para obtener un conocimiento base de la GPC.

Cada instancia del modelo que se va creando con el marcado es realmente una colección de componentes pertenecientes a cuatro categorias: componente de procedimiento (idea de escenario), causalidad, objetivo declaración y definición de conceptos.

La representación del modelo en Steeper respeta su carácter progresivo, consiste en múltiples módelos interconectados en forma de DTD's y refleja la evolución de los componentes durante la transformación.

El marcado en Steeper se va haciendo eligiendo una categoria y el texto para enlazarla, se generan reglas XKBT y va formandose el árbol con los nodos correspondientes. Cada nodo tendrá un formulario para introducir atributos de dicho nodo.

También se pueden transformar algunos elementos del documento al siguiente nivel, que de nuevo consiste en un árbol XML.

1.4.3 GemCutter

GemCutter [10] es otra herramienta para la representación semiformal de una GPC que sigue un enfoque centrado en el documento. El lenguaje que utiliza GEMCutter para el marcado de la GPC es GEM II. GEM, antecesor de GEM II, es un estándar que fue desarrollado con el objetivo de organizar el conocimiento de una GPC.
GEMCutter permite marcar texto de la GPC enlazándolo con cada uno de los elementos del GEM. El texto a su vez puede ser modificado y guardado en la GPC original. Al final la estructura etiquetada se guardará en un fichero xml.

1.4.4 Uruz

DEGEL [11] es un proyecto cuyo objetivo es proveer de un soporte automático para la especificación y uso de GPC en tratamiento de pacientes. El proyecto DEGEL ha desarrollado un conjunto de herramientas para el apoyo de la implementación y desarrollo de GPC.
Uruz es una herramienta web que pertenece al proyecto DEGEL. DEGEL permite cargar una GPC y con la herramienta Uruz se puede editar o marcar con etiquetas de algunas de las ontologías de las que dispone DEGEL. Uruz también permite crear una GPC nueva desde cero con alguno de los lenguajes de GPC disponibles en las librería de DEGEL.

1.4.5 Análisis comparativo

El siguiente cuadro representa un resumen de las características de cada herramienta:

Caracteristica / herramienta	**Formato GPC**	**Modelado semiformal**	**Modelado formal**	**Asignación de atributos a los elementos**
Delta	HTML	Puede utilizar varios lenguajes, entre ellos Asbru	No	No
Steeper	HTML	Propios	No	Si, atributos fijos para cada elemento
Uruz	HTML	Utiliza fichero xml con lenguaje Asbru	No	Si
GEM-Cutter	HTML, RTF	Utiliza GEM	No	No

2 ANÁLISIS DE OBJETIVOS Y METODOLOLOGÍAS

2.1 OBJETIVOS DEL PROYECTO

El objetivo del proyecto es desarrollar una herramienta que permita, de forma semiautomática la formalización de la parte procedimental de una GPC. Esta herramienta será un asistente que guiará al usuario a la formalización de acciones que se encuentran descritas en una GPC mediante lenguaje natural haciendo uso de distintos estándares y terminologías.
Partiremos de documentos de GPC en formato HTML, ya que la mayoría de las GPC están en este formato, y obtendremos una descripción de las acciones en el formato apropiado para facilitar el seguimiento de la realización de las tareas en la historia clínica de un paciente.

En particular, los subobjetivos del proyecto son los siguientes:

- Estructuración de GPC textuales.
- Etiquetado genérico de eventos para la integración con un sistema de HCE.
- Etiquetado y estructuración de mensajes estándar de farmacia o tratamiento para la comunicación de los eventos.
- Formalización de acciones de una GPC.

Los siguientes puntos no son objetivos del proyecto:

- Etiquetar y estructurar los mensajes que no tengan que ver con farmacia o tratamiento.
- Enviar los mensajes para la comunicación de eventos entre distintas herramientas.
- Poder realizar la adquisición de información sobre GPC en un formato distinto al HTML.
- Identificar casos particulares de un paciente con el flujo de trabajo construido.
- Gestión documental de las GPC.

2.2 TAREAS DEL PROYECTO Y PLANIFICACIÓN

Partiendo de los subobjetivos estipulados en el apartado anterior se plantean las siguientes tareas para el desarrollo del proyecto:

- Estructuración de GPC textuales.

1. Estudio del lenguaje de marcado de GPC eligiendo GEM II [16].
2. Desarrollo del módulo para etiquetar las partes de texto de la GPC con GEM II.

   El principal objetivo de este proyecto es la adquisición de la información de la GPC textuales. Para conseguir esto el primer paso será conseguir una representación estructurada documento de la GPC mediante el etiquetado del texto de la GPC con elementos del modelo GEM II.
   GEM II(Guideline Elements Model) es un modelo que representa los elementos de una GPC permitiendo así organizar la información contenida en una GPC. El objetivo principal de GEM II es facilitar la traducción de documentos de GPC en lenguaje natural a un formato estructurado del documento que pueda ser procesado por los ordenadores. Hemos elegido

GEM II ya que es la especificación estándar para el modelado de elementos de GPC, el estándar al que corresponde es ASTM E2210 – 06.

•Etiquetado genérico de eventos para la integración con un sistema de HCE (Historia clínica electrónica).

1. Estudio de las HCE y de CH4.
2. Relación entre las etiquetas del lenguaje de marcado elegido y la terminología de CH4.
3. Desarrollo del módulo para etiquetar los eventos CH4.

La historia clínica es una fuente de datos fundamental y una herramienta básica para la investigación biomédica, la formación de estudiantes y la educación médica continuada. Los avances en las tecnologías de la información y las necesidades impuestas por nuevos modelos de gestión clínica están favoreciendo un uso cada vez más extendido de historias clínicas en formato electrónico.
Este proyecto tendrá una visión aplicada, para lo cual se utiliza la herramienta CH4-UCI. La herramienta CH4-UCI es un sistema de HCE que tiene como objetivo la gestión de informes y de la historia clínica de pacientes en una UCI, facilitando así el trabajo diario y la comunicación entre los miembros de la plantilla, así como la consulta de historias clínicas. Esta herramienta utiliza la terminología CH4 para la comunicación de eventos, por esto, para una futura integración con CH4-UCI se podrán etiquetar eventos de CH4 que se reconozcan en la GPC.

•Estructuración de mensajes estándar para la comunicación de eventos.

1. Estudio de los protocolos.
2. Estudio de los mensajes y segmentos del protocolo elegido referentes a tratamientos y prueba.
3. Desarrollo del módulo para etiquetar y estructurar los mensajes del protocolo.

Para que esta herramienta pueda interoperar con otros SIC e intercambiar información de eventos, será necesario establecer un protocolo de comunicación. El protocolo utilizado pertenecerá al estándar HL7 ya que es el protocolo de comunicación entre SIC más utilizado actualmente. Al tener estos mensajes estructurados, cuando se tenga un motor de ejecución de workflow, este podría comunicarse directamente con otros subsistemas y solicitar y recuperar información transmitiendo estos mensajes.

•Formalización de acciones de una GPC.

1. Estudio de los posibles formalismos para representar las acciones de una GPC.
2. Elección del formalismo.
3. Estudio de las opciones para integrar o desarrollar una herramienta que edite el formalismo elegido.
4. Desarrollo del módulo para crear y editar la representación del formalismo de manera que exista un mapeo entre las acciones representadas y la GPC original.

• Otras tareas.

1. Documentación de la herramienta a nivel de usuario.
2. Documentación de la herramienta a nivel de desarrollador.

2.3 METODOLOGÍA

Para el desarrollo de la herramienta se utilizará un modelo de prototipado incremental. El desarrollo incremental es una metodología de desarrollo que permite construir el proyecto en etapas incrementales donde cada etapa agrega funcionalidad. Cada una de las etapas lleva un fase de requerimientos, diseño, codificación y pruebas. Con esta metodología la solución se va mejorando en forma progresiva a través de múltiples iteraciones.
Se ha elegido esta metodología ya que se conocen bien las etapas que tendría el proyecto. Se deberán desarrollar los siguientes módulos:

1. Módulo de etiquetado de la GPC con GEM.
2. Módulo de etiquetado con elementos de CH4
3. Módulo de etiquetado y estructuración de mensajes para eventos en HL7.
4. Módulo para la construcción del modelo en lenguaje formal que representa la GPC.

En cada una de estas etapas se va añadiendo funcionalidad a la aplicación pasando de la GPC en texto HTML a la formalización final de esta. En cada uno de los módulos que se indican anteriormente se consigue:

- Módulo de etiquetado de la GPC con GEM. Se obtiene una representación estructurada del GPC.
- Módulo de etiquetado con elementos de CH4 y módulo de etiquetado y estructuración de mensajes para eventos en HL7. Relacionando elementos de CH4 y HL7 con elementos GEM se obtiene una representación estructurad más completa de la GPC, se indicarán que partes corresponderán con codificación CH4 y mensajes HL7.
- Módulo para la construcción del modelo en lenguaje formal que representa la GPC. Se obtiene la formalización final de la GPC relacionando las tareas del modelo formal con elementos del documento estructurado y la GPC original.

2.4 ESTÁNDARES CLÍNICOS Y ALTERNATIVAS

En la actualidad existen una gran variedad de sistemas de información sanitarios, y a veces, es necesaria la comunicación e interoperabilidad entre ellos para un mejor funcionamiento del sistema sanitario. Para que dos sistemas puedan comunicarse e intercambiar información es necesaria un interoperabilidad tanto sintáctica como semántica. La interoperabilidad sintáctica hacer referencia a la estructura de la comunicación, por ejemplo HL7, del que hablaremos más adelante. La interoperabilidad semántica hace referencia al significado de la comunicación, asegura que los datos intercambiados puedan ser usados y entendidos por el que los recibe. Para que esto sea posible se han definido estándares semánticos para la interoperabilidad entre sistemas información sobre distintos temas y se pueden clasificar en seis categorías:

1. Estándares de mensajería e intercambio de datos: permiten el intercambio del flujo entre los

sistemas y organizaciones en forma consciente ya que contienen especificaciones para el formato, los elementos de los datos y la estructura. Algunos ejemplos son HL7 para los datos administrativos de los pacientes tales como los demográficos o relacionados con consultas, DICOM para las imágenes radiológicas y el NCDP para las prescripciones electrónicas.

2. Estándares de terminología: proveen códigos específicos para conceptos clínicos. Algunos son LOINC para resultados de laboratorio, SNOMED para términos clínicos.
   Aparte de estas terminologías estándares existen otras que han sido desarrolladas para casos particulares por lo que no son estándares. Este es el caso de la terminología que usaremos en nuestro proyecto, CH4. Esta terminología fue creada particularmente para la comunicación de eventos en el proyecto CH4-UCI.
3. Estándares de documentos: indican que tipo de información está incluida en un documento y donde puede ser hallada. Algunos ejemplos son CDA, estándar de intercambio para documentos clínicos tales como notas de alta y evoluciones. Otro es CCR que provee un formato estándar para la comunicación entre profesionales de la salud.
   GEM II es otros estándar que proporciona un lenguaje para modelar GPC de manera que el texto de esta quede estructurado.
4. Estándares conceptuales: permiten que los datos sean transportados a lo largo de los sistemas sin perder el significado o el contexto. Un ejemplo es el HL7 RIM que provee un marco para describir los datos clínicos y el contexto circulante.
5. Estándares de aplicaciones: Determinan el modo en que las reglas de negocio son implementadas su interacción con los sistemas de software, por ejemplo el CCOW
6. Estándares de arquitectura: definen los procesos involucrados en el almacenamiento y la distribución de datos.

De estos estándares, en este proyecto hemos hecho uso de GEM II y el estándar de mensajería de intercambio de datos HL7 en su versión 2.5. En cuanto a la terminología utilizada usaremos CH4, que no es estándar pero es la que utiliza la herramienta CH4-UCI para comunicar eventos.
El principal objetivo de utilizar HL7 y CH4 es conseguir en futuros trabajos la interoperabilidad de esta herramienta con CH4-UCI. Mediante el protocolo HL7 se conseguirá la interoperabilidad sintáctica y con CH4 la interoperabilidad semántica entre esta herramienta y la herramienta CH4-UCI.
A continuación se explican con más detalle los estándares y terminologías utilizados en este proyecto.

2.4.1 GEM II

GEM (Guidelines Elements Model) [15] es un lenguaje de modelado de GPC que permite organizar la información contenida en una GPC estructurándola con sus elementos. El formato en el que representa el modelo GEM es en XML, de esta manera se facilita el procesamiento de la información que contiene el modelo.
GEM II [18] está construido como una jerarquía de más de 100 elementos de los cuales 10 representan las ramas principales. En la Figura 2.1 se representan estos 10 elementos y a continuación se describe que parte de información de la GPC representa cada uno.

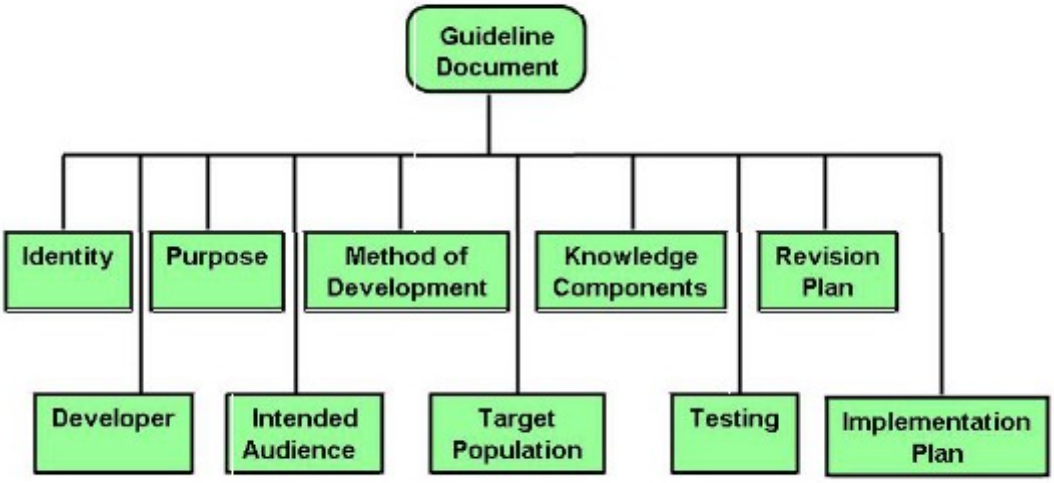

Figura 2.1: Principales elementos de GEM II. [16]

Identity: en esta rama se encuentran los elementos que identifican a la GPC. Los elementos de esta rama son los que se muestran en la Figura 2.2.

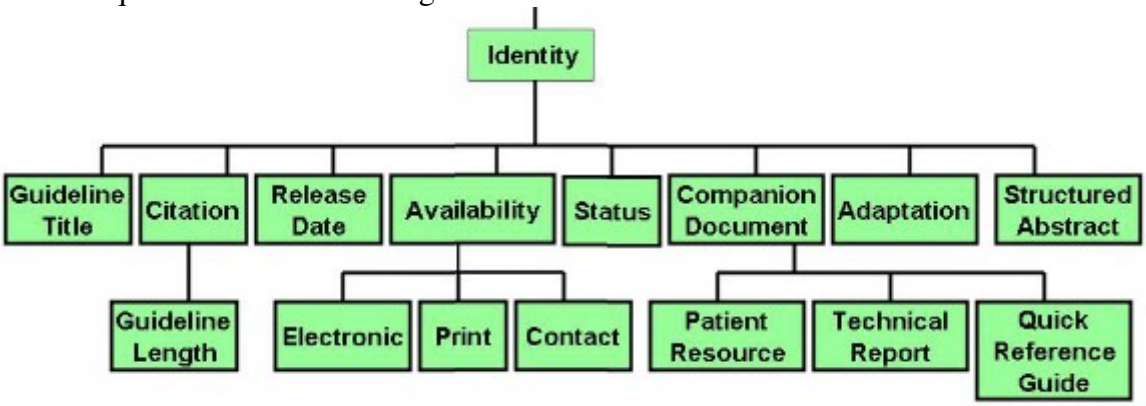

Figura 2.2: Elementos de Identity. [16]

Developer: en esta rama se encuentran los elementos que representa la información sobre los desarrolladores de la GPC. Los elementos de esta rama son los que se muestran en la Figura 2.3.

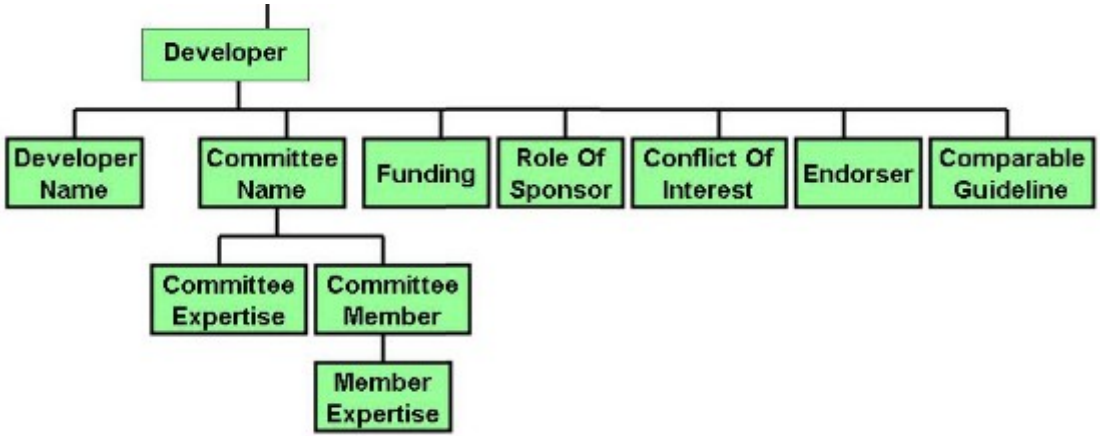

Figura 2.3: Elementos de Developer. [16]

Purpose: en esta rama se encuentran los elementos que representan el objeto por el cual se ha desarrollado la GPC. Se muestran en la Figura 2.4.

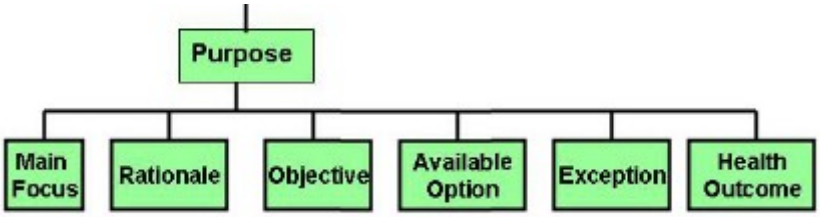

Figura 2.4: Elementos de Purpose. [16]

Intended Audience: en esta rama se encuentran los elementos que representan los destinatarios a los que va dirigida la GPC.
Target Population: en esta rama se encuentra los elementos que representan información sobre la población sobre la cual se puede aplicar las recomendaciones de la GPC.
Sus elementos se muestran en la Figura 2.5.

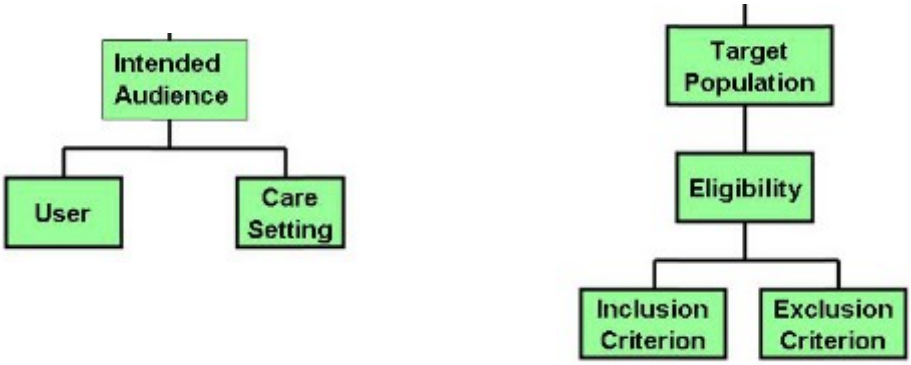

Figura 2.5: Elementos de Inteded Audience y Target Population. [16]

Method of Development: en esta rama se encuentran los elementos que representan información sobre el método de desarrollo de la GPC. Se muestran en la Figura 2.6.

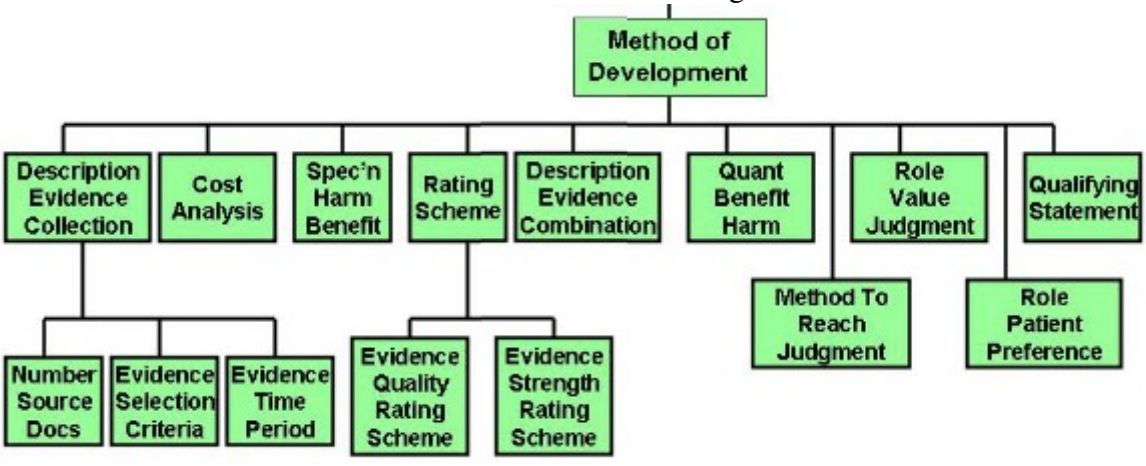

Figura 2.6: Elementos de Method of Development. [16]

Testing: en esta rama se encuentran los elementos que representan información sobre las pruebas.

Revison Plan: en esta rama se encuentran los elementos que representa información sobre futuros planes de revisión y actualización de la GPC.
Los elementos de las ramas de testing y revisión plan se representan en la Figura 2.7.

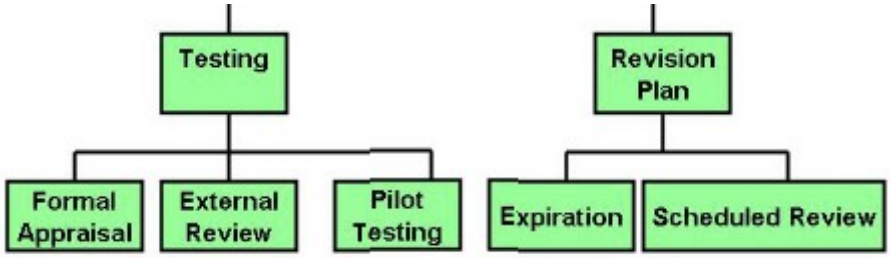

Figura 2.7: Elementos de Testing y Revision Plan. [16]

Implementation Plan: en esta rama se encuentras los elementos que representan información sobre el plan de aplicación de la GPC. Se muestran en la Figura 2.8.

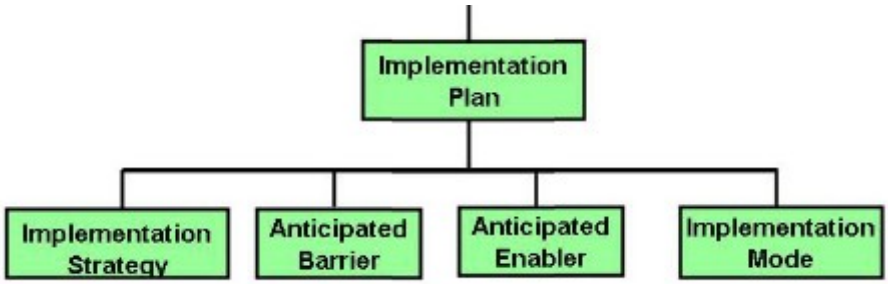

Figura 2.8: Elementos de Implementation. [16]

Knowledge Components: en esta rama se encuentran el mayor número de elementos de GEM II y representan la parte más importante de la GPC que son las acciones, las recomendaciones y la definición de la GPC. Se muestran en la Figura 2.9.

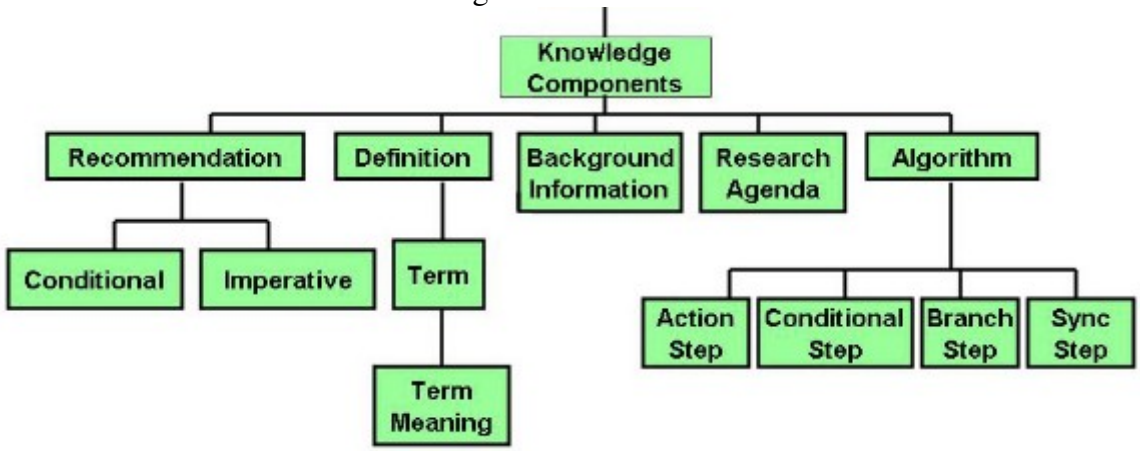

Figura 2.9: Elementos de Knowledge Componentes. [16]

Las ramas de recomendaciones condicional e imperativa a su vez contienen más elementos que representan dichas recomendaciones de forma más específica. Se muestran en las Figura 2.10 y 2.11.

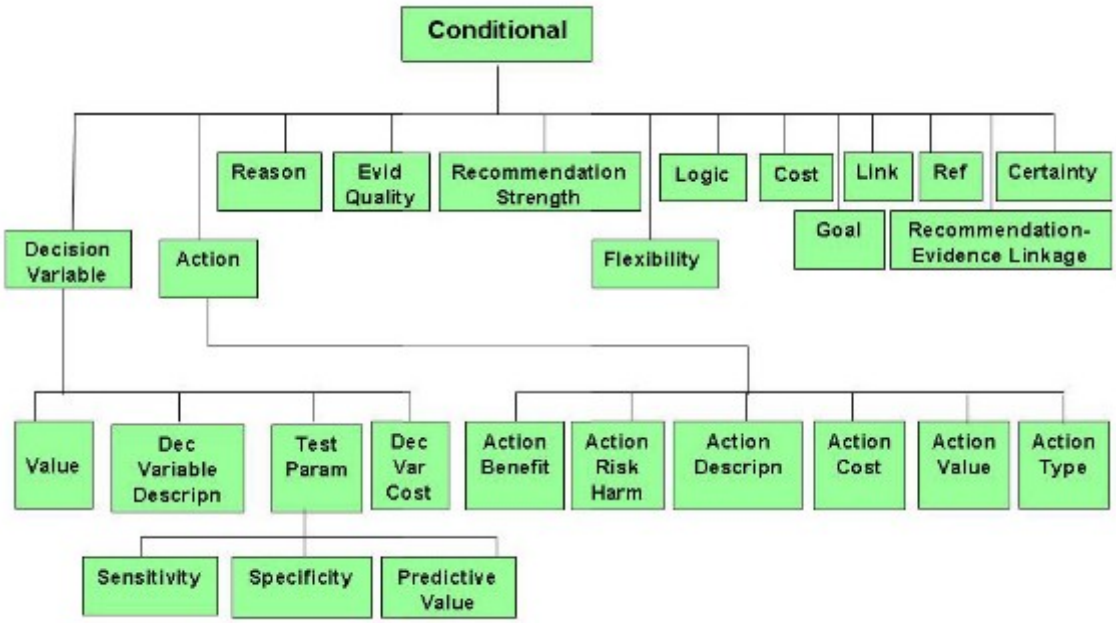

Figura 2.10: Elementos de Conditional. [16]

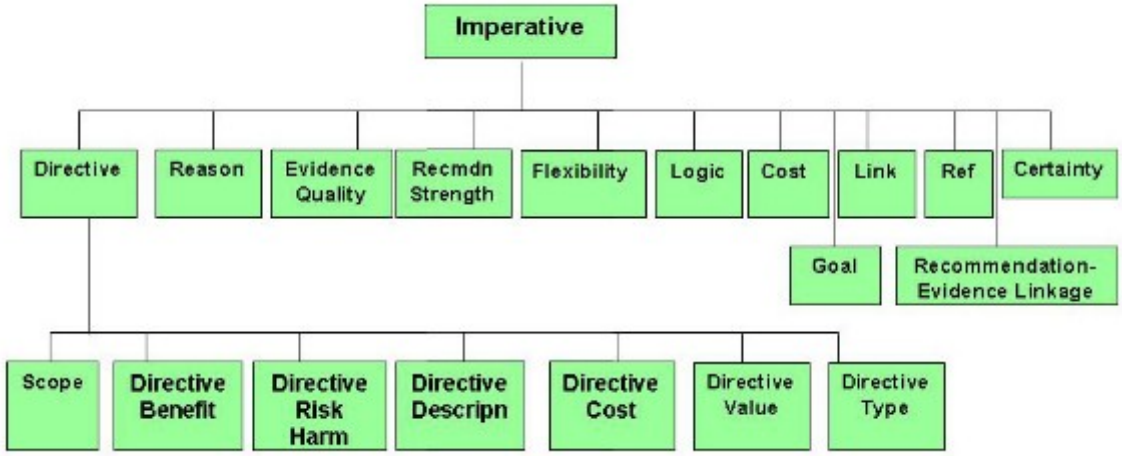

Figura 2.11: Elementos de Imperative. [16]

Para más información sobre el significado de cada elemento se puede consultar el anexo1.

2.4.2 CH4

La historia clínica es una fuente de datos fundamental y una herramienta básica para la investigación biomédica, la formación de estudiantes y la educación médica continuada. Los avances en las tecnologías de la información y las necesidades impuestas por nuevos modelos de gestión clínica están favoreciendo un uso cada vez más extendido de historias clínicas en formato electrónico. Los sistemas de HCE se comunican mediante eventos según unas determinadas terminologías.

Este proyecto se debe integrar con la herramienta CH4-UCI. La herramienta CH4-UCI es un sistema de HCE que tiene como objetivo la gestión de informes y de la historia clínica de pacientes en una UCI, facilitando así el trabajo diario y la comunicación entre los miembros de la plantilla, así como la consulta de historias clínicas. Los principales objetivos de la herramienta son [12]:

- Informatización de los datos relativos a pacientes ingresados en la UCI
- Generación automática de informes y gestión de una base de datos documental que los almacena
- Gestión de una base de datos con historias clínicas de pacientes de una UCI La herramienta está centrada en el proceso de gestión de pacientes a lo largo su estancia en la UCI.

  El principal motivo por el cual se decide desarrollar la herramienta CH4-UCI es debido a que los sistemas de información clínica convencionales no suelen adaptarse bien a casos particulares de la UCI, debido a características propias de las mismas, como son la necesidad de proporcionar mecanismos ágiles para intercambiar información entre distintos turnos de guardia, realizar un seguimiento evolutivo de las posibles complicaciones de los pacientes a lo largo de su estancia en la UCI o el alto volumen de datos manejado en una UCI y la necesidad de acceder a ellos en tiempo crítico. Por estos motivos se decide desarrollar la herramienta CH4 teniendo en cuentas estos requerimientos.

Por otra parte, la implantación de CH4 en una UCI permite simplificar otros procesos cada vez más demandados por la sociedad, como:

- La trazabilidad de la asistencia médica recibida por los pacientes, que queda perfectamente documentada.
- La gestión y el control de la calidad asistencial.

La herramienta CH4-UCI utiliza la terminología CH4 para la comunicación de eventos. En las GPC pueden aparecer términos referidos a algunos de los conceptos que representa esta terminología y es interesante etiquetarlos como tales para futura integración de este proyecto con la herramienta CH4-UCI consiguiendo así una interoperabilidad semántica.
La terminología CH4 está compuesta por los siguientes términos:

- Problema: Cualquier incidente del paciente en la UCI.
- Antecedente: Cualquier afección anterior al ingreso.
- Tratamiento: Acto terapéutico.
- Perfil Tratamiento: Conjunto de tratamientos para abordar una misma patología.
- Prueba de Analítica: Resultados de análisis.
- Prueba de Exploración Física: Resultados de estudio físico del paciente.
- Prueba de Imagen: resonancia, TAC, …
- Valoración: Estimación.
- Diagnóstico Evolutivo: Diagnóstico tras cada día del paciente en la UCI.
- Diagnóstico Alta: Diagnóstico final del paciente.
- Gráficas: Datos biométricos obtenidos del servicio de enfermería.
- Técnica: Método médico para conseguir un objetivo.
- Procedimiento: Pasos a seguir con para llevar a cabo una técnica.
- Estudio de Hemodinámica: Resultados del análisis de sangre.

- Estudio de Gravedad: Indicadores que cuantifican el estado del paciente: nivel de consciencia, gravedad, etc.

2.4.3 HL7

Uno de los objetivos es la estructuración de mensajes para intercambiar datos sobre tratamientos. Para esto necesitamos un protocolo de mensajería e intercambio de datos. En este proyecto se utilizará el estándar HL7 actualmente es el protocolo estándar para comunicación entre SIC más utilizado.
HL7 (Health Level Seven) [18] es una organización de desarrollo de estándares para el intercambio electrónico de información médica. Los estándares HL7 son desarrollados por la organización ANSI y siguen el modelo OSI. Level Seven hace referencia al nivel 7 del modelo OSI, el nivel de aplicación.
HL7 proporciona estándares de interoperabilidad entre sistemas sanitarios. Estos estándares hacen posible una mejora en la atención de la salud ya que optimizan el flujo de trabajo, reducen la ambigüedad y mejoran la transferencia de conocimientos entre los usuarios de los sistemas de información sanitaria.
Originalmente el nombre de HL7 se asociaba a un estándar de mensajería para intercambio electrónico de datos de salud entre aplicaciones. Poco a poco ha ido creciendo la necesidad de generar sistemas de información integrados regionalmente haciendo necesario el desarrollo de un espectro más amplio de estándares que faciliten la interoperabilidad por esto la organización HL7 ha desarrollado varios estándares que faciliten el intercambio de información de salud [7].
Algunos de estos estándares desarrollados por el HL7 son:

- **Mensajería HL7 Versión 2:** Estándar de mensajería para el intercambio electrónico de datos de salud.
- **Mensajería HL7 Versión 3:** Estándar de mensajería para el intercambio electrónico de datos de salud basada en el RIM (Reference Information Model).
- **CDA HL7:** (Clinical Document Architecture) Estándar de arquitectura de documentos clínicos electrónicos.
- **SPL HL7:** (Structured Product Labeling) Estándar electrónico de etiquetado de medicamentos.
- **HL7 Medical Records:** Estándar de administración de Registros Médicos.
- **GELLO:** Estándar para la expresión de reglas de soporte de decisiones clínicas.
- **Arden Syntax:** Es estándar sintáctico para compartir reglas de conocimiento clínico.
- **CCOW:** Es un framework estándar para compartir contexto entre aplicaciones.

Este proyecto se desarrolla teniendo que cuenta la futura integración con otros SIC para lo cuál elegimos el estándar HL7 en su versión v2.5 ya que es la más utilizada y la que tiene más aceptación a medio/largo plazo.

La versión 2.5 de HL7 es un protocolo para el intercambio de datos clínicos a través de mensajes. Para transferir la información, la sintaxis de los mensajes utiliza cadenas ASCII con delimitadores, tiene pocas restricciones semánticas.
Los mensajes en HL7 v2.5 están compuestos por segmentos, campos, caracteres delimitadores y componentes organizados jerárquicamente como se muestra en la Figura 2.12.

Figura 2.12: Estructura de mensajes HL7.

Un mensaje está compuesto por segmentos separados entre sí por el carácter <CR>. Cada segmento a su vez está conformado por campos divididos por el separador de campos (|). Cada campo está compuesto por uno no más componentes divididos por el separador de componentes (^) y cada componente se corresponde con un tipo de datos específico. Un componente puede estar formado por uno o más subcomponentes separados por el separador de subcomponentes (&).

El objetivo de este proyecto es estructurar los mensajes de tratamiento por lo tanto solo estructuraremos la parte de los mensajes del estándar HL7 que tenga que ver con tratamientos.
Los mensajes en HL7 se organizan en categorías como son ordenes, administración de pacientes, consultas, gestión financiera, resultados, etc. Para intercambiar información sobre tratamientos se utiliza un mensaje de orden. Los mensajes de orden proporcionan información sobre el conjunto de transacciones referentes a órdenes: petición y realización.

Una orden es una petición de materiales o servicios. Los servicios incluyen: medicación, observaciones clínicas, pruebas, dietas, etc., tanto para pacientes como para un departamento. Por lo tanto los mensajes de ordenes permiten intercambiar información referente a pedidos médicos en general y con variaciones específicas para laboratorio, dietas y farmacia.
El mensaje de HL7 v2.5 que se corresponde con orden de farmacia y tratamiento es el mensaje OMP_009. La estructura de dicho mensaje se muestra a continuación en la Figura 2.13.

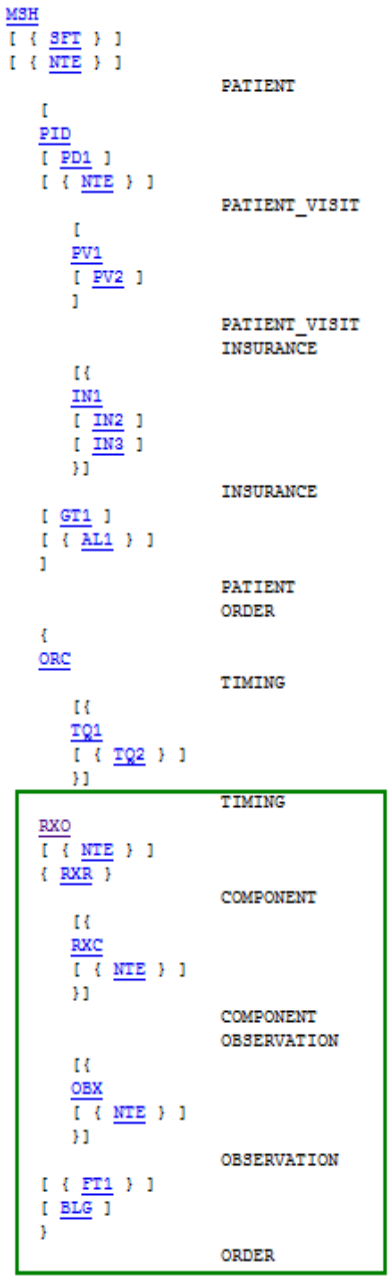

Figura 2.13: Estructura de mensaje de tratamiento de HL7.

Como vemos el segmento RXO, que aparece recuadrado en la Figura 2.13, pertenece a dicho mensaje y es en él donde se especifican los datos necesarios y más importante del tratamiento. Sin embargo para la descripción completa de un tratamiento el segmento deber ir complementado con un segmento de orden (ORC) y uno de tiempo/cantidad (TQ1) que indiquen la cantidad y el intervalo. En este proyecto veremos los segmentos suficientes y necesarios para el tratamiento, por lo tanto solo veremos el segmento RXO.

2.4.3.1 Segmento RXO

Como se ha explicado antes y se muestra en la Figura 2.12, en la estructura de los mensajes del HL7 v2.5 los segmentos están compuestos por campos con un tipo de datos y cada campo puede tener uno o más componentes. En la Figura 1.14 se muestran los campos del segmento RXO.

SEQ	LEN	DT	OPT	RP/#	TBL#	ITEM#	ELEMENT NAME
1	250	CE	C			00292	Requested Give Code
2	20	NM	C			00293	Requested Give Amount - Minimum
3	20	NM	O			00294	Requested Give Amount - Maximum
4	250	CE	C			00295	Requested Give Units
5	250	CE	C			00296	Requested Dosage Form
6	250	CE	O	Y		00297	Provider's Pharmacy/Treatment Instructions
7	250	CE	O	Y		00298	Provider's Administration Instructions
8	200	LA1	O			00299	Deliver-To Location
9	1	ID	O		0161	00300	Allow Substitutions
10	250	CE	O			00301	Requested Dispense Code
11	20	NM	O			00302	Requested Dispense Amount
12	250	CE	O			00303	Requested Dispense Units
13	3	NM	O			00304	Number Of Refills
14	250	XCN	C	Y		00305	Ordering Provider's DEA Number
15	250	XCN	C	Y		00306	Pharmacist/Treatment Supplier's Verifier ID
16	1	ID	O		0136	00307	Needs Human Review
17	20	ST	C			00308	Requested Give Per (Time Unit)
18	20	NM	O			01121	Requested Give Strength
19	250	CE	O			01122	Requested Give Strength Units
20	250	CE	O	Y		01123	Indication
21	6	ST	O			01218	Requested Give Rate Amount
22	250	CE	O			01219	Requested Give Rate Units
23	10	CQ	O			00329	Total Daily Dose
24	250	CE	O	Y		01476	Supplementary Code
25	5	NM	O			01666	Requested Drug Strength Volume
26	250	CWE	O			01667	Requested Drug Strength Volume Units
27	1	ID	O		0480	01668	Pharmacy Order Type
28	20	NM	O			01669	Dispensing Interval

Figura 2.14: Campos de un segmento RXO.

El significado de cada dato de las columnas es el siguiente:

- SEQ: indica la posición ordinal del campo dentro del segmento.

- LEN: máximo número de caracteres que el campo puede contener.
- DT: tipo de dato definido por HL7
- OPT: opcionalidad del campo pudiendo tomar los siguientes valores R (requerido), O (opcional), C (condicional, dependerá de un booleano), X (discontinuado).
- RP: repetición, máximo número permitido de ocurrencias.
- TBL: dominio de valores posibles.
- ITEM: un entero que identifica unívocamente al campo.
- ELEMENT NAME: describe el nombre del campo.

Para más información sobre cada uno de los campos del segmento RXO se puede consultar el anexo2.

2.4.4 Workflow

Uno de los puntos más importantes para conseguir representar el conocimiento de la GPC es conseguir una representación formal de esta. Trasladar un protocolo clínico, que posee texto, decisiones, descripciones, diagramas de flujo, etc., a una representación formal informatizada de una GPC no es una tarea fácil. Sin embargo, las GPC permiten la aplicación de pautas específicas ante situaciones de alta complejidad.
En los últimos años han surgido diferentes metodologías capaces de computarizar las GPC y facilitar de esta forma su aplicación. Algunas de estas metodologías son Asbru[14], Glif[17] o Proforma[6]. Todas estas metodologías proveen de distintos formatos de estructuras para la representación de metas/intenciones, acciones y decisiones de una GPC. Las acciones son las primitivas del modelado que se emplean para representar las tareas que se describen en una GPC. Estas metodologías en lo referente a las tareas que se llevan a cabo en una GPC presentan dos rasgos comunes: descomponen las guías en redes de subtareas y tienen capacidad para expresar estas redes y sus interrelaciones, por ejemplo anidando planes y construyendo así planes de alto nivel.
Asbru[14] es un lenguaje de representación de plan orientado a tiempo, basado en intenciones y especificaciones de planes, sirve para representación de GPC y protocoles en XML.
GLIF3 forma parte del proyecto SAGE [19], y se trata de una arquitectura realizada mediante componentes software e interfaces que se utiliza para realizar sistemas basados en decisiones sobre cuidados en GPC (Riaño 2007).
PROforma [6] es un proyecto desarrollado en el Laboratorio de Computación Avanzada de Investigación de Cáncer en Gran Bretaña. PROforma se un lenguaje de representación formal de conocimiento capaz de capturar la estructura y el contenido de una GPC y está basado en lógica.

Otra metodología que se está utilizando últimamente para modelar GPC computarizadas es el modelado mediante workflows.
Un workflow no sólo está pensado para representar flujos de procesos, sino que también es posible interpretarlos para permitir el guiado automático de los procesos para que tanto personal humano como los sistemas informáticos, estén coordinados en la ejecución de los flujos de los procesos. Se conoce como Sistema de Gestión de Workflow (o Motor de Workflow) a los sistemas ideados para interpretar y automatizar workflows. La automatización de un workflow consiste en la

interpretación de un workflow para ejecutar sus acciones de forma ordenada tal y como vienen especificadas en este. La mayoría de los modelos comerciales de representación de procesos de negocio que se encuentran en el mercado cuentan con herramientas de ejecución de estos workflows que permiten cierta interoperabilidad para facilitar la integración de los modelos de workflow con los sistemas ya existentes.
Basándonos en la tesis de máster "Comprobación retrospectiva del cumplimiento de workflows clínicos" [2] sabemos que los modelos de flujos de trabajo o workflow son formalizaciones útiles para modelar algunos aspectos de las GPC y han demostrado ser una aproximación efectiva para modelar parcialmente una GPC. Por esto, la metodología utilizada en este proyecto para la formalización de las GPC será el workflow.

2.5 HERRAMIENTAS UTILIZADAS.

2.5.1 Java

El proyecto ha sido desarrollado bajo tecnologías Java. Se ha elegido esta tecnología siguiendo varios criterios:

- Es un lenguaje libre orientado a objetos.
- La aplicación será desarrollada como una aplicación gráfica de escritorio y Java permite desarrollo sencillo de interfaces gráficas.
- Existen IDE libres para el desarrollo de aplicaciones bajo Java.
- La herramienta YAWL Editor elegida para integrarla en nuestro proyecto ha sido desarrollada en Java.

2.5.2 Entorno de desarrollo Netbeans

Para el desarrollo del proyecto se ha hecho uso del IDE Netbeans en su versión 6.5. Netbeans IDE es un entorno de desarrollo integrado para programadores, útil para escribir, compilar, depurar y ejecutar programas. Como ya se ha comentado antes el lenguaje utilizado ha sido Java, aunque para Netbeans existen una serie de módulos para poder desarrollar programas en otros lenguajes.
Netbeans IDE permite la instalación de plugins para añadir funcionalidades específicas al IDE, por ejemplo, algunos de los que se ha hecho uso en este proyecto son: el plugin SVN, para poder utilizar un servidor para control de versiones SVN, y el plugin UML para desarrollo de diagramas UML .
Existen otros IDE útiles para el desarrollo de aplicaciones Java como es Eclipse. Se ha utilizado Netbeans por su fácil integración con CH4-UCI y porque integra directamente su propia herramienta de edición de interfaces gráficas facilitando así el desarrollo de aplicaciones de escritorio.

2.5.3. JDOM

JDOM es un API para leer, crear y manipular documentos XML de una manera sencilla y muy intuitiva para cualquier programador en Java. La principal diferencia con DOM del consorcio W3C es que JDOM se creó para usarse específicamente con Java y beneficiarse de las características de

este, por ejemplo de las colecciones.
En este proyecto se utilizarán varios ficheros en formato XML. Por ejemplo, la especificación GEM II viene estructurada en un documentos XML Schema. Además utilizaremos ficheros XML para guardar la información del etiquetado de la GPC.

La API JDOM está formada por 6 paquetes:

- `org.jdom`: formado por clases para representar los componentes de un documento XML.
- `org.jdom.adapters:` formado por clases para crear interfaces con varias implementaciones de DOM.
- `org.jdom.filter`: tiene clases para permitir filtro de nodos de un documento basados en nombre, valor, u otros aspectos.
- `org.jdom.input`: tiene las clases necesarias para construir documentos XML.
- `org.jdom.output`: tiene las clases para interpretar la salida de documentos XML.
- `org.jdom.transform`: formado por clases para ayudar a transformaciones.
- `org.jdom.xpath`: contienen solo una clase Xpath, útil para realizar llamadas sobre nodos Xpath JDOM.

Las clases para hacer las operaciones básicas con un documento XML son las clases que se encuentran en el paquete `org.jdom`.
Con la clase `Document` se puede representar un documento XML. La clase `Document` tienen un elemento raíz o root llamado `rootElement` que representa la etiqueta padre del documento y es representado por la clase `Element` del mismo paquete que `Document`.
La clase `Element` representa una etiqueta XML. Provee de métodos que permiten al usuario obtener y manipular los elementos hijos de esta etiqueta, que a su vez serán representado por la clase `Element`. Con la clase `Element` también es posible manipular los atributos del elemento y el contenido del texto del elemento o etiqueta. Estas son algunas de las funcionalidades de esta API que se han usado en este proyecto para la gestión de los ficheros XML y XML Schema que se utilizan.

2.5.3.1 XML

XML (***Ex**tensible **M**arkup **L**anguage*) es un metalenguaje extensible de etiquetas desarrollado por el W3C y permite definir la gramática de lenguajes específicos utilizados para distintas necesidades. Pero XML no solo tiene aplicación en Internet, también se utiliza como un estándar para el intercambio de información estructurada entre diferentes plataformas, ya sean bases de datos, editores de texto, hojas de cálculo o cualquier plataforma en la que sea conveniente su utilización.

En XML se pueden diferenciar dos tipos de elementos: el marcado y los datos de carácter. El marcado es el texto incluido entre los caracteres menor que “<” y mayor que “>” y se denominan etiqueta y son las partes del documento que tiene que entender el procesador XML. Los datos de carácter son el resto del documento que no corresponden a las etiquetas.

El documento XML se compone de uno o más elementos cuyos límites están delimitados por las distintas etiquetas. Los elementos pueden pueden tener contenido, en este casó habrá una etiqueta de apertura y otra de cierre, o no tener contenido, en este caso habrá una etiqueta. Los elementos pueden tener ninguno o varios atributos. En las Figura 2.15 y 2.16 se muestra la estructura de un

elemento con contenido y sin contenido respectivamente.

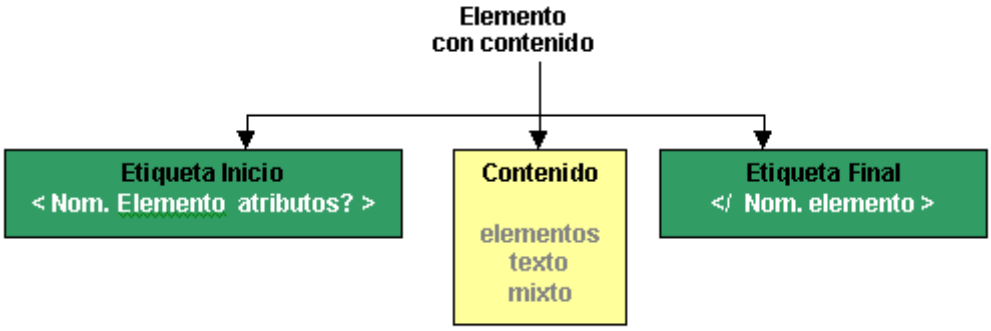

Figura 2.15: Estructura del elemento XML con contenido. [20]

En los elementos con contenido, como se indica en la Figura 2.15, el contenido pueden ser más elementos, texto, o ambas cosas. Los elementos que se encuentren en la parte de contenido de otro elemento serán elementos hijos directos o hijos de primer nivel de este.

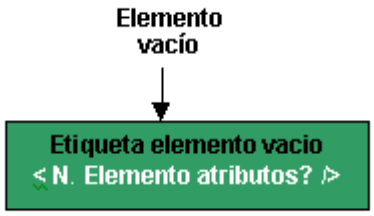

Figura 2.16: Estructura del elemento XML vacío. [20]

2.5.3.2 XML Schema

Cuando se trabaja con documentos XML, debemos comprobar que este documento está bien formado comprobando que las etiquetas se empiecen y terminen, se utilicen caracteres adecuados, etc. Aparte de estar bien formado el documento debe de ser válido para el caso concreto, es decir, las etiquetas y atributos de estas deben ser las esperadas y la estructura jerárquica de las etiquetas debe ser la correcta.

Los DTD son los documentos que definen las etiquetas válidas dentro de un documento XML. Un documento XML normalmente tiene asociado una DTD. El problema de las DTD es, que no son documentos XML en sí mismos, no son demasiado extensibles y además, no nos permite establecer validaciones más complejas que la propia existencia y orden de los elementos y atributos. Por esto los DTD han ido evolucionando y han surgido los XML Schema.

Los ficheros XML Schema, actualmente con extensión xsd, se concibieron como una alternativa a los DTD, intentando superar sus puntos débiles y buscar nuevas capacidades a la hora de definir estructuras para documentos XML. Sirven para definir qué elementos puede contener un documento XML, cómo están organizados y qué atributos y de qué tipo pueden tener sus elementos.

La principal aportación de XML Schema es el gran número de tipos de datos que incorpora, así se aumentan las posibilidades y funcionalidades de aplicaciones de procesado de datos incluyendo tipos de datos complejos como fechas, números y strings.

XML Schema fue diseñado completamente alrededor de namespaces y soporta tipos de datos típicos de los lenguajes de programación, como también tipos personalizados simples y complejos.

Cada Namespace contiene elementos y atributos que están estrechamente relacionados con el Namespace. Así, a la hora de definir un elemento o un atributo de un Namespace, siempre se creará una conexión entre los diferentes campos de éste. Además, esta forma de trabajar nos permite relacionar elementos que no están en el mismo Namespace.

En este proyecto, utilizaremos XML Schema para comprobar antes de cargar un fichero xml que está bien formado y para definir los elementos de cada terminología y estándar.

2.5.4. JTree y el paquete javax.swing.jtree.

Las terminologías y estándares clínicos utilizadas en este proyecto se pueden representar de forma jerárquica como una taxonomía. Estas taxonomías van a ser representadas en la interfaz gráfica de la aplicación para hacer más fácil al usuario asimilar y entender de manera correcta cada una de estas. La forma más adecuada de representarlas y gestionar de manera gráfica en Java conceptos jerárquicos es haciendo uso del componente `JTree` de `javax.swing`. El paquete `javax.swing.jtree` proporciona clases e interfaces para gestionar adecuadamente el componente `JTree`

JTree es una clase Java que permite mostrar datos de forma jerárquica, pero realmente no contiene los datos, simplemente muestra una vista de ellos. Un ejemplo de la vista de un JTree se muestra en la Figura 2.17:

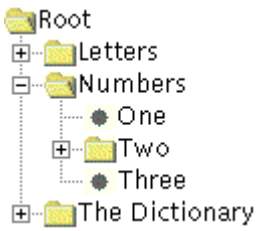

Figura 2.17: Vista de un JTree con un modelo por defecto.

Los datos se muestran verticalmente, como muestra la Figura 2.17. Cada fila del árbol contiene exactamente un elemento de datos llamado nodo. Cada árbol tiene un elemento raíz que en la figura se representa con Root.

`JTree` pertenece a la librería gráfica de Java Swing. En Swing cada tipo de componente tiene asociado un tipo particular de modelo que será el que contenga los datos del componente. Swing provee estos tipos de modelo como interfaces y además proporciona modelos por defecto que

proporcionan funcionalidad básica y lista para usarse. La ventaja de que los modelos de los componentes sean implementados como interfaces, es que el desarrollador puede implementar un modelo adecuado para un determinado componente sin tener que reestructurar sus datos según requerimientos de Swing.

En el caso del JTree el atributo treeModel será el encargado de representar el modelo de datos. Este atributo debe pertenecer a una clase que implemente la interfaz `TreeModel`, y en este proyecto se hará uso de la clase por defecto que proporciona Swing y que implementa esta interfaz, la clase `DefaultTreeModel`.

`DefaultTreeModel` proporciona la funcionalidad básica y lista para usar el modelo de datos de un JTree. Esta clase a su vez tiene un elemento root que representará el nodo raíz del `JTree` y debe implementar la interfaz `TreeNodes`. La clase por defecto que implementa esta interfaz, es la clase `DefaultMutableTreeNode`. Esta clase a su vez tendrá un atributo children que será un conjunto de objetos de este mismo tipo, representado así la jerarquía.

La clase `JTree` tampoco es la encargada de dibujar la celda de los nodos del árbol, para esto se utiliza el atributo `cellRenderer`. Este atributo debe pertenecer a una clase que implemente la interfaz `TreeCellRenderer` la cual define los requerimientos para cada nodo de un árbol. Al igual que en los casos anteriores existe una clase por defecto, la clase `DefaultTreeCellRenderer` que implementa los métodos necesarios para mostrar los nodos de un árbol como se muestran en la Figura 2.17. En este caso no se utilizará esta clase por defecto ya que se le quiere dar un aspecto propio a cada tipo de nodo del árbol, dependiendo del tipo de elemento que representen, GEM II, CH4 o HL7. Se implementa la clase `MyTreeCellRenderer`.

En la Figura 2.18 se muestra un diagrama de clases donde se representan las clases comentadas del `JTree`.

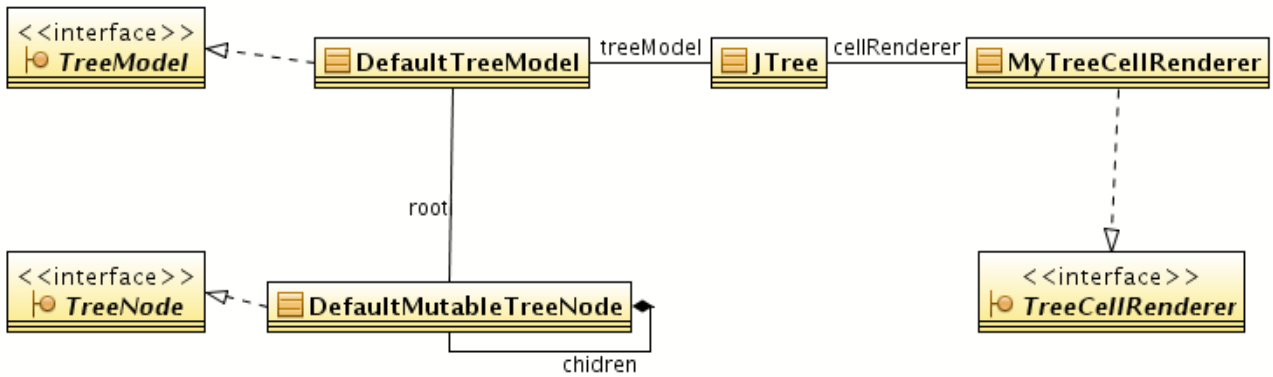

Figura 2.18: Diagrama de clases de los principales componentes de un JTree.

2.5.5. Control de versiones: SVN

SVN es un sistema de control de versiones usado para que varios desarrolladores puedan trabajar en un mismo proyecto de forma ordenada y para poder guardar versiones de un mismo proyecto conforme se va avanzando en él. SVN. Tiene una arquitectura cliente servidor. En el servidor se

monta un repositorio SVN y es donde se van registrando los cambios y con el cliente SVN es posible bajarse una copia local de alguna revisión, y subir los cambios realizados. Tanto a la hora de subir como de bajar el proyecto, SVN es capaz de detectar conflictos y avisar al usuario para que elija la operación que quiera en cada caso.
En el IDE Netbeans se puede instalar un plugin para integrar Netbeans con SVN. Este plugin ha sido utilizado en el desarrollo del proyecto para llevar un control de las versiones del proyecto desde Netbeans.

2.5.6. Plugin de Netbeans para diseño UML

Para el creación de los diagramas UML que se mostrarán en el apartado tercero de la memoria se ha utilizado el plugin de Netbeans 1.6 que se puede descargar e instalar desde el mismo IDE. Con este plugin se permite la creación de diagramas de secuencia, casos de uso, diagramas de clases y diagramas de estado.

2.5.7. YAWL Editor

Al elegir el formalismo workflow para el modelado de la GPC se debe desarrollar o elegir un editor de workflow para integrarlo en nuestra aplicación. El editor elegido ha sido YAWL Editor.
YAWL (Yet Another Markup Language) [7] es un lenguaje de workflow basado en patrones. Este lenguaje tiene la ventaja de que está soportado por un sistema de software que incluye un motor de ejecución y un editor gráfico.
YAWL Editor es el editor de YAWL, es una aplicación libre y gratuita para editar gráficamente los flujos de trabajo. YAWL Editor permite la creación y modificación de workflows. En la implementación de este editor se hace uso de la librería gráfica `JGraph` y ha sido desarrollado de tal manera que `JGraph` es el corazón de la aplicación, YAWL Editor se podría considerar como una extensión de `JGraph`.
Otra ventaja a la hora de usar YAWLEditor es que se ha desarrollado de manera que ofrece un punto importante de extensibilidad para interconectar aplicaciones externas con el motor de ejecución workflow. Esto permite que se pueda interactuar con otras aplicaciones notificando cambios en tareas.

3 DISEÑO Y RESOLUCIÓN

3.1 ANÁLISIS DE LA APLICACIÓN

La herramienta de este proyecto será desarrollada como una aplicación de escritorio. El principal motivo de desarrollarla como una aplicación de escritorio es la futura integración con CH4-UCI, que es otra aplicación de escritorio y que la versión que se desarrolla es un prototipo.

3.1.1 Análisis de requisitos

A continuación se hará un análisis informal de requisitos del proyecto. Dividiremos lo requisitos en funcionales y no funcionales.

Requisitos funcionales: son aquellos requisitos que especifican una funcionalidad que debe realizar un sistema o componente. Los requisitos funcionales de este sistema son los siguientes:

- El sistema permitirá visualizar GPC en formato HTML almacenadas localmente o descargadas de una url.
- El sistema permitirá crear nuevos modelos formales para una GPC.
- Un modelo formal final estará compuesto por un workflow y una representación estructurada de la GPC.
- El sistema permitirá crear el modelo formal de forma incremental siguiendo dos pasos principales:
  1. Construcción de la representación estructurada de la GPC.
  2. Construcción del workflow a partir de la representación estructurada.
- La representación estructurada de la GPC se representará gráficamente en un árbol.
- La representación estructurada de la GPC estará formada por elementos de los estándares y terminologías utilizados.
- Los estándares y terminologías utilizados para construir la representación estructurada serán GEM II, CH4 y HL7.
- Los elementos de la representación estructurada se utilizarán para etiquetar parte de texto de la GPC.
- Los elementos de la representación estructurada estarán enlazados con una parte de texto de la GPC que etiqueten.
- La representación estructurada se construirá de forma incremental siguiendo los siguientes pasos:
  1. Se etiquetarán elementos de GEM II con partes de la GPC.
  2. Se etiquetarán elementos de CH4 enlazándolos con elementos GEM II añadidos en el paso anterior y la parte de texto que corresponda.
  3. Se etiquetarán elementos de HL7 enlazándolos con elementos GEM II añadidos en el paso anterior y la parte de texto que corresponda.
- El sistema mostrará el texto etiquetado de la GPC de distinto color.
- Todos los elementos etiquetados pasarán a formar parte de la representación estructurada.
- El sistema permitirá eliminar elementos de la representación estructurada.
- El sistema permitirá cambiar el texto etiquetado de un elemento de la representación estructurada.

- El sistema permitirá la trazabilidad entre los elementos de la representación estructurada y el texto de la GPC original.
- Al etiquetar un elemento RXO de HL7 el sistema permitirá al usuario introducir los valores de los campos de este.
- El sistema permitirá al usuario editar los valores de los campos de un elemento RXO.
- El sistema permitirá introducir atributos a cada uno de los elementos GEM, CH4 o HL7 añadidos a la representación estructurada.
- El sistema permitirá eliminar atributos de cada uno de los elementos añadidos a la representación estructurada.
- El sistema permitirá construir un workflow y enlazar las tareas de este con elementos de la representación estructurada construida en el paso anterior.
- El sistema permitirá la trazabilidad entre las tareas del workflow y los elementos de la representación estructurada y así a su vez con la GPC original.
- El sistema mostrará en cuatro pestañas los pasos para conseguir el modelo formal.
- El sistema permitirá volver hacia atrás en los pasos sin perder información.
- El sistema añadirá automáticamente tareas al workflow cada vez que se etiqueten elementos CH4 o HL7.
- El sistema permitirá enlazar tareas del workflow añadidas por el usuario con elementos añadidos de la representación estructurada.
- El sistema permitirá romper enlaces entre tareas del workflow y los elementos de la representación estructurada.
- El sistema integrará la herramienta YAWL Editor para la edición del workflow.
- El sistema permitirá guardar el modelo formal que se construya.
- El sistema guardará la representación estructurada en un fichero xml.
- El sistema guardará el workflow en un fichero yawl.
- El sistema guardará en el fichero xml una relación al fichero yawl
- El sistema permitirá cargar un modelo con el que se ha trabajado anteriormente.

Requisitos no funcionales: son aquellos requisitos que describen características requeridas del sistema, del proceso de desarrollo, del servicio prestado o de cualquier otro aspecto del desarrollo, que señala una restricción del mismo.

- La herramienta se ejecutará en cualquier sistema operativo bajo el jdk 1.5 o versiones posteriores.

3.1.2 Casos de uso

3.1.2.1 Actores del sistema

Esta aplicación irá dirigida a usuarios de GPC, por lo tanto los usuarios del sistema serán por lo general médicos. Este sistema solo tendrá un actor que será el usuario experto o médico.

Figura 3.1: Usuarios del sistema.

3.1.2.2 Crear modelado

El usuario al iniciar el sistema podrá o bien cargar un modelado con el que ya haya trabajado o bien crear un nuevo modelado. Al crear un nuevo modelado tendrá que cargar la GPC con la que quiere trabajar.

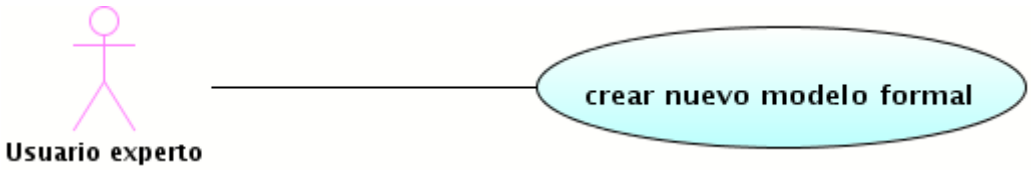

Figura 3.2: Caso de uso “crear nuevo modelo formal”,

La descripción del caso de uso “crear nuevo modelo formal” es la siguiente:

Caso de Uso	**Crear un nuevo modelo formal**
Objetivo	Definir una nueva formalización de la GPC
Actores	Usuario experto.
Precondiciones	El usuario ha iniciado el sistema correctamente.
Escenario principal	1. El usuario experto carga una GPC de una url. 2. El sistema muestra el texto de la GPC en pantalla. 3. El usuario pulsa el botón para crear un nuevo modelado. 4. El sistema muestra la pantalla para introducir el nombre de la formalización. 5. El usuario introduce un nombre de modelado y acepta. 6. El sistema inicializa la representación estructurada y se crea un workflow inicial con las tareas de inicio y fin.
Extensiones	1a. El usuario puede cargar la GPC en formato HTML teniéndola guardada localmente. 1.La extensión continua por el paso 2 del escenario principal. 5a. El usuario cancela la introducción del nombre de la formalización. 1.Termina el caso de uso.

3.1.2.3 Gestión del elementos GEM II

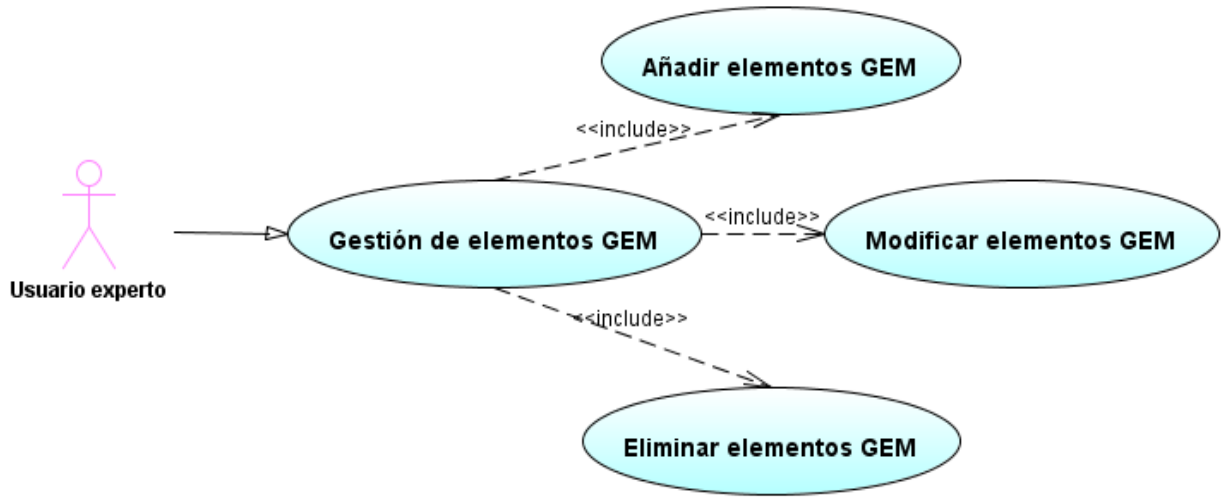

Figura 3.3: Caso de uso “Gestión de elementos GEM II”.

La descripción de los tres casos de uso son los siguientes:

Caso de Uso	**Añadir elementos Gem**
Objetivo	Añadir cualquier elemento del modelo GEMII a la representación estructurada.
Actores	Usuario experto.
Precondiciones	Se ha iniciado o cargado una formalización y haya una GPC mostrada en el panel.
Escenario principal	1. El usuario selecciona en el árbol que representa los elementos GEM II añadidos el elemento al cual le quiere añadir un elemento como hijo. 2. El usuario selecciona en el árbol GEM II el tipo de elemento GEM II que quiere añadir. 3. El usuario selecciona en la GPC la parte de texto que el usuario quiere etiquetar con el elemento GEM II a añadir. 4. El usuario pulsa el botón para insertar el elemento. 5. El sistema comprueba que todos los pasos han sido realizados y añade el elemento a la representación estructurada visualizándolo en el árbol.

Extensiones	5a. El sistema comprueba que el nodo que se quiere añadir no puede ser hijo del nodo seleccionado en el árbol del elementos GEM II añadidos. 1. El sistema avisa al usuario que no es posible insertar ese elemento como hijo del elemento seleccionado. 2. Termina el caso de uso. 5b. El sistema comprueba que no se ha realizado alguno de los pasos. 1.El sistema avisa al usuario y termina el caso de uso.

Caso de Uso	**Modificar elementos GEM**
Objetivo	Modificar el texto etiquetado por cualquier elemento GEM II de la representación estructurada.
Actores	Usuario experto.
Precondiciones	La representación estructurada tiene al menos un elemento GEM II.
Escenario principal	1. El usuario selecciona el elemento GEM que quiere modificar. 2.El usuario selecciona en la GPC la parte de texto que el usuario quiere etiquetar con el elemento GEM II a añadir. 4. El usuario pulsa el botón para modificar el texto etiquetado. 5. El sistema comprueba que todos los pasos han sido realizados y pide confirmación de la operación. 6. El usuario acepta. 7. El sistema modifica el texto etiquetado por el elemento.
Extensiones	7a. El usuario cancela la operación. 1. Termina el caso de uso.

Caso de Uso	**Eliminar elementos GEM**
Objetivo	Eliminar cualquier elemento GEM II de la representación estructurada.
Actores	Usuario experto.

Precondiciones	La representación estructurada tiene al menos un elemento GEM II.
Escenario principal	1. El usuario selecciona el elemento GEM II que quiere eliminar. 2. El usuario pulsa el botón de eliminar. 3. El sistema muestra un mensaje de confirmación. 4. El usuario acepta eliminar el elemento. 5. El sistema elimina el elemento del modelo y lo hace visible en el árbol.
Extensiones	4a. El usuario cancela eliminar el elemento. 1.Termina el caso de uso.

3.1.2.4 Gestión de elementos CH4

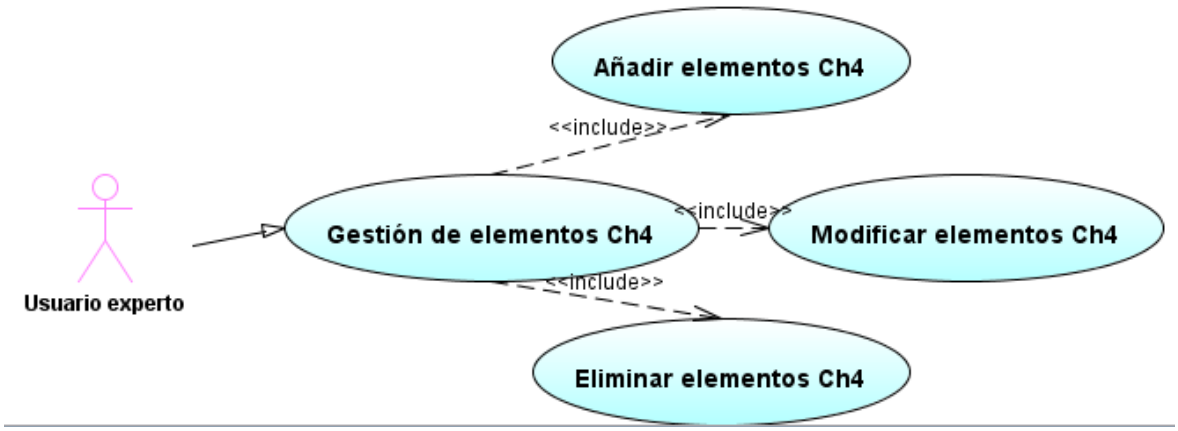

Figura 3.4: Caso de uso “Gestión de elementos CH4”.

La descripción del caso de uso “añadir elementos CH4” es la siguiente:

Caso de Uso	**Añadir elementos CH4**
Objetivo	Añadir cualquier elemento de la terminología CH4 a la representación estructurada de la GPC.
Actores	Usuario experto.
Precondiciones	Existe al menos un elemento GEM añadido a la representación estructurada.
Escenario principal	1. El usuario selecciona el elemento padre del árbol que representa los elementos CH4 añadidos.

	2. El usuario selecciona el tipo de elemento de la terminología CH4 que se quiere añadir. 3. El usuario selecciona el elemento GEM II con el que se va a relacionar el elemento CH4 a añadir. 3. El usuario selecciona la parte del texto etiquetado con el elemento GEM II que se relaciona con el elemento a añadir. 4. El usuario pulsa el botón para insertar el elemento CH4. 5. El sistema comprueba que todos los pasos han sido realizados y añade el elemento a la representación estructurada y lo hace visible en el árbol. 6. El sistema añade automáticamente una tarea relacionada con este elemento al workflow.
Extensiones	5a. El sistema comprueba que no se ha realizado alguno de los pasos. 1.El sistema avisa al usuario que no se puede insertar el elemento y termina el caso de uso.

El caso de uso de eliminar elementos CH4 y modificar elementos CH4 es el mismo que el de eliminar elementos GEM solo que el elemento a seleccionar tendrá que ser del árbol de elementos CH4 añadidos.

3.1.2.5 Gestión de elementos HL7

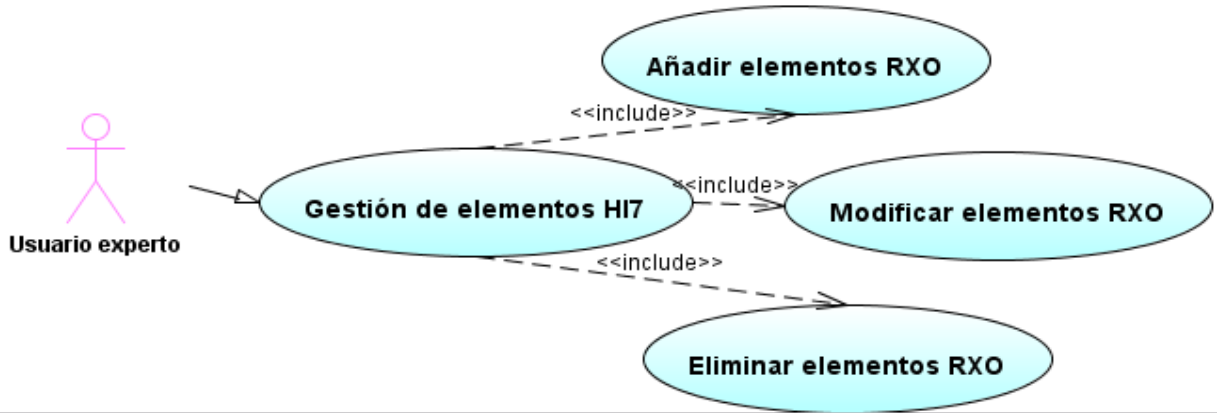

Figura 3.5: Caso de uso “Gestión de elementos GEM HL7”.

La descripción del caso de uso “Añadir elementos RXO” es la siguiente:

Caso de Uso	**Añadir elementos RXO**
Objetivo	Añadir un elemento RXO a la representación estructurada.
Actores	Usuario experto.
Precondiciones	Existe al menos un elemento GEM añadido a la representación estructurada.
Escenario principal	1. El usuario selecciona el nodo padre del árbol que representa los elementos HL7 añadidos. 2. El usuario selecciona el tipo de elemento de HL7 que se quiere añadir, será RXO. 3. El usuario selecciona el elemento GEM II de la representación estructurada con el que se va a relacionar el elemento RXO a añadir. 3. El usuario selecciona la parte del texto que se relaciona con el elemento a añadir. 4. El usuario pulsa el botón para insertar el elemento RXO. 5. El sistema comprueba que todos los pasos han sido realizados y muestra una pantalla de formulario para rellenar los campos del elemento RXO. 6. El usuario rellena los campos y acepta. 7. El sistema añade el elemento a la representación estructurada y lo visualiza en el árbol. 8. El sistema añade automáticamente una tarea relacionada con este elemento al workflow.
Extensiones	5a. El sistema detecta que alguno de los pasos no se ha realizado adecuadamente. 1.Termina el caso de uso. 6a. El usuario cancela el formulario de inserción. 1.Termina el caso de uso.

Al igual que pasaba con el caso de uso de eliminar elementos CH4, los casos de uso de eliminar elementos RXO y modificar elementos RXO serán el mismo que el de eliminar elementos GEM solo que el elemento a seleccionar tendrá que ser del árbol de elementos HL7 añadidos.

3.1.2.6 Gestión del workflow.

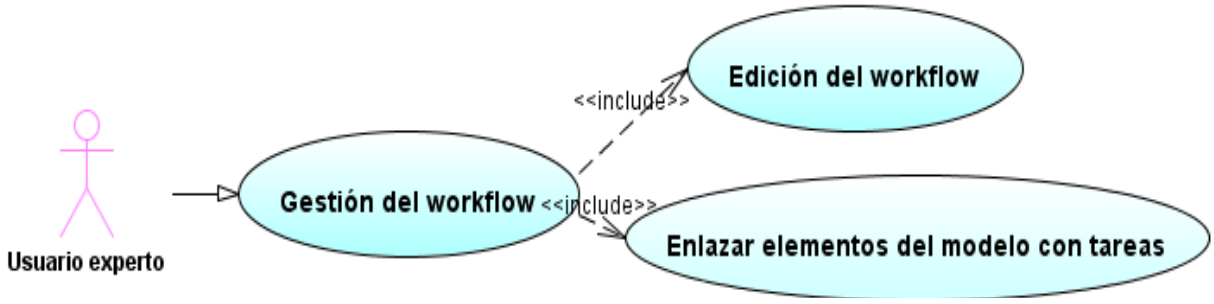

Figura 3.6: Caso de uso “Gestión del workflow”.

La edición del workflow ya está implementada por el YAWL Editor que se integrará en la aplicación. Tendremos que implementar el caso de uso “Enlazar elementos del modelo con tareas.”

La descripción del caso de uso “Enlazar elementos del modelo con tareas” es la siguiente:

Caso de Uso	**Enlazar elementos del modelo con tareas**
Objetivo	Enlazar una tarea del workflow con un elemento de la representación estructurada.
Actores	Usuario experto.
Precondiciones	Exista al menos un elemento en la representación estructurada.
Escenario principal	1. El usuario selecciona la tarea del workflow y el elemento en el modelo. 2.El usuario pulsa el botón para enlazar. 3.El sistema enlaza la tarea y el elemento.
Extensiones	3a. El elemento ya tiene una tarea enlazada, por lo tanto el sistema muestra un mensaje de confirmación para cambiar el enlace del elemento. 1.El usuario acepta y sigue por el paso 3 del escenario principal.

3.1.2.7 Gestión de atributos de los elementos del modelo

En todos los elementos añadidos a la representación estructurada del documento se podrán añadir atributos con los nombres y valores que el usuario desee. Para gestionar esta tarea se destacan 3

casos de uso: añadir los atributos, modificar los atributos y eliminar los atributos.

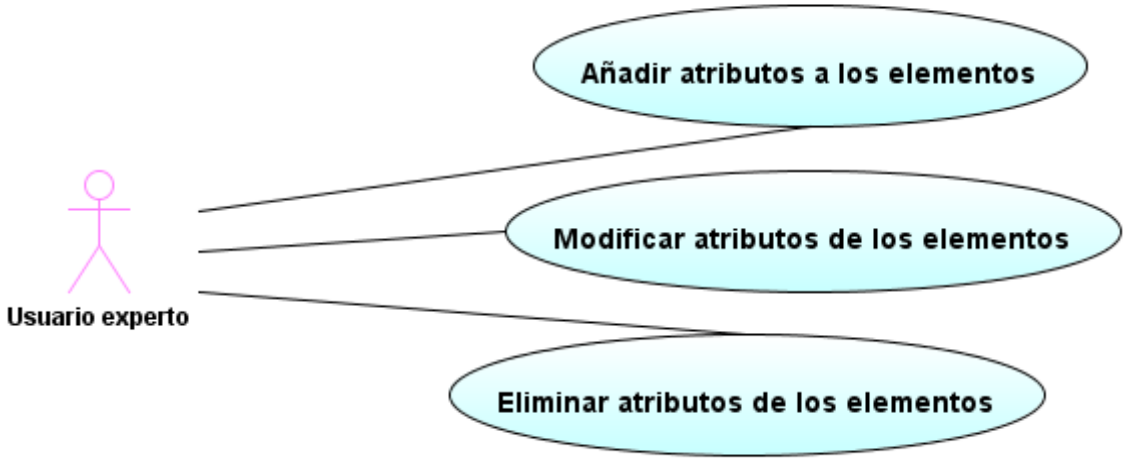

Figura 3.7: Otros caso de uso para gestionar la formalización.

La descripción de los casos de uso de la Figura 3.7 son los siguientes:

Caso de Uso	**Añadir atributos a los elementos**
Objetivo	Añadir un atributo con un valor a un elemento de la representación representación estructurada.
Actores	Usuario experto.
Precondiciones	Existe al menos un elemento en la representación estructurada.
Escenario principal	1. El usuario selecciona el elemento al cual le quiere añadir el atributo y pulsa el botón para añadir el atributo. 2. El sistema muestra el cuadro de dialogo para insertar el nombre y valor del atributo. 3. El usuario introduce el nombre y valor del atributo y pulsa aceptar. 4. El sistema muestra en la tabla de atributos del elemento el nuevo atributo.
Extensiones	3a. El usuario cancela la introducción de los valores. 1.Termina el caso de uso.

Caso de Uso	**Modificar atributos de los elementos**
Objetivo	Modificar un atributo de un elemento añadido a la representación representación estructurada.

Actores	Usuario experto.
Precondiciones	Exista al menos un elemento en la representación estructurada.
Escenario principal	1. El usuario selecciona el elemento al cual le quiere modificar el atributo y selecciona el atributo en su tabla de atributos. 2.El usuario pulsa el botón para modificar el atributo. 3.El sistema muestra el cuadro de dialogo para modificar el nombre y valor del atributo. 4.El usuario introduce los nuevos valores de nombre y valor del atributo y pulsa aceptar. 5.El sistema muestra en la tabla de atributos el atributo modificado.
Extensiones	4a. El usuario cancela la introducción de los valores. 1.Termina el caso de uso.

Caso de Uso	**Eliminar atributos de los elementos**
Objetivo	Eliminar un atributo de un elemento añadido al modelo.
Actores	Usuario experto.
Precondiciones	Existe al menos un elemento en la representación estructurada.
Escenario principal	1. El usuario selecciona el elemento al cual le quiere modificar el atributo y selecciona el atributo en su tabla de atributos. 2.El usuario pulsa el botón para eliminar el atributo. 3.El sistema muestra el cuadro de dialogo para confirmar la eliminación. 4.El usuario acepta. 5.El sistema elimina el atributo del modelo y visualiza los cambios en la tabla.
Extensiones	3a. El usuario cancela la eliminación. 1.Termina el caso de uso.

3.1.2.8 Guardar y cargar formalización

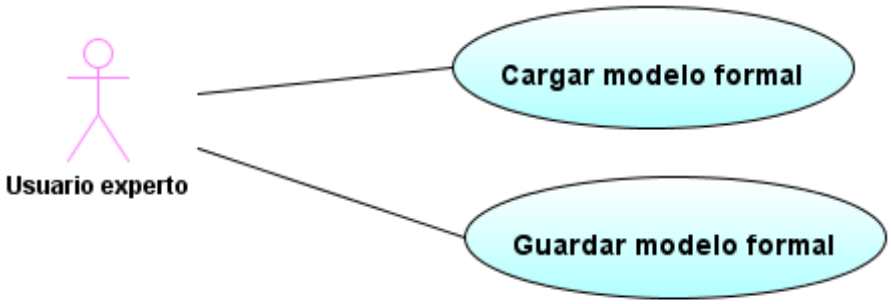

Figura 3.8: Casos de uso cargar y guardar modelo formal.

La descripción de los casos de uso de la Figura 3.8 son las siguientes:

Caso de Uso	**Guardar modelo formal.**
Objetivo	Guardar el modelo formal que se ha creado de una GPC.
Actores	Usuario experto.
Precondiciones	Exista un modelo creado y cargado en la aplicación.
Escenario principal	1. El usuario pulsa el botón para guardar. 2.El sistema muestra un cuadro de dialogo para que el usuario elija el nombre del fichero yawl para guardar el workflow. 3.El usuario introduce el nombre del fichero y la ruta para guardarlo y acepta. 4.El sistema muestra un cuadro de dialogo para que el usuario elija el nombre del fichero xml para guardar la representación estructurada. 5.El usuario introduce el nombre del fichero y la ruta para guardarlo y acepta. 6.El sistema guardar en el xml el modelo con los datos de los elementos y la relación con el fichero yawl.
Extensiones	1a. El sistema muestra un mensaje para confirmar que se desean guardar los cambios y machacar la información de los ficheros

	anteriores. 1.El usuario acepta. 2.Termina el caso de uso.

Caso de Uso	**Cargar modelado**
Objetivo	Cargar un modelo formal.
Actores	Usuario experto.
Precondiciones	Exista un modelo formal guardado anteriormente.
Escenario principal	1. El usuario pulsa el botón para cargar la formalización 2. El sistema muestra un cuadro de dialogo para que el usuario elija el nombre del fichero xml que contienen la representación estructurada a cargar. 3. El usuario selecciona el fichero a cargar. 4. El sistema tramita el fichero y carga la GPC, el la representación estructurada y el workflow.
Extensiones	4a. El sistema no encuentra la GPC en la ruta indicada o no coincide con el modelo. 1. El sistema da la opción al usuario para que la cargue de un fichero local o de una url. 2. El usuario selecciona la GPC de una url o de un fichero local. 3. El sistema carga la GPC, el modelo del árbol y el workflow. 5a. El sistema no encuentra el fichero yawl en la ruta especificada 1. El sistema da la opción para que se cargue el fichero. 2. El sistema carga la GPC, el modelo del árbol y el workflow.

3.1.3 Modelo conceptual

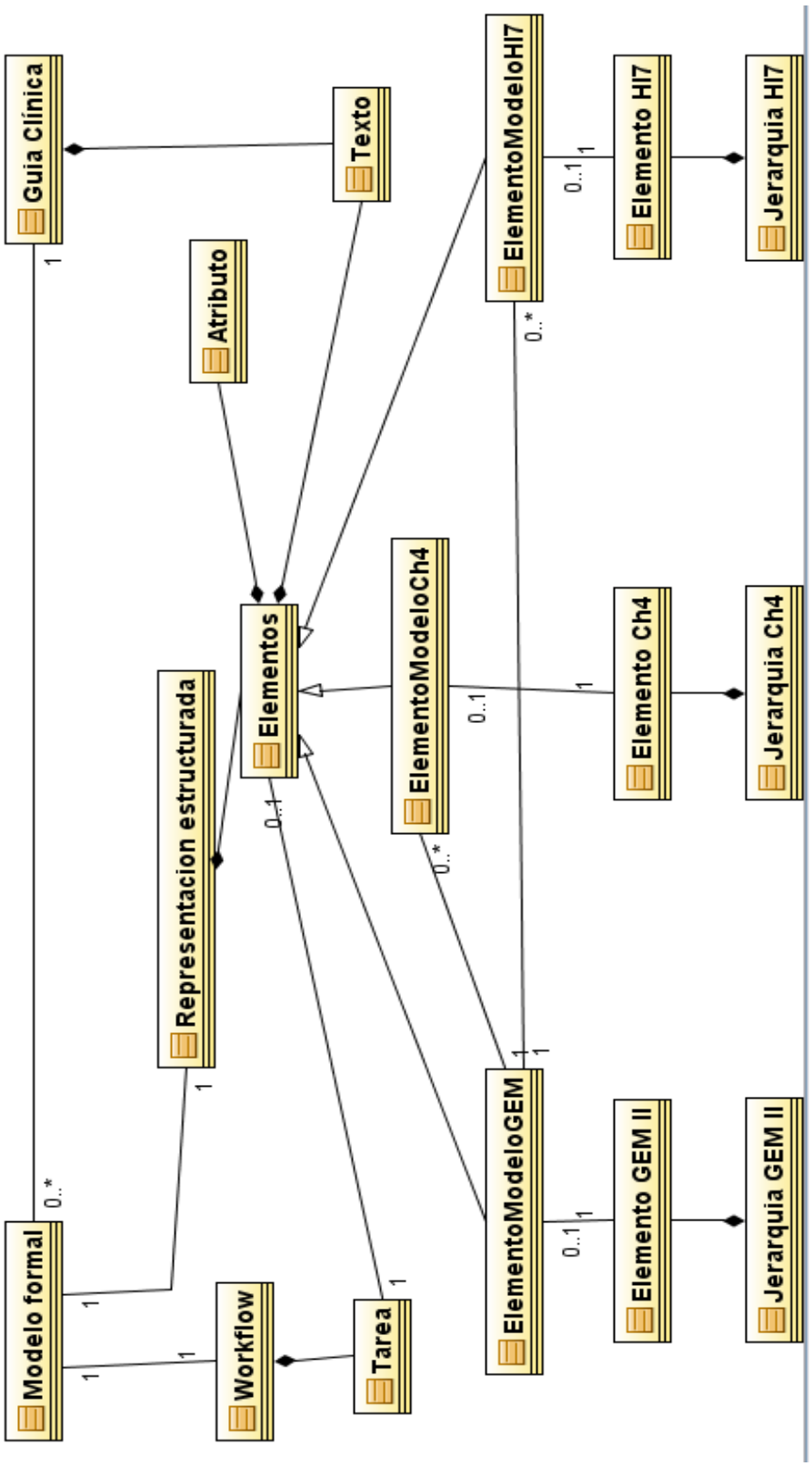

Figura 3.9: modelo conceptual del sistema.

En la Figura 3.9 se muestra el modelo conceptual del sistema. En el sistema se cargarán tres jerarquías que contendrán los elementos de cada uno de los estándares y terminologías que se utilizarán para etiquetar la GPC. El modelo GEM II se representa en el diagrama con el concepto

JerarquiaGEMII y contendrá los elementos del modelo GEMII estructurados jerárquicamente. La jerarquíaCh4 estará formada por los elementos de la terminología CH4. Por último, la jerarquíaHL7 estará formada por los segmentos HL7 disponibles para etiquetar, en esta primera versión del sistema solo tendrá un tipo de elementos, el RXO.
El sistema a su vez tiene una clase principal "modelo formal" la cual representa la formalización que se construye de la GPC con la que se trabaje. El modelo tiene asociada la GPC con la cuál se trabaja y a la cual hace referencia dicho modelo, una Representacion estructurada que representará los elementos utilizados para etiquetar la GPC y el Workflow para representar la GPC como un formalismo y obtener así un modelo formal. La Representación estructurada y el Workflow, que representa el modelo formal, están relacionados mediante asociaciones entre los elementos de la representación estructurada las tareas de workflow.
El modelo que representa los elementos que se han ido etiquetando en la GPC está compuesto por elementos que podrán ser elementosGEM, elementosCh4 o elementosHL7, los elementos estarán asociados al tipo de elemento al cual pertenecen de su jerarquía, por eso tienen una asociación a la clase que representa un elemento de su jerarquía. Los elementos del modelo serán utilizados para etiquetar partes de texto de la GPC por eso tendrán una asociación a una parte del texto de la GPC y podrán tener un conjunto de atributos con sus respectivos valores. Cada elemento representará un elemento xml con sus respectivos atributos en el fichero xml como atributos del elemento y con el texto de la GPC como contenido a la hora de guardar la formalización.

El workflow asociado al modelo formal corresponden al workflow que se irá formando y que finalmente representará la GPC como un formalismo. Un workflow está compuesto por una serie de tareas, estas tareas podrán estar asociadas a elementos que forman el modelo y que han sido marcados anteriormente en la GPC. El workflow se guardará en un fichero yawl y el fichero xml tendrá una correspondencia a este.

3.2 INTEGRACIÓN CON YAWL Editor.

YAWL Editor es el editor de workflow que se utilizará en nuestro sistema para editar y formar el workflow que representará el formalismo de la GPC.

Para la integración de YAWLEditor en nuestro programa se ha descargado el código del paquete org.yawlfondation.yawl.editor ya que se necesitarán hacer algunas modificaciones y se ha agregado a nuestro código con el paquete del mismo nombre. Ademas se han descargado las API necesarias para el correcto funcionamiento de YAWL Editor. Todo ha sido descargado de sourceforge de la dirección: https://yawl.svn.sourceforge.net/svnroot/yawl. La licencia del código es LGPL

El paquete YAWLEditor a su vez contienen 9 paquetes y dos clases principales. Las clases principales son YAWLEditor que representa el editor y en la cual tendremos que hacer una serie de modificaciones y la clase TestYAWLEditor. Los paquetes que se incluyen son: actions, analyser, data, elements, foundations, net, reductionrules, resources, resourcing, specification, swing y thirdparty.

El paquete elements es el que implementa cada uno de los elementos que pueden componer el workflow. Este paquete se divide en dos paquetes, el paquete model y el paquete view. El paquete model implementa el modelo de los elementos y el paquete view implementa las vistas de los elementos. Las clases del paquete modelo son las que se representan en el diagrama de la Figura 3.10. Como se puede ver se hace uso de dos clases de la librería JGraph la clase

`DefaultGraphCell` y la clase `DefaultPort` representadas en la figura dentro del cuadro verde. Las clase `YAWLVertex` va a representar la celda de una tarea y la clase `VertexLabel` representará el nombre de una tarea.

La clase `VertexLabel` será la clase que utilizaremos para enlazar el elemento del modelo formal con la tarea del workflow ya que esta clase es la clases más general que representa el la tarea de un workflow. En realidad esta tarea representa le etiqueta de la tarea pero a su vez tiene a asociado un objeto de tipo YAWLVetex que representa la tarea.

Para la integración de YAWL Editor en nuestra aplicación es ha tenido que modificar la clase principal, la clase YAWL Editor.
Se elimina el método `main` y se hace publico el `constructor` ya que en el código fuente se crea desde el `main` que está dentro del YAWLEditor y por esto es privado.

```
public YAWLEditor () {
    super ();
    updateLoadProgress (5);
    buildInterface ();
    SpecificationFileModel.getInstance ().subscribe (this);
    hideBottomOfSplitPane ();
       INSTANCE = this;
  }
```

En la operación de guardar, tras guardar el fichero yawl se debe guardar el modelo con los elementos, por esto se hace un método nuevo igual al método de guardar como pero que devuelva un valor booleano que indique si se ha guardado correctamente o no. En caso de que se guarde correctamente se procede a guardar el fichero xml con los elementos.
Igual que al guardar sucede al cargar, no se podrá cargar el modelo de elementos hasta que se cargue correctamente el workflow ya que se irán enlazando los elementos con las tareas del workflow conforme se vayan creando.

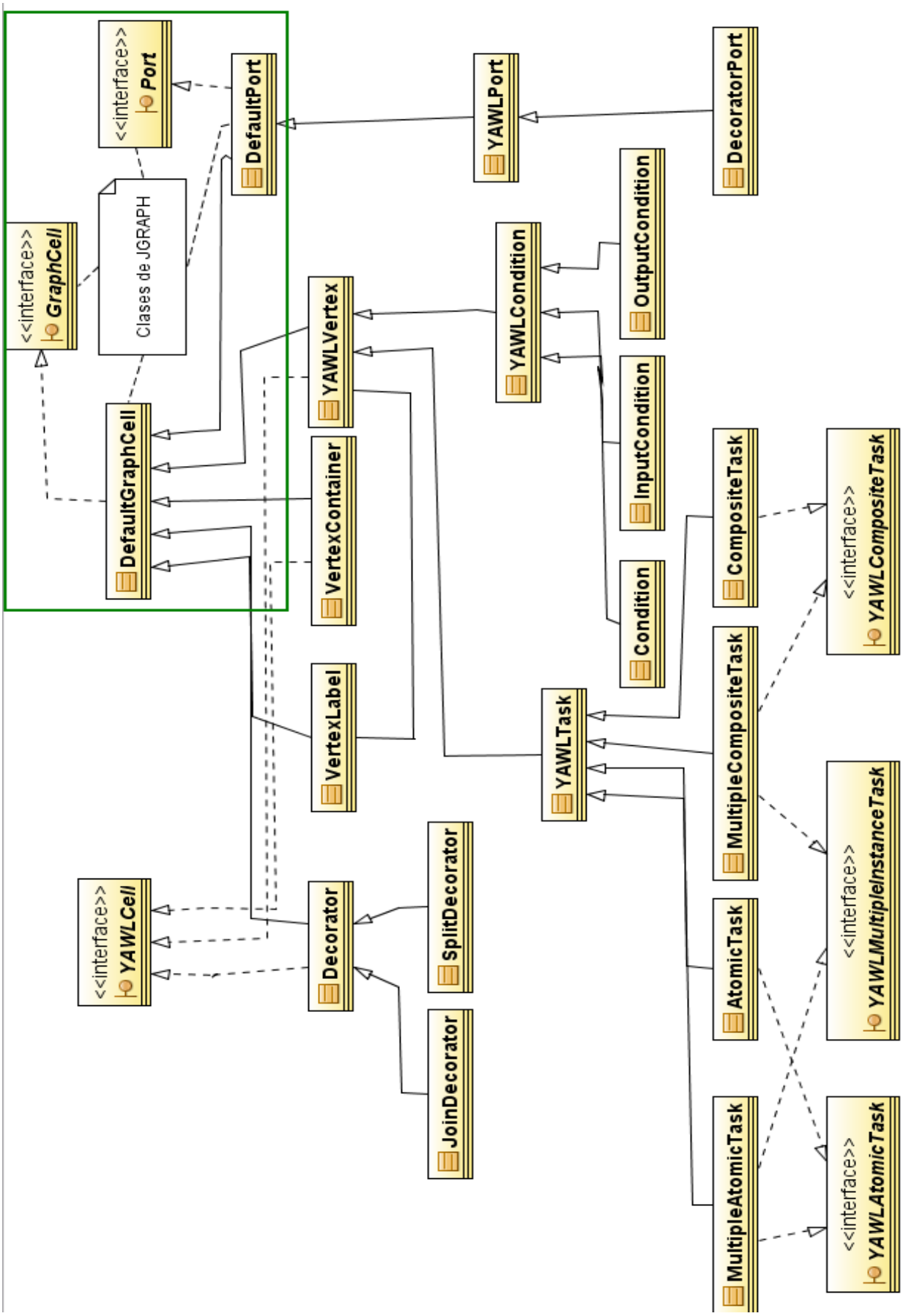

Figura 3.10: Diagrama de clases de los elementos de YAWLEditor.

3.3 DISEÑO DE LA APLICACIÓN

3.3.1 Gestión de ficheros XML

En este proyecto utilizamos ficheros XML para los elementos etiquetados en la GPC y que componen la representación estructurada de la GPC. Se podrán etiquetar elementos de los 3 estándares o terminologías utilizados: GEM II, terminología CH4 y HL7. Cada elemento que se etiquete estará relacionado con una parte textual de la GPC la cual deberá indicar el usuario. Esta parte de texto de la GPC será el contenido del elemento, por esto todos los elementos que se añadan serán elementos con contenido, menos los elementos RXO como comentaremos más adelante.

En el documento XML donde se guarde el marcado tendrá un elemento raíz que será el elemento "Modelo" y tendrá siempre los siguientes atributos que se añadirán por defecto:

•url: tendrá un valor de verdadero o falso según sea el origen de la GPC, de una url o de un fichero local.

- rutaFichero: para guardar la url o la ruta del fichero local de la GPC.
- FicheroYAWL: contiene ruta donde se encuentra el fichero para cargar en el editor YAWL correspondiente la GPC que se carga.

Los demás elementos que compondrán el fichero XML serán elementos GEM, HL7, CH4 o RXO. Cada elemento añadido de GEM, CH4 o HL7 tendrán unos atributos que se añadirán por defecto estos atributos son:

- tipo: indican el tipo de elemento dentro de cada estándar o terminología.
- inicio: indica la posición del carácter en el que comienza el texto etiquetado por este elemento.
- fin: indica la posición del carácter con el que finaliza el texto etiquetado por este elemento.

En algunos elementos es posible que se quiera guardar información adicional. Por esto será posible añadir atributos a cada uno de los elementos con la información que el usuario desee.

A continuación explicaremos como se componen y estructuran los elementos GEM, HL7, CH4 y RXO.

- **GEM II:** todos los elementos hijos directos del elemento raíz "Modelo" serán elementos pertenecientes a GEM II. Los elementos de GEM II que se podrán marcar en la GPC serán todos los pertenecientes a este modelo y comentados en el apartado 2 de la memoria. Un ejemplo de elemento GEM II representado en XML quedaría así:

```
<Gem tipo="Identity" id="7" inicio="0" fin="0">
    <Gem tipo="GuidelineTitle" id="8" inicio="500" fin="543">Allergic rhinitis and its
    impact on asthma.</Gem>
</Gem>
```

El contenido de los elementos GEM es texto y opcionalmente más elementos GEM, CH4 o HL7.

- **CH4:** los elementos CH4 añadidos están relacionados siempre con algún elemento GEM añadido anteriormente. En el fichero XML todos los elementos CH4 serán hijos directos de un elemento Gem. Este elemento Gem es el elemento al cuál esta relacionado el elemento CH4.Los elementos CH4 representados en el XML quedan así:

<CH4 tipo="Problema" id="31" inicio="33" fin="57" tareaYawl="Problema_3" nodoGem_asociado="Action">rhinitis, conjunctivitis</CH4>

- **HL7:** en HL7 se pueden añadir aquellos de tipo RXO. Al igual que con los elementos CH4, estos elementos deben estar relacionados con un elemento GEM II del cual es hijo directo. El elemento HL7 siempre tiene otro elemento llamado SegmentoRXO que representa los campos del segmento RXO. Los elementos HL7 en XML quedan representados así:

<HL7 tipo="RXO" id="33" inicio="0" fin="41" tareaYawl="RXO_6">

Carefully selected patients with rhinitis

<SegmentoRXO requestedGiveCode="codigo" requestedAmountMin="2" requestedAmountMax="" requestedGiveUnits="" requestedDosageForm="" providersPharmTreatInstruc="" providersAdministrationInstruc="" deliverToLocation="" allowSubstitutions="" requestedDispenseCode="" requestedDispenseAmount="" requestedDispenseUnits="" numbreOfRefills="" orderingProviderDEANumb="" pharmTreatSuppliID="123" needsHumanReview="si" requestedGivePer="" requestedGiveStrenght="" requestedGiveStrenghtUnits="" indication="" requestedGiveRateAmount="" requestedGiveRateUnits="" totalDailyDose="" supplementaryCode="" requestedDrugStrengthVolume="" requestedDrugStrengthVolumeUnits="" pharmacyOrderType="" dipensingInterval=""/>

</HL7>

Aparte del uso del fichero XML en la aplicación se utilizan ficheros XML Schema para cargar la estructura de las terminologías y estándares y otro para comprobar que un modelo cargado está bien formado.

GEM II ya tienen implementado su fichero XML Schema donde indica la estructura jerárquica de sus ficheros. Este fichero es utilizado para cargar la estructura del GEM y mostrar en un árbol todos los elementos de GEM. Así, el usuario puede tener visualizada esta estructura y facilitar el proceso de etiquetar la GPC.

Para HL7 y CH4 se han creado los ficheros esquemas con la estructura de sus componentes. Estos

ficheros se han creado haciendo uso de la etiqueta `element` y se ha utilizado el namespace xs. Para cada elemento se utiliza la etiqueta `xs:element` y `xs:documentation` para la descripcion del elemento. Esta es la misma estructura que se sigue en el fichero esquema de GEM II. Un ejemplo de un elemento de CH4 es el siguiente:

```
<xs:element name="Problema">
    <xs:annotation>
        <xs:documentation>Descripcion del problema</xs:documentation>
    </xs:annotation>
</xs:element>
```

A la hora de cargar un modelo para seguir trabajando con él se debe cargar el fichero XML que guarda los elementos etiquetados. Para comprobar que el contenido de este fichero se correcto, tanto el formato XML como la jerarquía, se utilizará un esquema XML para parsearlo y verificar que es correcto antes de cargarlo.

3.3.2 Diseño del modelo

La aplicación se ha construido siguiendo el patrón Modelo-Vista-Controlador.
EL patrón MVC fue introducido por primera vez en 1979 por un desarrollador de Smalltalk. Este patrón ayuda a desacoplar el acceso a datos y la lógica de negocio, para esto se distinguen tres elementos en el desarrollo:

- Modelo : El modelo representa los datos y las reglas que gobiernan el acceso y actualización de los mismos.
- Vista : La vista muestra el contenido de un modelo. Especifica exactamente cómo los datos del modelo deben ser presentados. Si los datos del modelo cambian, la vista debe actualizar su representación según corresponda.
- Controlador: El controlador traduce las interacciones del usuario con la vista en acciones que el modelo puede realizar.

El modelo no contiene una referencia a la vista, pero utiliza un modelo de eventos para realizar la notificación de los cambios a las partes interesadas. Una de las consecuencias de este poderoso diseño es que múltiples vistas pueden compartir un modelo. Cuando ocurre un cambio en el modelo de datos, cada vista es notificada mediante un evento de cambio de propiedad y puede actualizarse según corresponda.

En este proyecto se ha utilizado una versión modificada del MVC, se representa en la Figura 3.11.

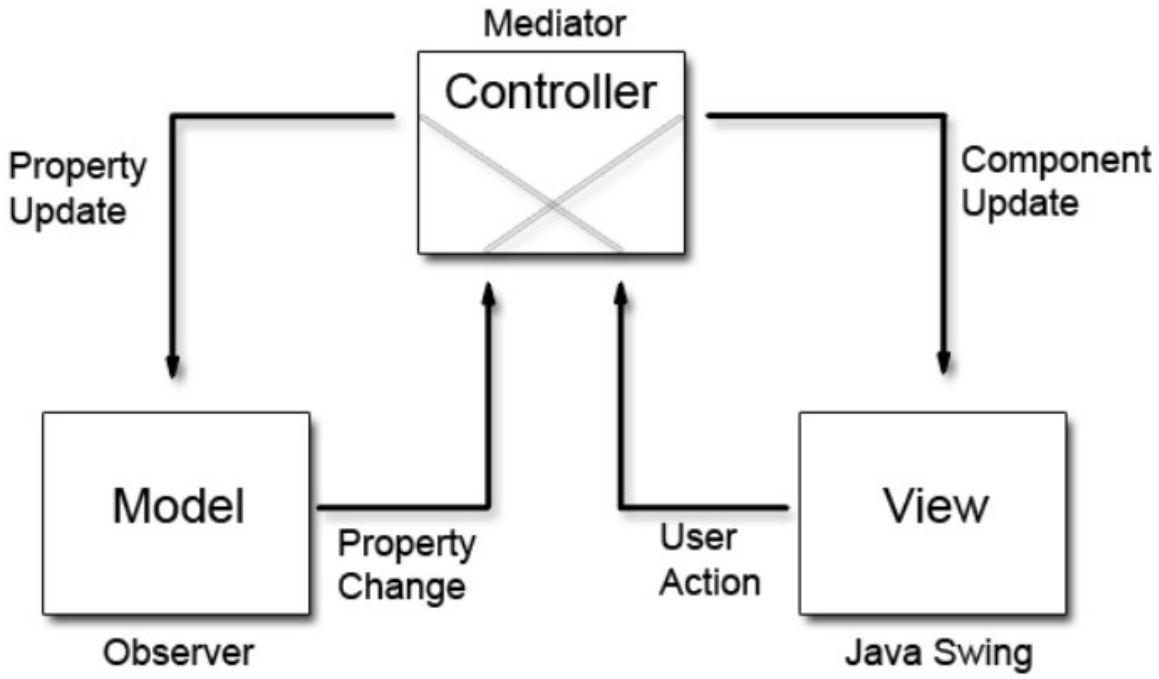

Figura 3.11: MVC en su versión modificada. [21]

En esta versión las notificaciones de los cambios de estado en los modelos son comunicadas a la vista a través del controlador. Por tanto, el controlador es un mediador del flujo de datos entre el modelo y la vista en ambas direcciones. Las vistas siguen usando el controlador para traducir las acciones del usuario en actualizaciones de propiedades en el modelo. Además, cambios en el estado del modelo son comunicados a los objetos vista mediante los objetos controlador de la aplicación.

Por lo tanto, tras instanciar los tres componentes, la vista y el modelo se registran en el controlador. Una vez que el usuario interactúa con la vista los eventos son los siguientes:

1. La vista reconoce que ocurrió una acción de la GUI --por ejemplo, pulsar un botón o arrastrar una barra de desplazamiento-- utilizando un método listener que es registrado para ser llamado cuando ese tipo de acciones ocurran.
2. La vista invoca al método apropiado en el controlador.
3. El controlador accede al modelo, posiblemente actualizándolo de una forma adecuada a la acción del usuario.
4. Si el modelo ha sido alterado, notifica del cambio a los listeners interesados. Sin embargo, en este caso, la notificación del cambio es enviada al controlador

Esta versión modificada de MVC, frente al modelo estándar, ayuda a desacoplar más el modelo de la vista. En este caso, el controlador puede dictaminar las propiedades del modelo que espera encontrar en uno o más modelos registrados frente al controlador. Además, también puede proveer los métodos que efectúan los cambios de propiedades en el modelo para una o más vistas que se hayan registrado.

Esta versión modificada ha sido la utilizada para desarrollar la aplicación. En el siguiente apartado se detalla el diagrama de clases donde se indica a qué componente (Modelo, Vista o Controlador) pertenece cada clase.

3.3.3 Diagrama de clases

A continuación se detallará el diagrama de clases que compone la aplicación.
Por un lado tenemos las clases que modelan cada una de las terminologías y estándares utilizados (GEM, CH4 y HL7). Para modelar esto hemos usado el patrón Singleton.
El patrón Singleton "garantiza que una clase solo tenga una instancia, y proporciona un punto de acceso global a ella" [8]. De esta manera tenemos construida de manera global una jerarquía que modela los elementos de los estándares y terminologías y así poder acceder a ellos para cualquier consulta sobre un elemento específico. La clase que modela cada jerarquía será `Taxonomia` y tendrá tantas subclases como estándares o terminologías se utilicen. El diagrama de clases de la parte que modela esto se muestra en la Figura 3.12.

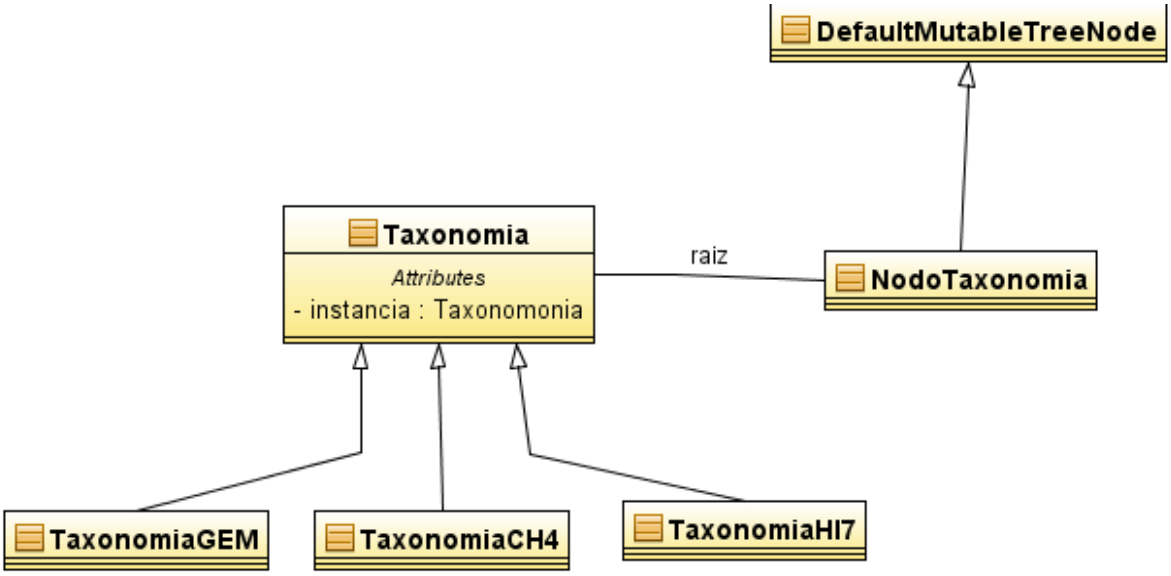

Figura 3.12: Diagrama de clases. Estándares y terminologías.

La clase `NodoTaxonomia` representará la raíz de la taxonomía y hereda de `DefaultMutableTreeNode` ya que las taxonomías se representarán en un `JTree`. Al ser un `DefaultMutableTreeNode` tendrá un conjunto de hijos, estos hijo serán de nuevo de tipo `NodoTaxonomia`. Para cada árbol de la taxonomía el elemento `root` será el raíz de la clase `Taxonomia`.

Para diferenciar las clases que pertenecerán a las vistas y las que pertenecerán al modelo se han creado dos interfaces `InterfazVista` e `InterfazModelo` que implementarán cada una de las clases según correspondan a partes de la vista o del modelo. La interfaz `InterfazModelo` tendrá los métodos necesarios para añadir listener al modelo, eliminarlos y notificarlos cuando el modelo cambie.

```
public void addPropertyChangeListener (PropertyChangeListener l);
public void removePropertyChangeListener (PropertyChangeListener
l);
public void firePropertyChange (String propertyName, Object
oldValue, Object newValue);
```

La vista tendrá el método necesario para recibir notificaciones de que debe ser cambiada:

```
public void modelPropertyChange (PropertyChangeEvent evt);
```

Con este método se recupera el evento con `evt.getPropertyName ()` y dependiendo de los valores se ejecutará una acción u otra.
Respecto a los controladores existirá una clase genérica `Controlador` y cuatro subclases específicas de `Controlador`, una por cada una de las pestañas que se van a implementar en la aplicación.
La aplicación tendrá cuatro pestañas:

- 1ª: pestaña "Etiquetar Gem", para hacer el primer etiquetado a la GPC con elementos del estándar GEM.
- 2ª: pestaña "Etiquetar CH4" para hacer el etiquetado de elementos de la terminología CH4 sobre elementos ya etiquetados de GEM.
- 3ª pestaña: "Etiquetar HL7" para hacer el etiquetado de elementos RXO y estructurar los mensajes RXO sobre elementos ya etiquetados de GEM.
- 4ª pestaña: "YAWL" para construir el workflow.

El controlador de cada pestaña gestionará eventos que ocurran en cada una de ellas siendo el intermediario entre las vista y los modelo contenidos en cada pestaña. El controlador guardará la referencia a las vistas y a los modelos y será el intermediario entre ellos. La parte del diagrama de clases que representa los controladores se muestra en la Figura 3.13.

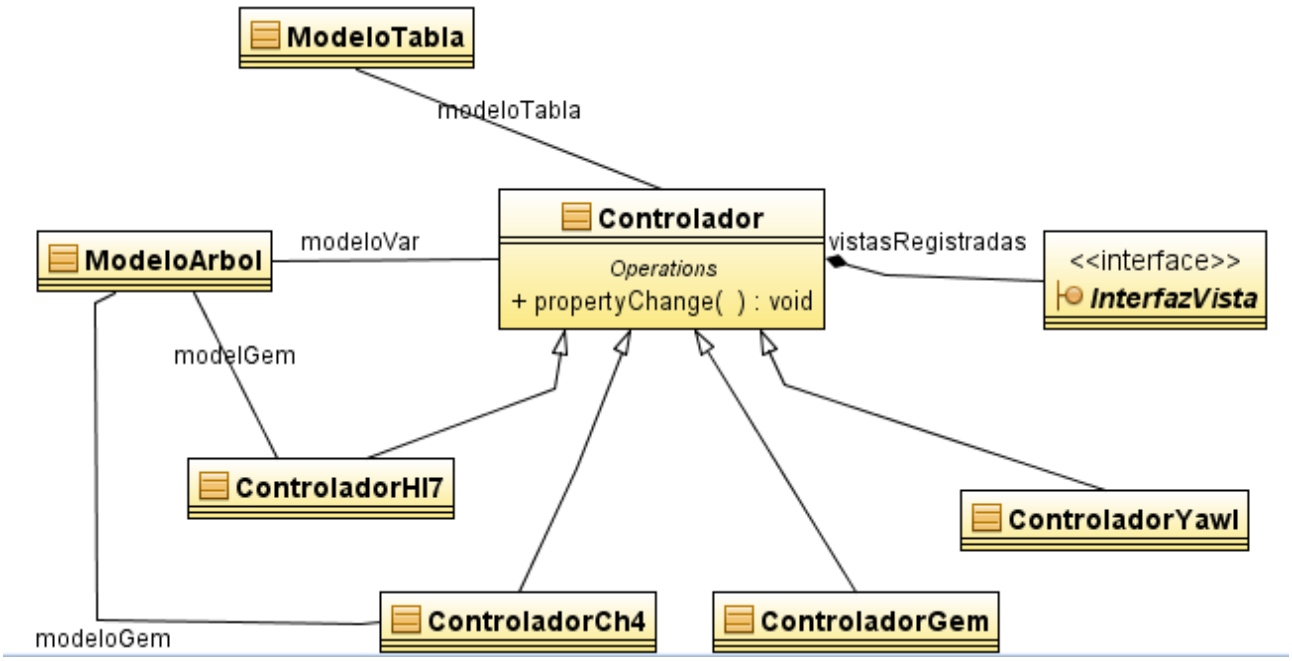

Figura 3.13: Diagrama de clases. Controladores.

En la Figura 3.13, el atributo `modeloTabla` representa el modelo del `JTable` que se utilizará para representar los atributos del elemento seleccionado en el `JTree`. Este `JTree` representará los elementos añadidos de cada estándar o terminología, dependiendo de la pestaña a la que se refiera.

El atributo `modeloVar` representa el modelo del `JTree` de los elementos añadidos de cada pestaña. El `controlaodorCh4` y `controladorHl7` a su vez tendrá el modeloGem que representará los elementos Gem añadidos para enlazarlos con elementos de su taxonomía.

Para representar los modelos de los `JTree` que representarán los elementos añadidos de cada estándar o terminología se ha creado una clase `ModeloArbol` que extiende de `DefaultTreeModel`. A su vez tenemos cuatro clases que heredan de esta para representar los modelos específicos de cada árbol. Tres para cada una de las taxonomías y una para representar el modelo global de etiquetas añadidas, contendrá las etiquetas añadidas de todas las taxonomías representando así la GPC estructurada.

La figura 3.14. representa la parte del diagrama de clases del modelo del árbol. Cada `ModeloArbol` tendrá un nodo raíz de tipo `NodoArbolGem`, y esta a su vez un atributo `tareaYAWL` que representará las tareas del workflow de cada nodo.

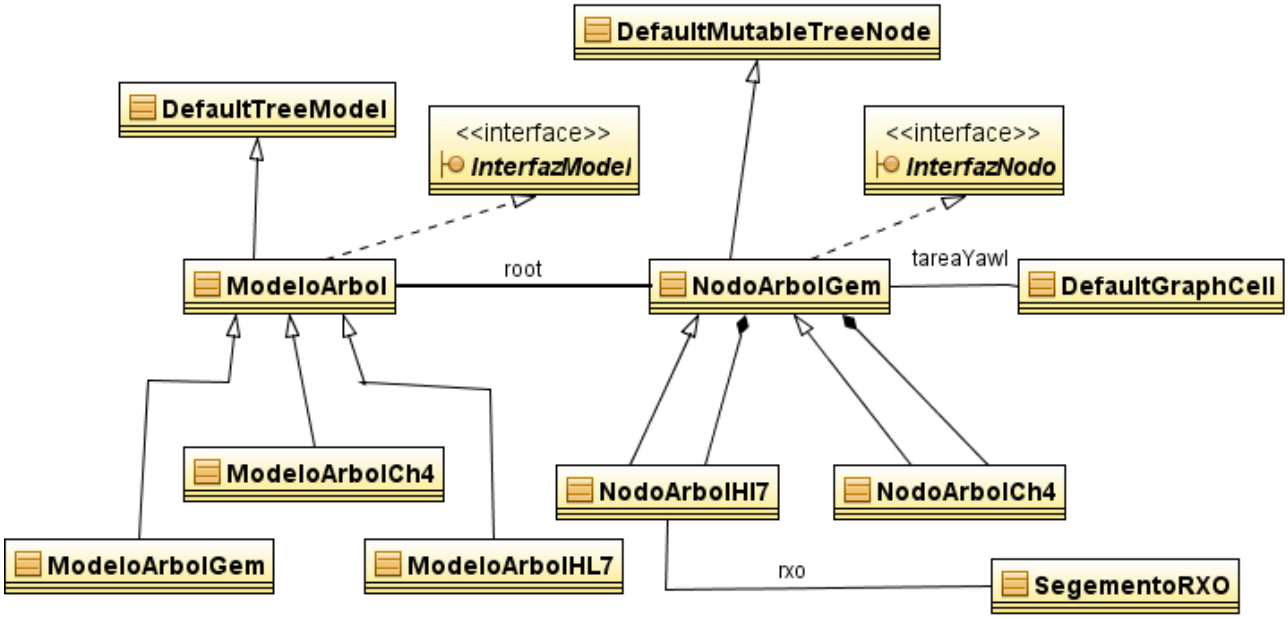

Figura 3.14: Diagrama de clases. Modelo del JTree.

Las clases `GuiaHTML` y `FicheroXML` se utilizan para representar la GPC cargada y el `ficheroXML` utilizado para guardar el modelo de elementos. Estas clases serán representadas según el patrón Singleton explicado anteriormente.

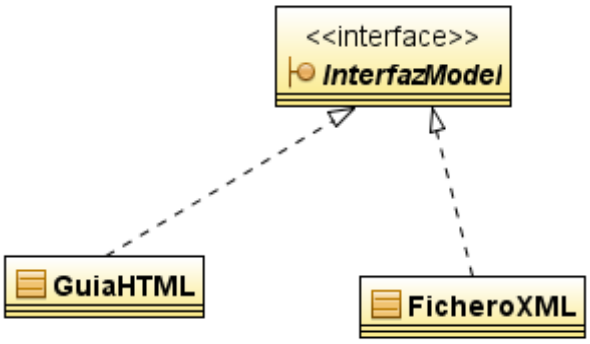

Figura 3.14: Diagrama de clases. Modelo FicheroXML y GuiaHTML.

La parte de la vista se representa en la Figura 3.15. Tenemos una clase genérica Panel para representa las tres pestañas para etiquetar los elementos y otra clase `PanelYawlEditor` que para integrar el YAWL Editor. Todos los paneles tendrán un atributo `panelGuia` que representará la GPC. También tendrán un árbol que representa los elementos elementos añadidos a la representación estructurada hasta el momento. En el caso del Panel será el atributo `arbolVar` y en el caso del `PanelYawlEditor` será `arbolCompleto`. Panel a su vez tendrá un atributo `tablaAtributos` que representa los atributos del elemento seleccionando en el panel y un `arbolFijo` que representa los elementos de cada taxonomía.

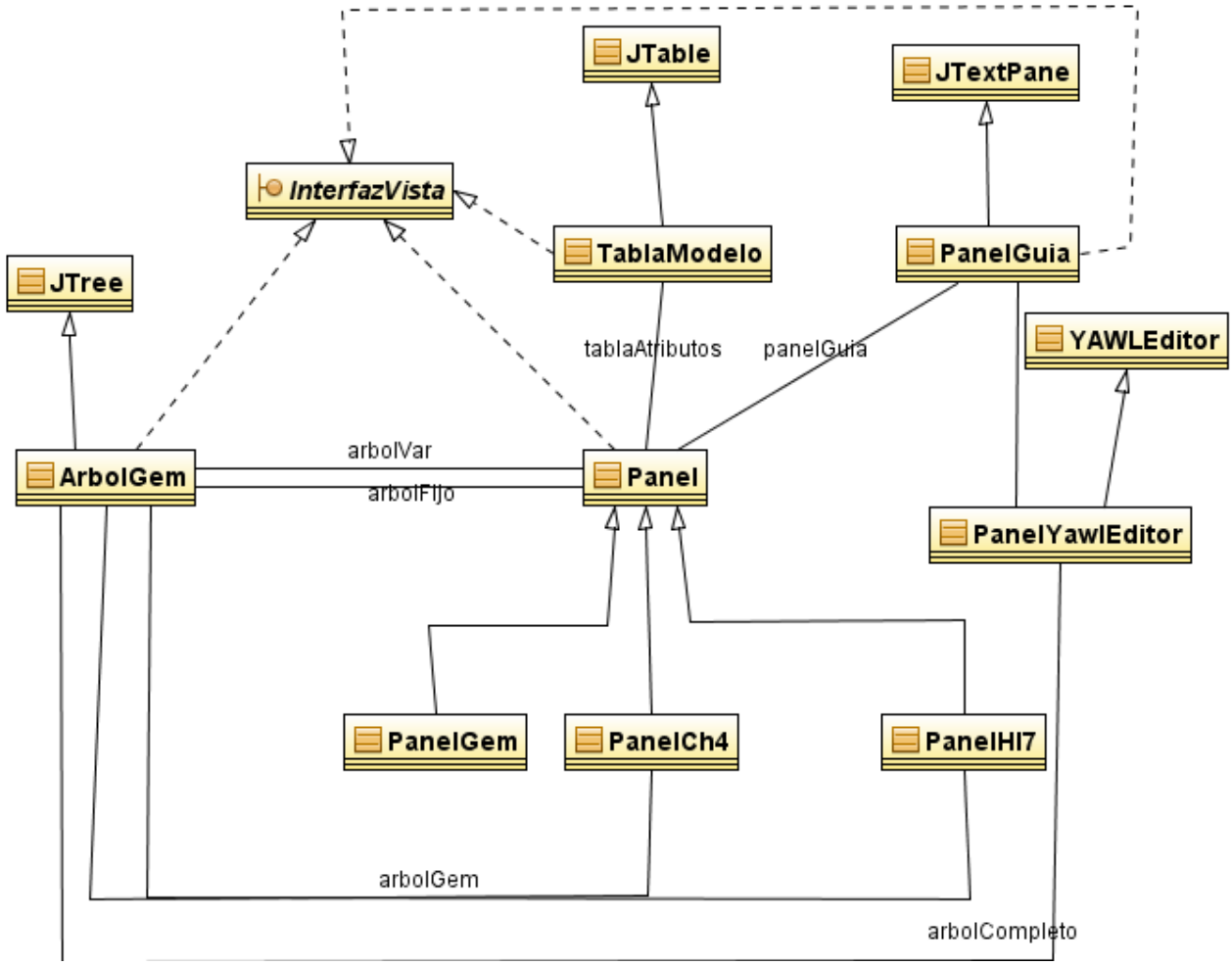

Figura 3.15: Diagrama de clases. Vista.

3.3.4. Diseño de la interfaz de usuario

En este apartado se mostrarán imágenes de las distintas vistas de la interfaz gráfica de la herramienta desarrollada y la funcionalidad que proporciona cada una de ellas.
Como ya se ha comentado, el proceso de construcción del modelo formal de la GPC debe ser un proceso por pasos donde se va construyendo una representación estructurada de la GPC hasta

construir la representación formal. Por esto la aplicación se ha diseñado de manera que tenga una ventana principal con cuatro pestañas o módulos donde cada uno proporciona la funcionalidad de cada uno de estos pasos. Las funcionalidades que proporciona cada módulo son:
1.Etiquetado de elementos GEM II.
2.Etiquetado de elementos CH4.
3.Etiquetado de elementos HL7.
4.Construcción del workflow final.

Con los tres primeros módulos se obtiene la representación estructurada de la GPC y en el módulo final se construye el workflow relacionado con la representación estructurada. Como se verá en todos los módulos aparecerá el texto original de la GPC para poder guardar en todo momento la trazabilidad entre los elementos de las representaciones y el texto de la GPC.
A continuación veremos con más detalle el diseño de cada uno de estos módulos.

3.3.4.1. Diseño del Módulo de etiquetado GEM II

Este módulo proporciona la funcionalidad para el primer paso que se debe llevar a cabo en la construcción del modelo formal, etiquetar el texto de la GPC con los elementos del modelo GEM II. La interfaz gráfica del módulo se muestra en la Figura 3.16.

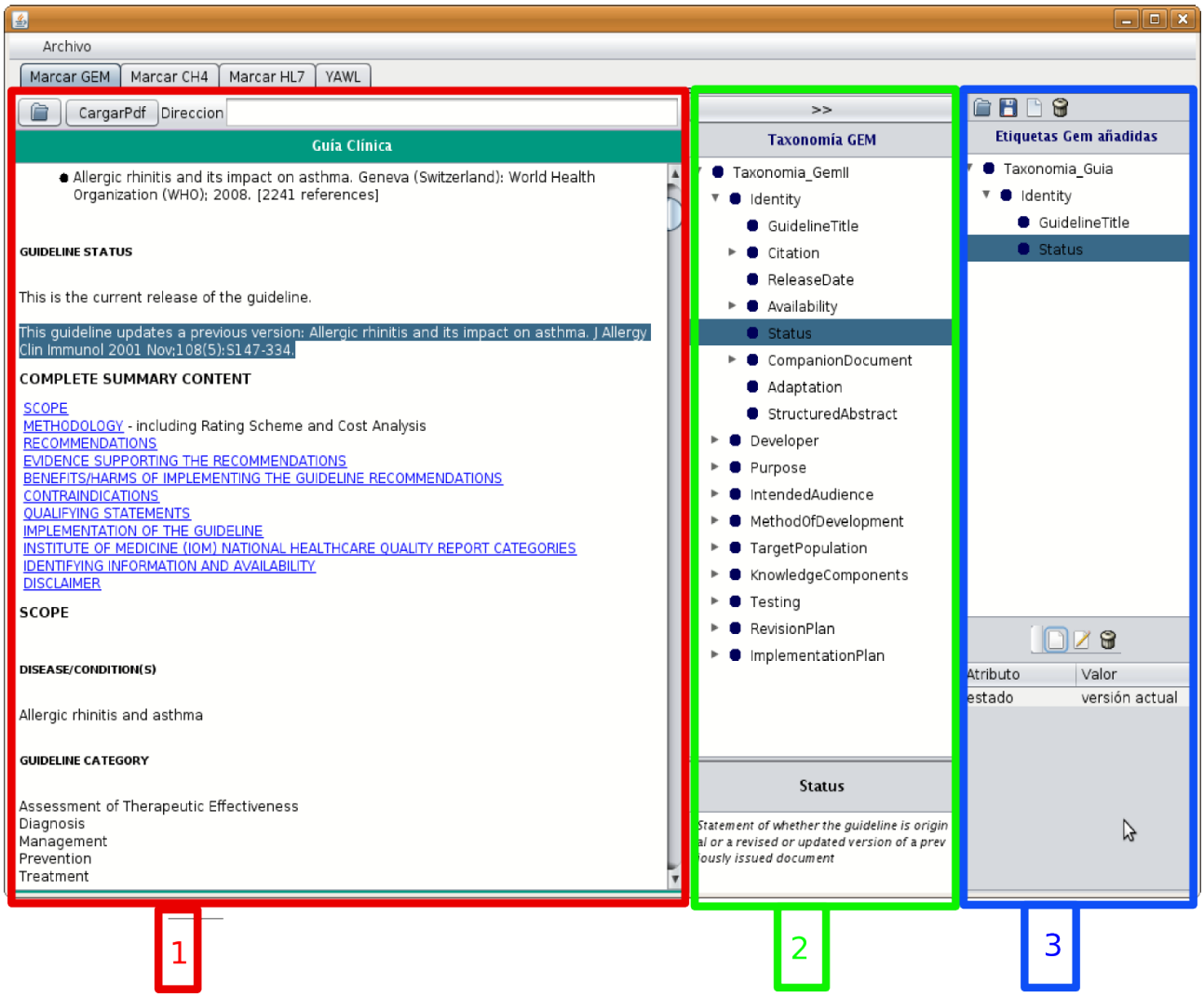

Figura 3.16: Módulo de etiquetado de GEM II.

Como se muestra la Figura 3.16 este módulo se puede dividir en tres partes principales que han sido recuadradas y numeradas en dicha figura. Explicaremos cada una de estas partes:

1. En esta parte se visualiza el texto de la GPC y en la parte superior se encuentra la barra de herramientas que permite cargar la GPC en HTML almacenada localmente o introducir la url de la GPC.

2. Esta parte representa de forma jerárquica, utilizando un árbol, los elementos del modelo GEM II. En la parte inferior se muestra la descripción del elemento seleccionado en cada momento. En la parte superior se localiza el botón que se utilizará para añadir elementos GEM II que etiqueten parte del GPC a la representación estructurada.

3. En esta parte se visualiza, jerárquicamente en un árbol, la información de los elementos GEM II que han sido añadidos a la representación estructurada y que así etiquetan partes de la GPC. En la parte inferior se muestra la tabla con los atributos del elementos seleccionados y una barra de herramientas para poder añadir, editar y eliminar atributos del elemento seleccionado en el árbol.

En la parte superior de esta parte se visualiza una barra de herramientas que proporciona los métodos necesarios para eliminar y modificar elementos GEM II añadidos. Además proporciona

dos botones, uno para cargar un modelo formal y otro para crear un nuevo modelo. Se ha decidido poner estos botones aquí ya que es la primera pantalla que se muestra al abrir la aplicación, pero esta funcionalidad también está añadida en el menú principal.

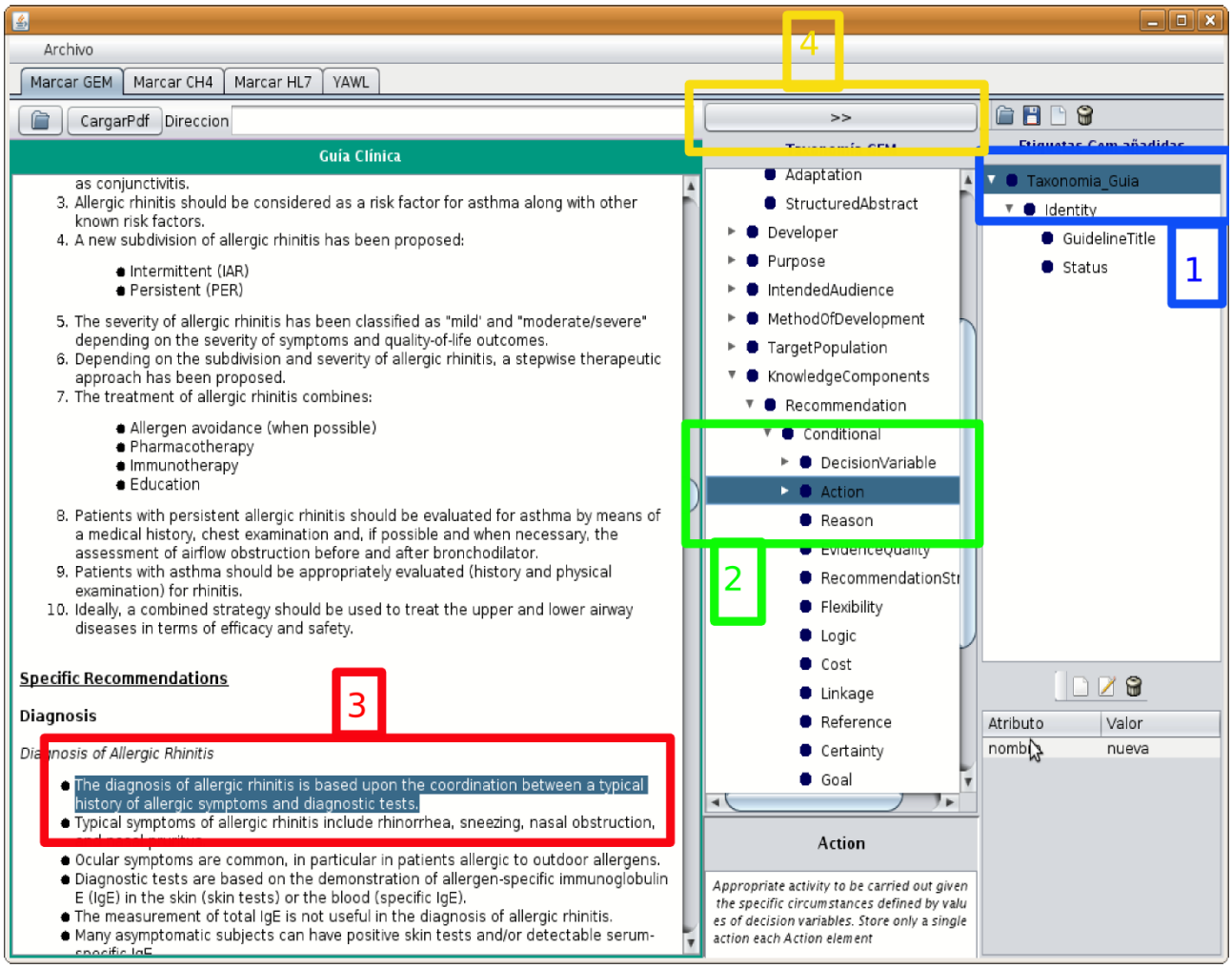

Figura 3.17: proceso para etiquetar un elemento GEM II.

En la Figura 3.17 se muestra el proceso para etiquetar un elemento de GEM II. Los pasos, enumerados y recuadrados, son los siguientes:

1. Se selecciona el nodo padre en los elementos GEM II al cual quiere añadirse el elemento GEM II.

2. En la taxonomía que representa los elementos GEM II se selecciona el tipo de nodo que se quiere añadir.

3. En la GPC se selecciona el texto que se quiere etiquetar con el elemento GEM II

4. Finalmente se pulsa el botón para añadir el elemento y añade el el árbol de elementos GEM II añadidos.

3.3.4.2. Diseño del Módulo de etiquetado CH4

Este módulo proporciona la funcionalidad de etiquetar y así añadir a la representación estructurada elementos que pertenezcan a la terminología CH4. Estos elementos se añadirán siempre relacionándolos con partes del texto etiquetadas por un elemento GEM II anteriormente. Por esto, los elementos CH4 añadidos irán relacionados con un elemento GEM II añadido en el primer paso.

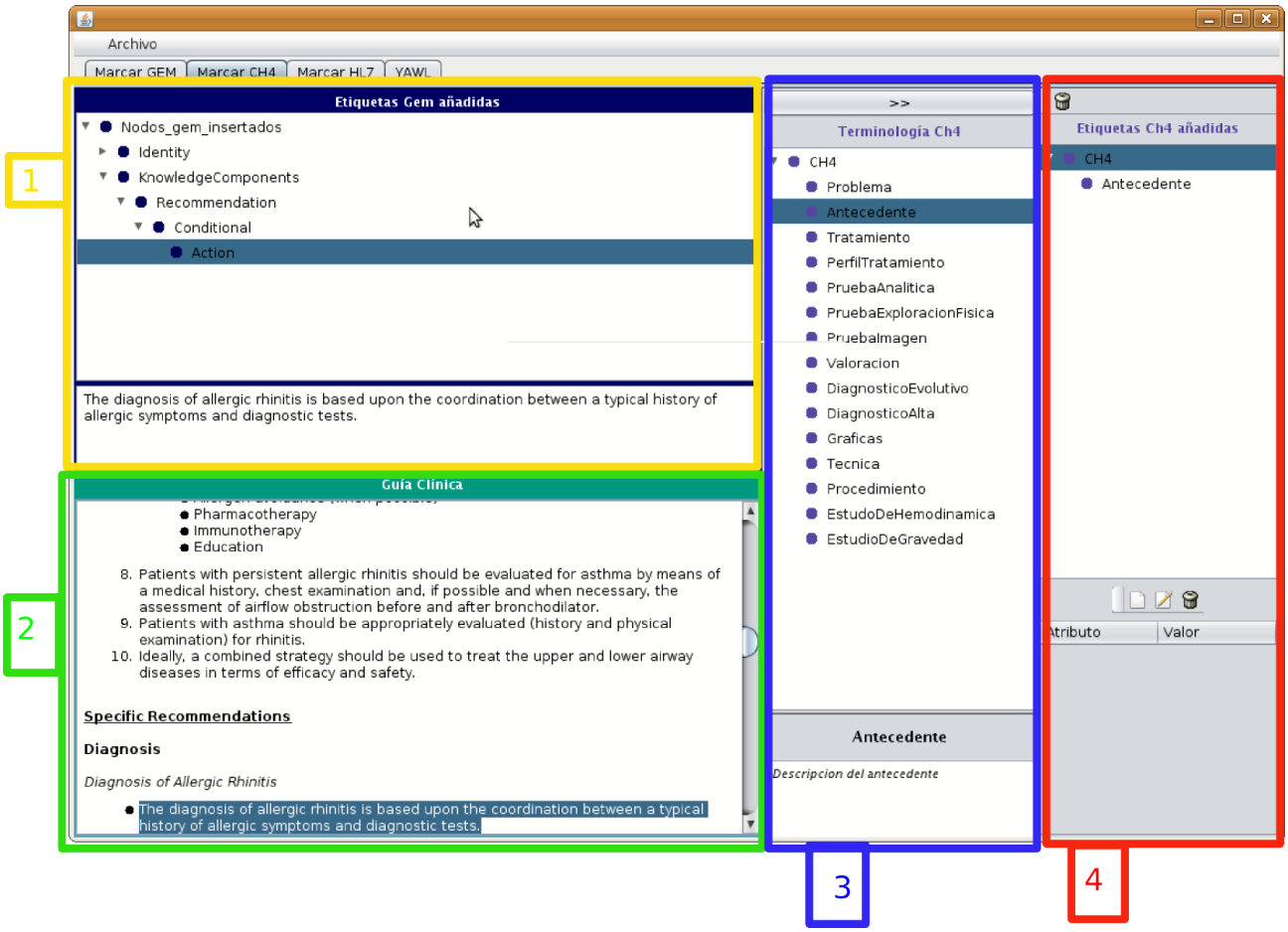

Figura 3.18: Módulo de etiquetado de CH4.

En la Figura 3.18 se muestra la interfaz del módulo de etiquetado de CH4. La interfaz es parecida a la del módulo de etiquetado de GEM II. Las partes en las que se divide son:

1. Representa los elementos GEM II añadidos hasta el momento.

2. Representa el texto de la GPC.

3. Representa de forma jerárquica en un JTree los elementos de la Terminología CH4 y que podrán ser etiquetados y añadidos a la representación estructurada. En la parte de abajo se mostrara el texto con la información del elemento seleccionado. En la parte de arriba se encuentra el botón para que lanzará el evento para insertar el elemento.

4. Representa los elementos CH4 que se han añadidos a la representación estructurada en cada momento. En la parte de abajo se muestra una tabla con los atributos del elemento seleccionado, esto es igual que en el módulo de GEM II. En la parte superior se muestra una

barra de herramientas con funcionalidad para eliminar elementos CH4 añadidos y para poder cambiar el texto que etiqueten estos.

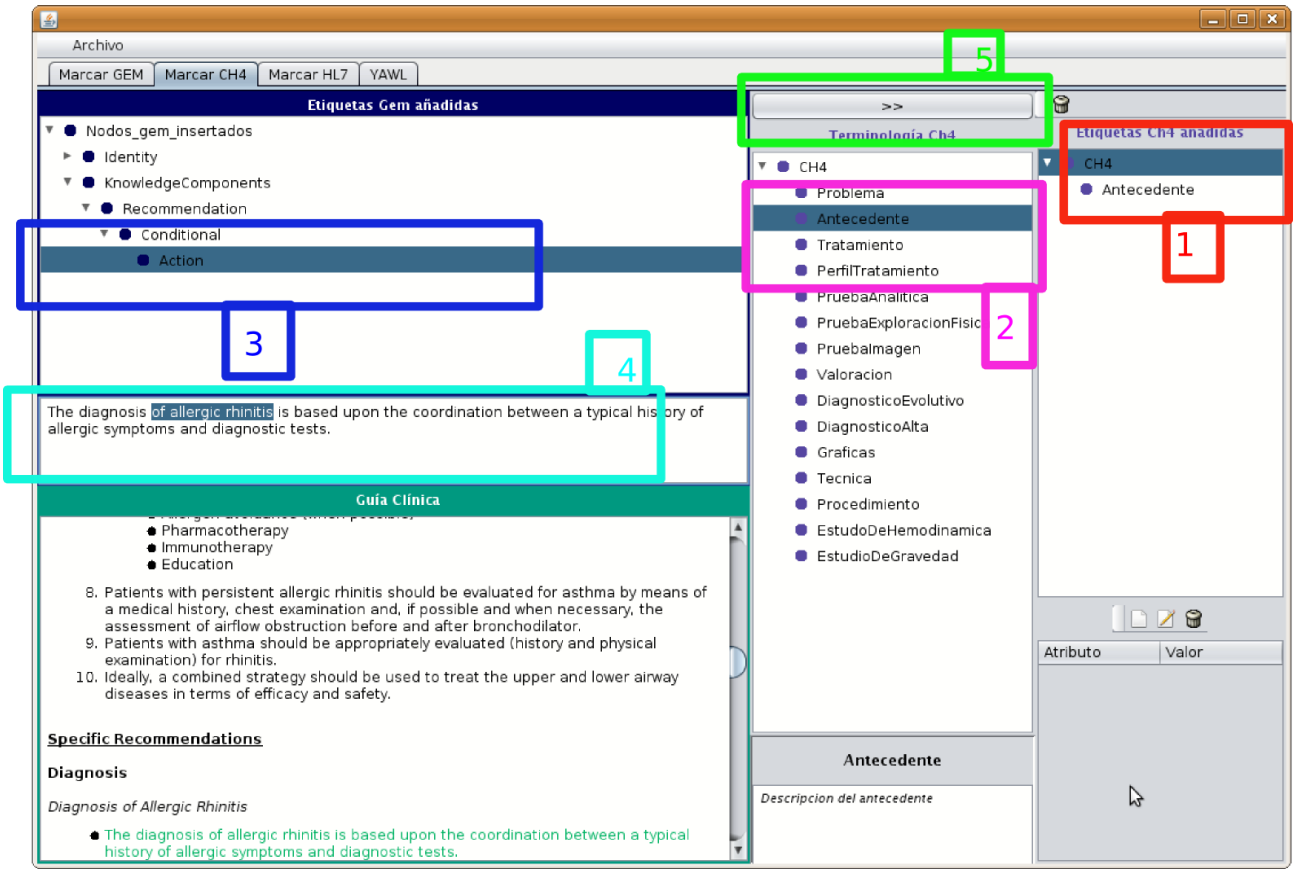

Figura 3.19: Pasos a seguir para insertar un elemento CH4.

En la Figura 3.19 se muestran los pasos a seguir para insertar un elemento CH4 que pase a formar parte de la representación estructurada. Los pasos son los siguientes:

1. Se selecciona el elemento padre de los elementos CH4 añadidos.
2. Se selecciona el tipo de elemento de la terminología CH4 que se quiere insertar.
3. Se selecciona el elemento GEM II con el que se quiere relacionar el elemento CH4 a añadir.
4. Se selecciona la parte de texto etiquetada por el elemento GEM II que se etiquetará con el elemento CH4 a añadir.
5. Se pulsa el botón para añadir el elemento.

3.3.4.3. Diseño del módulo de etiquetado HL7

Este módulo proporciona la funcionalidad de etiquetar con el fin de añadir a la representación estructurada elementos que pertenezcan al estándar HL7. Estos elementos se añadirán siempre relacionándolos con partes del texto etiquetadas por un elemento GEM II anteriormente, igual que pasaba con los elementos CH4.

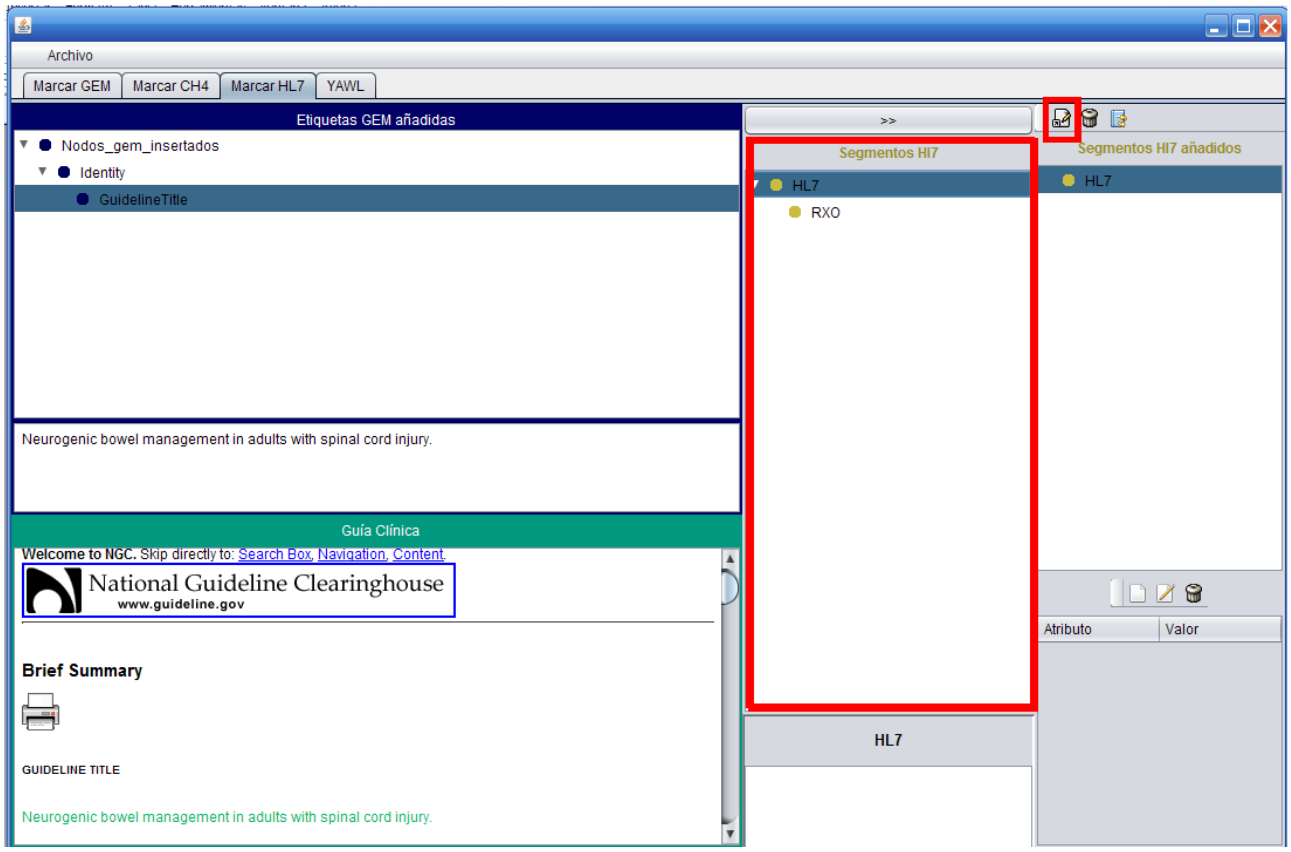

Figura 3.20: Módulo de etiquetado de HL7.

La Figura 3.20 muestra la interfaz principal. Como se puede ver la interfaz es muy similar a la de CH4. Las diferencias son que esta muestra la funcionalidad para etiquetar elementos HL7 en vez de elementos CH4. Por esto en la parte central se muestran los elementos HL7 que se podrán añadir, que en nuestro caso sólo serán elementos RXO. En la parte de la derecha se mostrarán los elementos HL7 añadidos y en la barra de herramientas se añade un botón para editar los campos del segmento que se crean con el elemento añadido.

El proceso para añadir un elemento HL7 es el mismo que para añadir un elemento CH4, solo que en el paso 5 se abrirá un formulario con todos los campos del segmento RXO para que el usuario pueda rellenarlos y así formar el segmento. El formulario es el que se muestra en la Figura 3.21.

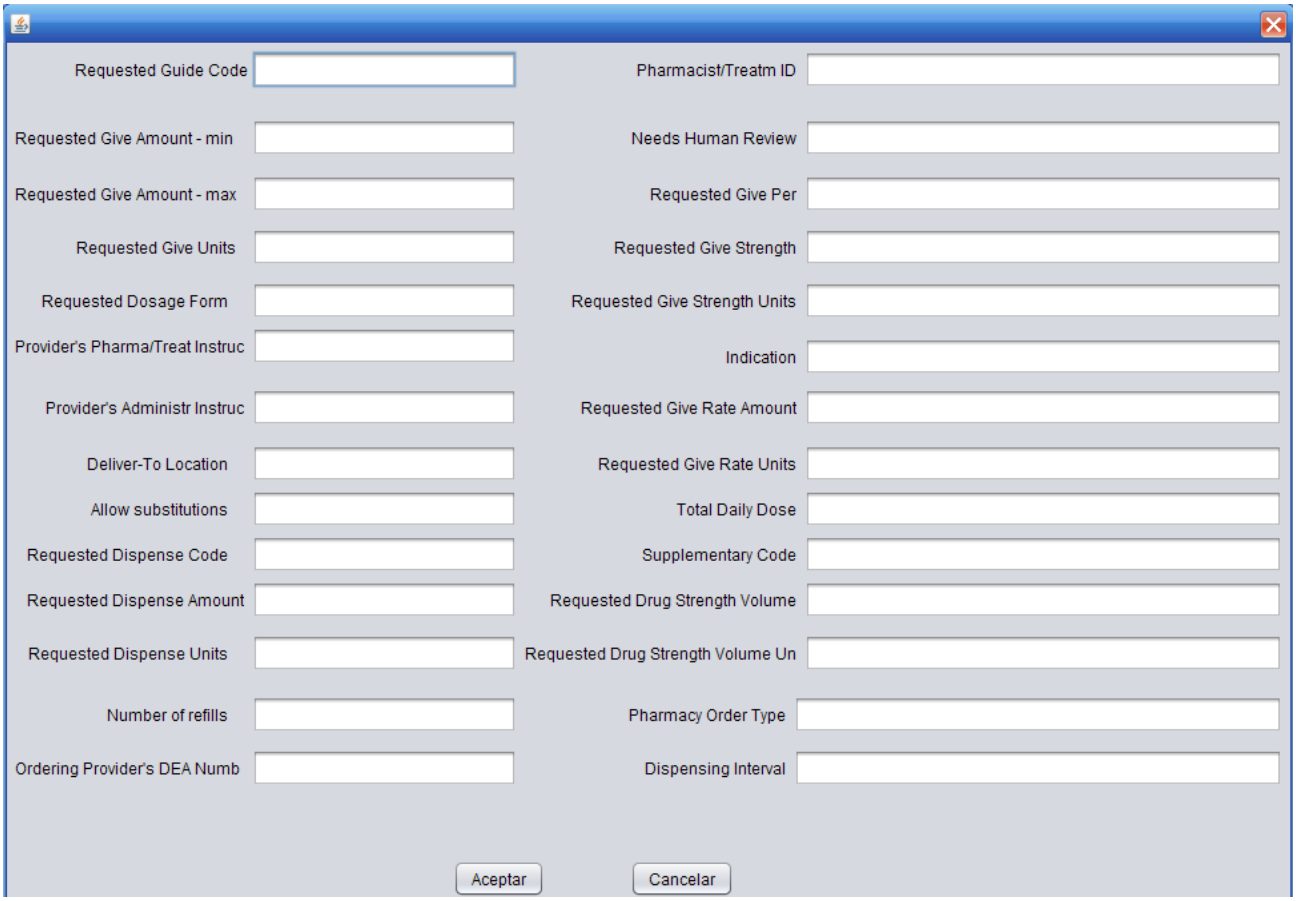

Figura 3. 21: Formulario con los campos del segmento RXO.

3.3.4.4. Diseño del Módulo YAWL.

Una vez construida la representación estructurada de la GPC etiquetando esta con elemento GEM II, CH4 y HL7 el último paso que queda es construir el workflow que representará el modelo formal de esta. Esto se consigue con la funcionalidad que proporciona el último módulo. La Figura 3.22 representa la vista de este módulo.

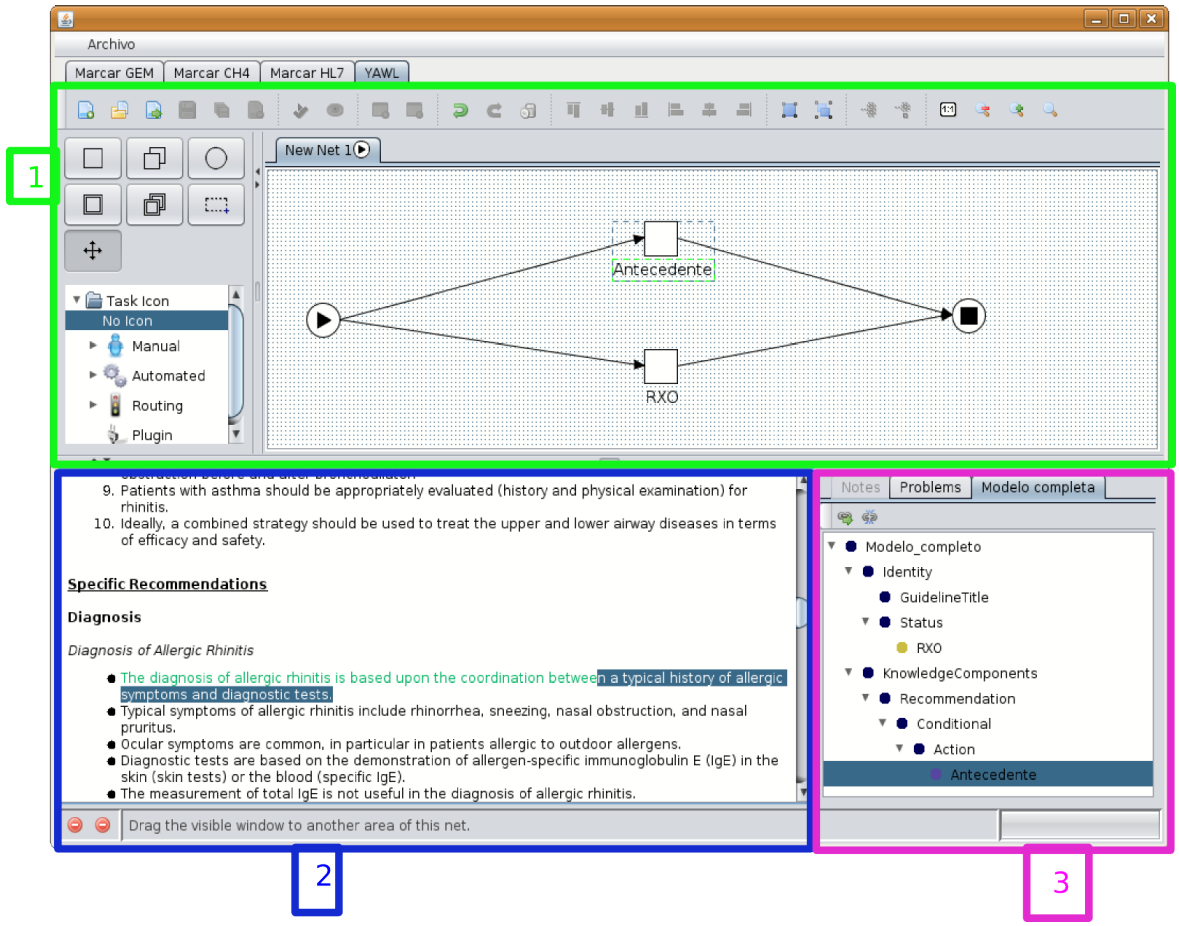

Figura 3.22: Módulo para el representación formal de la GPC.

Como se muestra en la Figura 3.22 este módulo se divide en 3 partes principales:

1. Esta parte corresponde con la herramienta YAWL Editor. YAWL Editor ha sido integrada en esta aplicación para poder construir el workflow.
2. Esta parte se encuentra el texto de la GPC.
3. Esta parte muestra en forma de árbol los elementos que forman la representación estructurada construida en los módulos anteriores. Se mostrará de forma jerárquica los elementos GEM II, CH4 y HL7 añadidos. Los elementos CH4 y HL7 serán hijos de los elementos GEM II con los que están relacionados.
   En esta parte también existe una barra de herramientas que proporciona funcionalidad para enlazar los elementos de esta jerarquía con tareas del workflow que se construye.

Se mantiene la trazabilidad entre los elementos, las tareas y la GPC. Cuando se selecciona un elemento de la representación estructurada, se seleccionará en workflow la tarea que tiene relacionada y, a la vez, el texto de la GPC que representa. Lo mismo ocurre cuando se selecciona una tarea del workflow.

4 CONCLUSIONES Y TRABAJO FUTURO

El objetivo principal de este proyecto es el desarrollo de una herramienta que, haciendo uso de estándares y terminologías clínicas, permita al usuario construir una formalización de una GPC mediante workflow. En esta formalización se representan las acciones de la GPC en el formato apropiado para facilitar el seguimiento de la realización de tareas de la historia clínica de un paciente.

La herramienta desarrollada es un asistente que, mediante una serie de pasos y haciendo uso de los estándares GEM II y HL7 y la terminología CH4, facilita el procedimiento para conseguir la formalización de la GPC mediante un lenguaje de workflow. Las acciones de la GPC son representadas como un flujo de tareas que mantienen en todo momento una relación con los elementos de los estándares y terminologías utilizados y a su vez con el texto original de la GPC en formato HTML.

El proceso para construir el workflow de la GPC partiendo del HTML se divide en dos procesos principales. En el primer proceso se obtiene una representación estructurada del documento a partir del marcado del texto de la GPC con elementos de GEM II en primer lugar y elementos CH4 y HL7 posteriormente. En el segundo proceso se construye la formalización utilizando un lenguaje workflow y enlazando las tareas de este con los distintos elementos de la representación estructurada obtenida en el primer proceso. Así se guardará la relación entre las tareas del workflow, los elementos de la representación estructurada y la GPC original.

Para resolver el problema planteado en este proyecto existen algunas herramientas que han sido desarrolladas, mayoritariamente para facilitar la construcción de un formalismo de una GPC. Algunas de estas herramientas han sido comentadas en el apartado 1.4. Estas herramientas se pueden dividir en dos categorías, las que siguen un enfoque centrado en el modelo o un enfoque centrado en el documento [3].
Algunos ejemplos de herramientas que siguen un enfoque centrado en el modelo son AsbruView [4], AsbruFlow [3] o Arezzo [3]. Este tipo de herramientas utilizan algún formalismo desarrollado para implementar GPC como puede ser Asbru[5], Prodigy[13] o Proforma [6], pero no guardan relación directa con el texto original de la GPC. Por el contrario las herramientas centradas en el documento, por ejemplo Delta [9], Uruz [11], GemCutter [10] o Steeper [1], guardan en todo momento una relación directa con el texto original de la GPC ya que se va etiquetando el texto original de esta pero finalmente se consigue un modelo semiformal con una representación estructurada de la GPC.

Con la herramienta que se ha desarrollado en este proyecto se consiguen las principales ventajas de cada uno de estos enfoques. Se consigue la ventaja del enfoque centrado en el documento ya que en todo momento habrá una relación entre los elementos, tanto del modelo semiformal como del workflow, con el texto original de la GPC. De este modo se consigue obtener un mapeo desde el texto original de la GPCP hasta las tareas del workflow final. A la vez se obtiene la ventaja del enfoque centrado en el modelo ya que se obtiene un modelo formal de la GPC representado en un workflow.
Otra diferencia con las herramientas de este tipo es la representación del modelo formal. En la actualidad las herramientas que siguen un enfoque centrado en el modelo utilizan formalismos como Asbru, Prodigy o Proforma.

En nuestra herramienta el formalismo elegido para representar la GPC ha sido workflow ya que en estudios actuales [2] se ha demostrado que los workflow son formalizaciones útiles para modelar aspectos de las GPC y han demostrado ser una aproximación efectiva para modelar parcialmente una GPC. Otra ventaja de esta herramienta respecto a otras del mismo ámbito es su aplicación futura usando un lenguaje workflow. Por ejemplo, en el caso de disponer de un motor de ejecución de workflow, las tareas ejecutadas podrían tener una comunicación directa con otros SIC.

Entre los trabajos futuros de este proyecto se encuentra extender su funcionalidad con el fin de conseguir los siguientes objetivos:

- Modelado del tiempo y cantidad. Para una descripción completa de un tratamiento el segmento RXO debe ir acompañado de un segmento que modelo el tiempo y la cantidad. Este segmento en HL7 v2.5 se corresponde con el segmento TQ1.

- Permitir trabajar con GPC en formato PDF. En este proyecto solo se permite trabajar con GPC en formato HTML. Una extensión que se podría añadir es que se pudiera trabajar con GPC en formato PDF consiguiendo al final los mismos objetivos que con una GPC en HTML.

- Integrar el motor de ejecución de workflow para que interactúe correctamente con otras aplicaciones. Al utilizar workflow como modelo de representación formal se puede integrar un motor de ejecución. De esta manera, teniendo relación entre las tareas del workflow y elementos CH4 y HL7, cuando es ejecuten tareas de este tipo se podrán se podrá comunicar directamente con otros SIC mediante los mensajes debidos y solicitar o recuperar información.

- Modificar los segmentos RXO con versiones posteriores a la 2.5 de HL7. El segmento RXO de HL7 que se ha utilizado en el proyecto corresponde con la versión 2.5 de este estándar. Un trabajo futuro será actualizar la implementación de este segmento con versiones posteriores de HL7 que vayan surgiendo.

- Un proceso de validación médica de la herramienta con expertos en la UCI.

BIBLIOGRAFÍA

[1] Marek Rzicka and Voijtech Svátek. Mark-up based analysis of narrative guidelines with the stepper tool. *Stud Health Technol Inform*, 101:132–136, 2004.

[2] Martínez Cano, Patricia. *Comprobación Retrospectiva Cumplimiento de Workflows Clínicos*. Tesis de Master. Universidad de Murcia. 2009

[3] Annette ten Teije, Silvia Miksch, Peter Lucas. *Computer-Based Medical Guidelines and Protocols: A Primer and Current Trends.* 2008.

[4] R. Kosara, S. Miksch: "*AsbruView: Capturing Complex, Time-Oriented Plans Beyond Flow-Charts*";in:"*Diagrammatic Representation and Reasoning*", P. Olivier et al. (Hrg.); Springer, Berlin, 2002.

[5] R. Kosara, S. Miksch, A. Seyfang, P. Votruba: "*Tools for Acquiring Clinical Guidelines in Asbru*";Society for Design and Process Science, 2002, S. 22 - 27.

[6] Bury, J., Fox, J., Sutton, D.. *The PROforma guideline specification language: progress and prospects*

[7] ter Hofstede, Arthur H. and van der Aalst, Wil M. (2005). *YAWL: yet another workflow language*. Information Systems 30(4):pp. 245-275.

[8] Erich Gamma, Richard Helm, Ralph Johnson y John Vlissides . *Patrones de diseño*. Addison-Wesley, 1995

REFERENCIAS WEB

[9] http://ieg.ifs.tuwien.ac.at/projects/delta/

[10] http://gem.med.yale.edu/GEM_CutterII/gem_cutterii.htm

[11] http://medinfo.ise.bgu.ac.il/medlab/ResearchProjects/RP_DeGeLhtm.htm

[12] http://perseo.dif.um.es/~aike/htm/ch4.htm

[13] http://www.openclinical.org/gmm_prodigy.html

[14] http://www.openclinical.org/gmm_asbru.html

[15] http://gem.med.yale.edu/gemarchive/default.htm

[16] http://gem.med.yale.edu/default.htm

[17] http://www.glif.org/glif_main.html

[18] http://www.hl7spain.org/VerPagina.asp?IDPage=0

[19] www.sageproject.net

[20] http://www.programacion.net/html/xml/htmdsssl/capitulo3/capitulo3.htm

[21] http://www.comunidadjava.com.ar/tec/disenojavasemvc/