Facultad de Informática

Grado de Ingeniería Informática

INGENIERÍA DEL SOFTWARE

Noviembre 2014

CONSTRUCCIÓN DE UNA APLICACIÓN BASADA EN MODELOS PARA LA GENERACIÓN DE GUÍAS DE TEST INTEGRADAS EN LA PLATAFORMA DE EJECUCIÓN DE GUÍAS DE PRÁCTICA CLÍNICA AIDE-GTP

MEMORIA

DATOS DEL ALUMNO/A

NOMBRE: EDUARDO

APELLIDOS: GARCÍA GARCÍA

DNI: 45816813 P

FECHA: 03/11/2014

DATOS DEL DIRECTOR/A

NOMBRE: TOMÁS ANTONIO

APELLIDOS: PÉREZ FERNÁNDEZ

DNI: 11948508P

DEPARTAMENTO: LENGUAJES Y SISTEMAS INFORMÁTICOS

FECHA: 03/11/2014

Resumen

Muchos sistemas de información disponen de mecanismos que intervienen en el estudio de las personas con el propósito de clasificarlas en distintos grupos según diversos criterios. A menudo se hace complejo establecer a cada uno de ellos en un determinado segmento e incluso determinar sus características de forma correcta. Una forma de estimar estos niveles para con los individuos, es mediante la utilización de Tests basados en diferentes tipos de preguntas como (respuestas múltiples, verdadero-falso, pares pregunta-respuesta, etc.).

Este proyecto aborda las transformaciones entre diferentes estándares para Test Informatizados (Aiken, Gift, IMS QTI), de manera automática, gracias técnicas de la Ingeniería Dirigida por Modelos (MDE - Model Driven Engineering), para así obtener un test capáz de formar parte de la plataforma de ejecución de Guías de Práctica Clínica Aide-GTP. Esto se pretende conseguir mediante la generación automática de Guías con Tests, partiendo de Test Informatizados en los formatos estándar Aiken y Gift.

La aplicación se ha desarrollado en el entorno de desarrollo Eclipse versión Indigo Service Release 2 (con el paquete Eclipse Modeling Tools apoyado en el Framework EMF) mediante el uso de lenguajes y gramáticas como Java, Xtext, Xtend, ATL Transformation Language y OCL (Object Constraint Language).

Abstract

Many educational Systems are provided with mechanisms involved in the study of people with the purpose of classifying them into different groups according to distinct criteria. It is often difficult to place the subject at their stage or even detect their characteristics correctly. One way to estimate these skills or levels is by means of Tests based on different types of questions (multiple-choice answers, true-false answers, question-answer pairs, etc.).

This project addresses automatic transformations among different standards for Computerized Test (Aiken, Gift, IMS QTI), through Model Driven Engineering (MDE), in order to be part of the platform for implementing Clinical Practice Guidelines Aide-GTP. These goals will be achieved by automatic generation of Test Guides, from Computerized Test in either Aiken or Gift standard formats.

The application has been developed through the IDE Eclipse Indigo Service Release 2 version (with Eclipse Modeling Tools package supported by the EMF framework) using languages and grammars like Java, Xtext, Xtend, ATL Transformation Language and OCL (Object Constraint Language).

Palabras Clave

Gramáticas	Test Adaptativos Informatizados	Aplicación basada en Modelos	Ingeniería Dirigida por Modelos	Guías de Práctica Clínica

Índice

Índice de Figuras

Índice de Tablas

1. Documento de Objetivos del Proyecto

En este apartado se describen los objetivos del proyecto que se va a desarrollar, así como la tecnología y la arquitectura que se va a utilizar y la planificación para su desarrollo.

1.1. Introducción

Muchos sistemas de información disponen de mecanismos que intervienen en el estudio de las personas con el propósito de establecer a estas en distintos grupos según diversos criterios. A menudo se hace complejo establecer a cada uno de ellos en un determinado segmento e incluso determinar sus características de forma correcta. Normalmente una forma de estimar estos niveles, es mediante la utilización de Test Adaptativos Informatizados basados en diferentes tipos de preguntas (respuestas múltiples, verdadero-falso, pares pregunta-respuesta, etc.).

Normalmente, test basados en ítems, preguntas de respuesta múltiple, pregunta verdadero-falsa, pares pregunta-respuesta, etc. son la opción más elegida a la hora de estimar el nivel o grupo en el cual una persona o colectivo puede ser incluido.

Una opción de evaluación y agrupación con este tipo de ítems, cada vez más extendida, [1] es la utilización de Test Adaptativos Informatizados (TAI) [2].

Se parte de la idea de crear un prototipo de aplicación capaz de transformar de manera automática Test Informatizados cuyo formato sean estándares utilizados en plataformas de evaluación, como son los formatos estándar Aiken, Gift e IMS QTI, en Guías de Test Informatizadas que formarán parte de la plataforma de ejecución de Guías de Práctica Clínica Aide-GTP. Una vez insertadas en dicha plataforma, estas Guías de Test podrán ayudar tanto a médicos como a pacientes en el estudio y diagnóstico de enfermedades. Esto se consigue gracias a la ingeniería dirigida por modelos (MDE-Model Driven Engineering) empleando altos niveles de abstracción para su desarrollo y la utilización de Modelos Independientes de la Plataforma (PIM).

En el presente proyecto se han diseñado e implementado cuatro módulos para este prototipo de aplicación (Importación/Exportación, Transformación entre metamodelos a MTest, Transformación entre metamodelos a MGPC, Generador de GPC).

Un primer módulo será el encargado de realizar las transformaciones T2M (Text to Model) y M2T (Model to Text) para tareas de "**importación/exportación**" consistentes en, a partir de un documento con un test en formato Aiken o Gift, crear un modelo que represente el mismo test conforme al meta modelo referente a su estándar, MAiken (metamodelo Aiken) o MGift (metamodelo Gift). A partir de un modelo conforme a los metamodelos MAiken o MGift, obtener un documento que represente ese modelo en su formato de test Aiken o Gift.

Un segundo módulo será el encargado de realizar las **transformaciones entre metamodelos a MTest**, dando un paso intermedio antes de pasar al meta modelo de Guías de Práctica Clínica (MGPC). Este paso consiste en la transformación de modelos conformes a metamodelos MAiken o MGift a modelos conformes a metamodelos MTest (meta modelo que será la notación canónica de todo test que se vaya a utilizar y que está basado en el estándar de IMS QTI).

El tercer módulo se ocupará de **transformar modelos conformes a metamodelos MTest en modelos conformes a metamodelos MGPC**. Esta es la principal aportación y pretende que los test de MTest se puedan ver como Guías de Práctica Clínica Informatizada (GPCI).

Por último, el cuarto módulo **transforma modelos conformes a metamodelos MGPC en Guías de Práctica Clínica (GPC)** en Aide-GTP.

1.2. Objetivos del proyecto

Se pueden separar los objetivos del proyecto en dos grandes grupos, los que a la propia aplicación se refieren y los que se refieren al ámbito personal de alumno, esto es, objetivos personales del alumno.

En cuanto a los primeros, hay que destacar que los principales objetivos son simplificar la tarea de transformación y generación de Guías de Práctica Clínica, aprovechando como elementos de partida, estándares de Test Informatizados. Concretamente el sistema tendrá que:

- Transformar documentos de test en formatos Aiken o Gift en modelos que representen el mismo test conforme a su meta modelo MAiken o MGift respectivamente. Transformación Text to Model (T2M), "Importar".

- A partir de un modelo conforme al metamodelo MAiken o MGift, obtener un documento de test que represente ese modelo en su correspondiente formato Aiken o Gift. Transformación Model to Text (M2T), "Exportar".

- Transformar modelos conformes a metamodelos MAiken o MGift en modelos conformes a metamodelos MTest (meta modelo que será la notación canónica de todo test que se vaya a utilizar y que está basado en el estándar de IMS QTI).

- A partir de modelos conformes a metamodelos MTest obtener modelos conformes a metamodelos MGPC. Esta es la principal aportación y pretende que los test de MTest se puedan ver como Guías de Práctica Clínica Informatizada (GPCI).

- Transformar modelos conformes a metamodelos MGPC en Guías de Práctica Clínica (GPC) en Aide-GTP[1].

En cuanto a los objetivos personales, destacar que como alumno del Grado en Ingeniería Informática, son investigar sobre distintas tecnologías para aprender su funcionamiento y sus beneficios, implementar y desarrollar un prototipo de aplicación como complemento añadido a la formación adquirida durante la carrera universitaria, enfrentarse con la realización de una aplicación enfocándola a un ámbito más profesional, y aprender a administrar todos los recursos disponibles (capital, tiempo, herramientas…), de tal forma que la finalización del proyecto se encuentre dentro del alcance, tiempo y costes definidos y planificados.

[1] Una versión accesible de la misma se puede ver (www.e-guidesmed.ehu.es – guider.ehu.es)

1.3. Decripción General

En este bloque se van a tratar los siguientes puntos: la duración del proyecto, su alcance, el trabajo que se va a desarrollar en él y una primera descripción del sistema.

1.3.1. Alcance del proyecto

En el alcance del proyecto se ha incluido el informe del alcance en el que se indica qué tareas pertenecen al desarrollo del mismo.

1.3.1.1. Informe del alcance

Entre el trabajo a desarrollar se incluye:

- Diseño e implementación del prototipo de aplicación.
- Prueba del buen funcionamiento de la aplicación.
- Las modificaciones oportunas durante la realización del proyecto para cumplir la funcionalidad deseada y satisfacer los objetivos del proyecto.

1.3.1.2. Descripción del sistema

La aplicación para la generación de Guías de Test quiere proporcionar un entorno automatizado, sencillo y accesible. En este prototipo a desarrollar en el presente proyecto se tratará de realizar:

- Un módulo de "importación - exportación", Text to Model (T2M) y Model to Text (M2T), que permita al usuario realizar dichas transformaciones de un modo sencillo y simple.

- Un módulo encargado de la transformación entre modelos conformes a metamodelos MAiken y MGift en modelos conformes a metamodelos MTest, Model to Model (M2M).

- Un módulo capaz de obtener modelos conformes a metamodelos MGPC (Modelos de Guías de Práctica Clínica), partiendo de modelos conformes a metamodelos MTest.

- Un módulo que realice la conversión desde un modelo conforme a MGPC a una Guía de Práctica Clínica (GPC) en en Aide-GTP.

1.4. Arquitectura de la aplicación

Para este proyecto, el prototipo de aplicación seguirá una arquitectura local, de aplicación de escritorio. Sin embargo, la idea final sería poder alojar la aplicación en un servidor de aplicaciones. De modo que estuviera constituida por la arquitectura cliente-servidor, basada en la separación de tipo lógico, donde un cliente es el que inicia un requerimiento de servicio y consume dicho servicio. Por su parte, el servidor es cualquier recurso encargado de responder a los requerimientos del cliente, esto es, un proveedor de servicios.

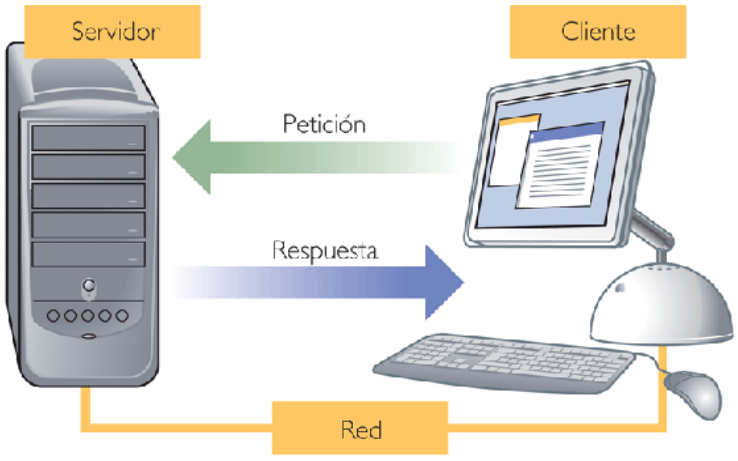


Figura 1. Arquitectura cliente-servidor

Dicho esto, siguiendo esta arquitectura general, el prototipo de la aplicación presenta la arquitectura concreta que se puede apreciar en la Figura 2.

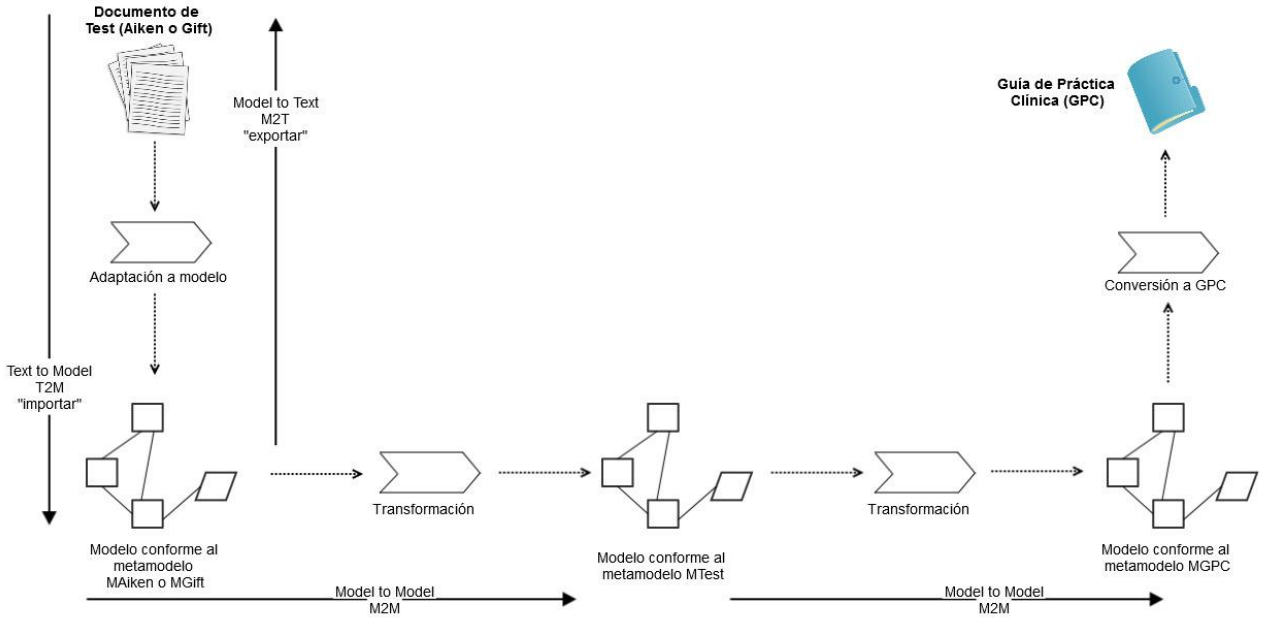


Figura 2. Arquitectura de la Aplicación para la generación de Guías de Test

De acuerdo a la funcionalidad del primer módulo, el primer paso conlleva el diseño y creación de los diferentes metamodelos o ".ecore", conformes a cada estándar, meta modelo MGift y meta modelo MAiken. Además de estos, se ha de diseñar y desarrollar un Lenguaje Específico de Dominio (Domain Specific Language - DSL) para cada uno de los dos formatos estándar de los que se partirá, Aiken y Gift. Para especificar dichos lenguajes se escribirá una gramática para cada formato escrita en el lenguaje de la gramática Xtext. Gracias a ello se podrá obtener la transformación Text to Model (T2M) inicial. Por otra parte, se crearán "plantillas de código" escritas en el lenguaje Xtend que se encargarán de, partiendo de un modelo conforme al meta modelo MAiken o MGift, recorrer esos modelos y obtener los datos o información necesaria para luego representarlos como documentos de test en formato Aiken o Gift. La funcionalidad es la inversa a la anterior, transformación Model to Text (M2T).

Además de lo anterior se podrá transformar modelos conformes a los metamodelos MAiken y MGift en modelos conformes a MTest (modelo de representación canónica de todo test a utilizar y basado en el estándar IMS QTI). Se procederá a meta modelar el meta modelo o ".ecore" para MTest, que será un meta modelo independiente de la plataforma (Platform Independent Model - PIM). Para llevar a cabo estas transformaciones ser hará uso del lenguaje de transformación de modelos ATLAS (ATL Transformation Language). En el campo de la Ingeniería Dirigida por Modelos (MDE), ATL proporciona mecanismos para producir modelos destino desde modelos fuente. El meta modelo para MGPC (Meta Modelo Guía de Práctica Clínica) ha sido facilitado por Raúl Barrena, de modo que, se desarrollará la transformación necesaria entre modelos conformes al meta modelo MTest y modelos conformes al meta modelo MGPC, al igual que antes mediante el uso del lenguaje ATL. El último paso, que consiste en la adaptación de un modelo conforme a MGPC en una Guía de Práctica Clínica, se realizará gracias a un sistema que realiza dicha conversión en en Aide-GTP.

Para el desarrollo de este prototipo se utilizará la tecnología del IDE Eclipse versión Indigo Service Release 2 (con el paquete Eclipse Modeling Tools apoyado en el Framework EMF) combinado con el uso de lenguajes de programación y gramáticas como Java, Xtext, Xtend, ATL Transformation Language y OCL (Object Constraint Language).

1.5. Herramientas y tecnologías de la aplicación

A continuación se describen las herramientas que serán utilizadas para desarrollar el proyecto: lenguajes utilizados, entorno de desarrollo sobre el que se va a trabajar y las tecnologías necesarias para lograr su éxito.

1.5.1. Herramientas

Para el desarrollo del presente proyecto se hará uso de diferentes lenguajes de programación. El lenguaje de programación en torno al cual gira la aplicación es Java. Ésta será implementada en el entorno de desarrollo (IDE) Eclipse versión Indigo bajo el Framework de desarrollo EMF (Eclipse Modeling Framework). Además se hará uso de los lenguajes Xtext (entorno Open Source para diseñar e implementar tu propio lenguaje de programación y Lenguaje Especifico de Dominio (DSL), Xtend (lenguaje de programación estáticamente tipado que se traduce en código fuente Java), ATLAS (ATL Transformation Languaje), herramienta y lenguaje usado para la transformación de modelos y OCL (Object Constraint Languaje) lenguaje declarativo que se usará para describir reglas sobre los modelos UML desarrollados. Estas herramientas han sido seleccionadas por querer aprender más acerca de ellas, por ser herramientas Open Source, o de código abierto, y por el gran potencial que ofrecen. El framework de modelado de Eclipse o Eclipse Modeling Framework (EMF) ha sido elegido por considerarse una herramienta esencial para el Desarrollo de Software dirigido por Modelos (MDE) y querer adquirir más conocimientos acerca de esta tecnología, por otro lado, la utilización del IDE de desarrollo Eclipse era un requerimiento del proyecto.

1.5.2. Tecnologías

MDE: (http://en.wikipedia.org/wiki/Model-driven_engineering) Model Driven Engineering, se define como Ingeniería dirigida por Modelos, metodología para el desarrollo de software centrada en el diseño y creación de modelos de dominio para cubrir y modelar una necesidad o solución ante un problema para un dominio en particular. Trata de alcanzar el nivel de abstracción en la especificación de programas e incrementar la automatización en su desarrollo. Aporta beneficios como productividad, portabilidad, interoperabilidad, labores de mantenimiento y documentación, calidad aumentada y un alto nivel de reusabilidad.

Eclipse: (https://www.eclipse.org/) Se define como un entorno de desarrollo integrado (IDE), de código abierto y multiplataforma. Soporta varios lenguajes de programación tales como Java, PHP, Ada, C, C++, Python, JavaScript, etc. y la integración de diferentes Framework, que van desde marcos de trabajo para el desarrollo de aplicaciones gráficas hasta editores de diagramas UML o entornos de desarrollo para aplicaciones Web. Este potente software, entre otros, permite a los desarrolladores crear sitios y aplicaciones Web, así como aplicaciones de servidor, aplicaciones de escritorio o aplicaciones Android entre otros, que podrán ser usados en cualquier entorno soportado por su correspondiente plataforma.

EMF: (http://www.eclipse.org/modeling/emf/) Eclipse Modeling Framework, Framework de modelado y generación de código, utilizado en el Desarrollo de Software dirigido por Modelos o Ingeniería dirigida por Modelos (MDE), para desarrollar herramientas y aplicaciones centradas en modelos de datos y Modelos de Dominio. Partiendo de un modelo, cuya especificación será un fichero en formato XMI, gracias a EMF, sus herramientas y soporte de ejecución, se podrán obtener clases y código Java para permitir su visualización y manipulación.

Xtext: (http://www.eclipse.org/Xtext/) Se cataloga como un Framework para el desarrollo de lenguajes de programación y lenguajes específicos de dominio (Domain Specific Language - DSL). Proporciona una infraestructura completa para el desarrollo de lenguajes, desde enlazador de código, pasando por compilador y hasta intérprete de dicho lenguaje. Se está desarrollando como parte del proyecto Eclipse Modeling Framework (EMF) y además ofrece una perfecta integración en el IDE de desarrollo Eclipse.

Xtend: (http://www.eclipse.org/xtend/) Es un lenguaje de programación de alto nivel y propósito general para su uso en la máquina virtual Java, desarrollado y diseñado para Eclipse. Es muy similar a Java en términos tanto semánticos como sintácticos. Xtend se originó de Xtext y se integra perfectamente con este y con todas las bibliotecas de Java. Su principal atractivo es la reducción sustancial de código a escribir frente a Java (elimina declaración de variables intermedias, bucles, definición de variables de retornos, métodos getter y setter, etc.). Este código es compilado y reescrito en un fichero .java.

Cabe hacer mención de otras tecnologías como procesadores de texto y herramientas de gestión, presentación y planificación que han sido necesarias. Éste circulo está constituido por los programas: Microsoft Office Word, Microsoft Office PowerPoint, Microsoft Project y Microsoft Visio, todos ellos en su versión de 2010.

1.6. Estructuración y calendario del sistema

En este punto se explica la forma en la que se va a desarrollar el proyecto, (el ciclo de vida que va a seguir), y el método de trabajo establecido.

1.6.1. Ciclo de vida y método de trabajo.

El desarrollo del proyecto se hará en etapas (determinadas posteriormente en el diagrama EDT) siguiendo un **ciclo de vida mediante prototipos de manera incremental.** (Se iteran varios ciclos de vida en cascada hasta lograr satisfacer los requisitos del sistema). Este modelo de ciclo de vida es caracterizado por construir e ir añadiendo tras cada nueva iteración funcionalidades del programa. Esta forma de trabajo se consigue gracias a la separación en módulos de las diferentes funcionalidades que ha de cumplir el sistema. Así se consigue gradualmente aumentar las capacidades del software. Proporciona algunos beneficios tales como los que se describen a continuación:

- Al construir por etapas o ciclos, los errores graves solo afectarán a la parte final.
- Tras cada iteración se obtiene una nueva versión, así, a medida que se avanza se podrá controlar la calidad.
- Construir un sistema pequeño tiene menos riesgo que construir un sistema grande.
- Facilita la labor del desarrollo con la conocida filosofía de divide y vencerás.
- En contra de los beneficios descritos, este ciclo de vida necesita de una gestión algo complicada.

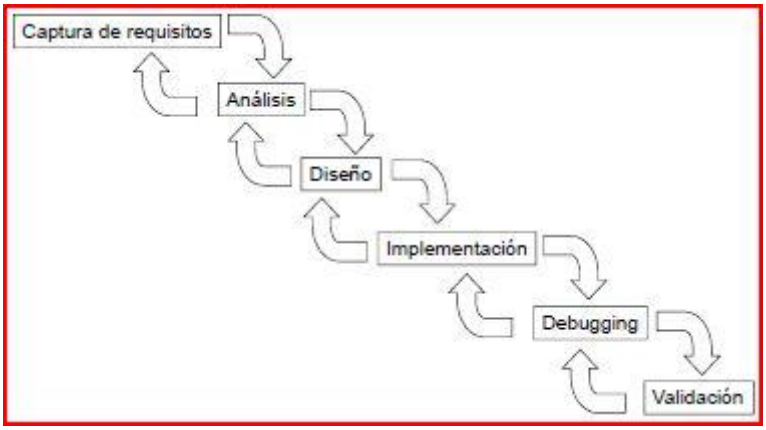


Figura 3. Ciclo de vida en cascada

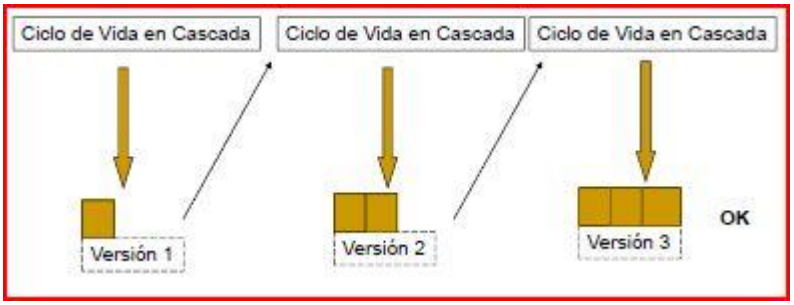


Figura 4. Ciclo de vida incremental

El **método de trabajo** será adaptable, dependiendo de la tarea y la etapa que se esté desarrollando y lo avanzado que se encuentre el proyecto, pudiéndose paralizar o posponer el desarrollo por un periodo determinado de tiempo por causas externas.

1.6.2. Diagrama de estructura de descomposición del trabajo (EDT)

En este apartado se describen brevemente las tareas en las que se ha dividido el proyecto. Estas tareas aparecen estructuradas en el diagrama EDT de descomposición de trabajo.

La primera tarea corresponde a la planificación inicial del proyecto, dónde se engloban actividades cómo la recopilación de información y la primera reunión con el director del proyecto.

La segunda tarea, la redacción del documento de viabilidad, se desarrolla con el fin de definir las necesidades que el nuevo sistema debe satisfacer y deducir a partir de ellas las operaciones y servicios que debe proveer la aplicación.

En la captura de requisitos y el análisis se intenta buscar una solución apropiada y óptima para cada servicio o funcionalidad diseñada.

Por otra parte, en la fase de diseño del sistema se detallará tanto el diseño de la arquitectura, como el diseño de los Modelos de Dominio y metamodelos, diseño de las adaptaciones y transformaciones entre modelos, diseño de sus gramáticas y también el de los módulos de la aplicación.

La siguiente tarea le corresponde a la implementación, cuya finalidad es la de implementar los módulos de la aplicación que se diseñaron en la fase anterior.

Las tareas de documentación y presentación consisten en documentar el trabajo realizado por medio de la redacción de la memoria.

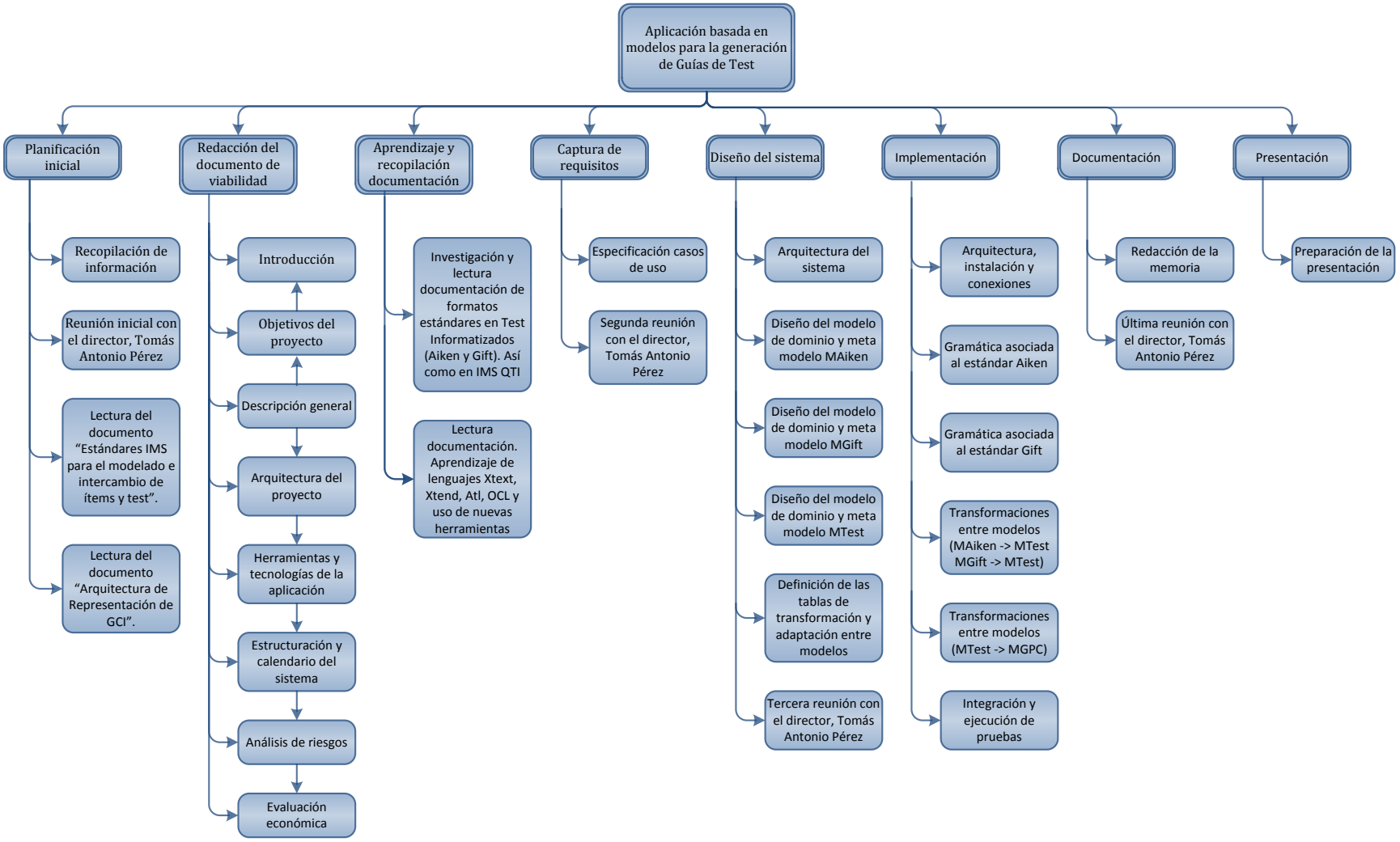


Figura 5. Diagrama de estructura de descomposición del trabajo (EDT)

1.6.3. Calendario

Teniendo en cuenta el calendario que presentamos a continuación se realizará la planificación temporal del proyecto.

Figura 6. Calendario

La fecha de inicio del proyecto se establece el día 5 de mayo, día en el que se comienza a realizar la recopilación de información, lectura de documentos y una primera fase de planificación.

Inicialmente se dedicarán entre tres y cuatro horas diarias, mínimo, al proyecto. Los fines de semana en cambio, se dedicarán una o dos horas cada día. Así mismo, en periodo de vacaciones, durante el mes de agosto, se estimará una carga de trabajo de cinco a seis horas diarias.

El proyecto se presentará ante el tribunal en la convocatoria de septiembre, por lo que a mediados del mes de julio y el mes de agosto se realizará una carga de trabajo mayor (siete horas al día aproximadamente) para finalizar a tiempo todas las partes pendientes del proyecto, o que hayan acarreado mayores dificultades.

1.6.3.1. Esquema de planificación de tareas

A continuación se presenta una tabla en la que se enumeran todas las tareas con las horas que se han planificado para realizarla.

Tabla 1. Planificación de tareas

1. Planificación inicial	**15 horas**
2. Recopilación de información	14 horas
3. Reunión con el director	1 hora
4. Redacción del documento de viabilidad	**28 horas**
5. Introducción	1 hora
6. Objetivos del proyecto	2 horas
7. Descripción general	4 horas
8. Arquitectura del proyecto	1 hora
9. Herramientas y tecnologías de la aplicación	4 horas
10. Estructuración y calendario del sistema	8 horas
11. Análisis de riesgos	4 horas
12. Evaluación económica	4 horas
13. Aprendizaje y recopilación de información	**30 horas**
14. Implementación aplicación	**180 horas**
15. Instalación y compatibilidades	3 horas
16. Metamodelado de los modelos	22 horas
17. Gramáticas asociadas a los formatos	75 horas
18. Transformaciones entre modelos	80 horas
19. Integración y pruebas	**20 horas**
20. Documentación	**50 horas**
21. Redacción de la memoria	50 horas
22. Preparación de la presentación	**10 horas**

1.6.4. Fecha de inicio y finalización prevista

A continuación se puede apreciar la tabla con la fecha de inicio y final del proyecto.

Tabla 2. Inicio y finalización del proyecto (estimación)

Fecha de inicio	Fecha de finalización
5 de mayo de 2014	25 de agosto de 2014

1.6.5. Diagrama de Gantt

	Nombre de la tarea	Comienzo	Fin	Duración
1	**Planificación inicial**	**lun 05/05/14**	**jue 22/05/14**	**15 hrs**
2	Recopilación de información	lun 05/05/14	jue 22/05/14	14 hrs
3	Reunión con el director	mar 20/05/14	mar 20/05/14	1 hr
4	**Redacción del documento de viabilidad**	**mar 20/05/14**	**lun 02/06/14**	**28 hrs**
5	Introducción	mar 20/05/14	mié 21/05/14	1 hr
6	Objetivos del proyecto	mié 21/05/14	mié 21/05/14	2 hrs
7	Descripción general	jue 22/05/14	dom 25/05/14	4 hrs
8	Arquitectura del proyecto	lun 26/05/14	lun 26/05/14	1 hr
9	Herramientas y tecnologías de la aplicación	lun 26/05/14	mar 27/05/14	4 hrs
10	Estructuración y calendario del sistema	mar 27/05/14	jue 29/05/14	8 hrs
11	Análisis de riesgos	vie 30/05/14	vie 30/05/14	4 hrs
12	Evaluación económica	sáb 31/05/14	lun 02/06/14	4 hrs
13	**Aprendizaje y recopilación de información**	**lun 26/05/14**	**lun 16/06/14**	**30 hrs**
14	**Implementación aplicación**	**mar 27/05/14**	**dom 03/08/14**	**180 hrs**
15	Instalación y compatibilidades	mar 27/05/14	mié 28/05/14	3 hrs
16	Meta modelado de los modelos	lun 02/06/14	mar 10/06/14	22 hrs
17	Gramáticas asociadas a los formatos	mié 11/06/14	mié 02/07/14	75 hrs
18	Transformaciones entre modelos	jue 03/07/14	dom 03/08/14	80 hrs
19	**Integración y pruebas**	**lun 04/08/14**	**lun 11/08/14**	**20 hrs**
20	**Documentación**	**lun 04/08/14**	**jue 21/08/14**	**50 hrs**
21	Redacción de la memoria	lun 04/08/14	jue 21/08/14	50 hrs
22	**Preparación presentación**	**jue 21/08/14**	**lun 25/08/14**	**10 hrs**

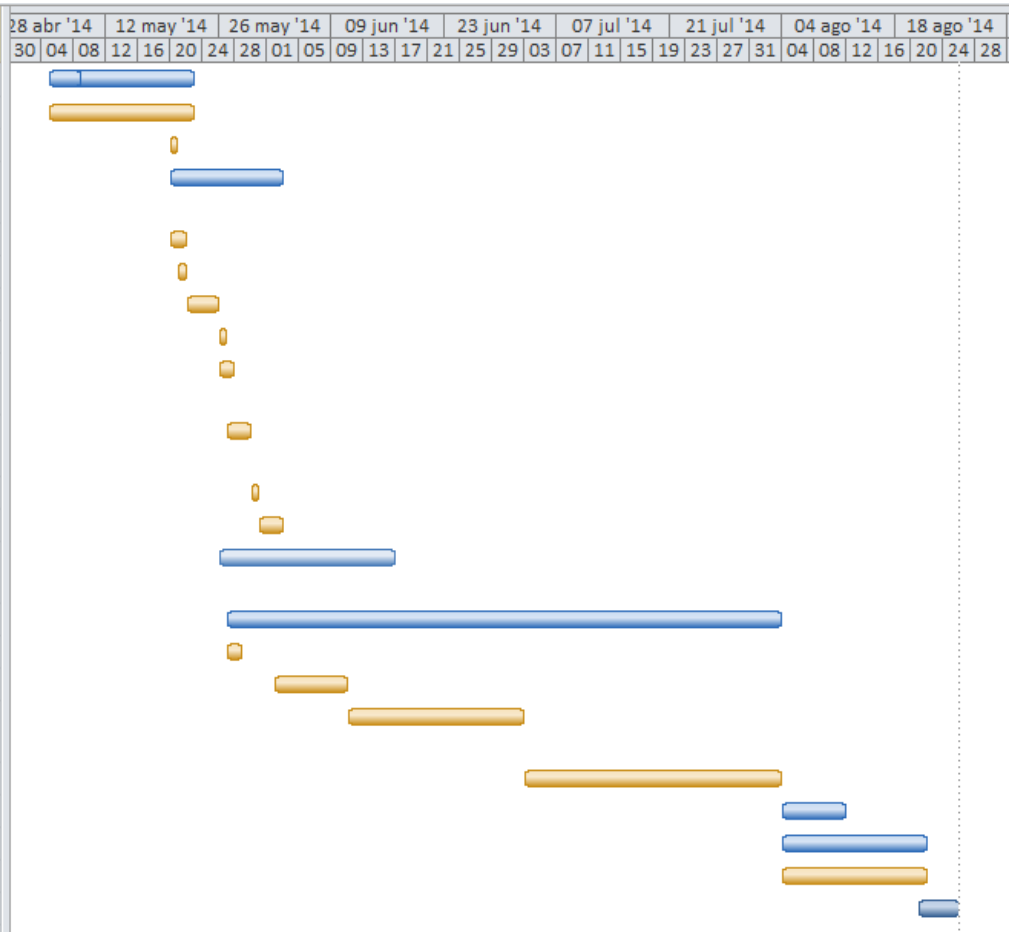


Figura 7. Diagrama de Gantt

1.7. Análisis de riesgos

Seguidamente se exponen los principales riesgos identificados. Para que el proyecto se pueda llevar a cabo dentro del plazo previsto, se crea un plan de contingencia que se pondrá en marcha si se diera alguno de los riesgos descritos. En el siguiente punto se evalúa la cuantificación de los riesgos identificados.

1.7.1. Identificación de los riesgos y plan de contingencia

El desarrollo del proyecto presenta factores que conllevan una serie de riesgos que en caso de no ser detectados a tiempo, podrían afectar gravemente al desarrollo del mismo. Donde hay un factor de riesgo considerable, se desarrolla un plan de contingencia, (una vez identificados los riesgos), que se activará automáticamente al detectar algún síntoma del riesgo.

Los distintos riesgos que podemos encontrarnos en el desarrollo del proyecto y su plan de contingencia son los que se describen a continuación:

1. **Problemas con el software.** Puede que las distintas versiones no sean compatibles, que se produzcan problemas con virus, perdida de datos….
   Efecto: Este riesgo podrá causar la pérdida de información y de trabajo que supondrá su nueva realización.
   Modo de evitar / mitigar: Instalación y actualización periódica de software antivirus, creación periódica de backups o copias de seguridad.
   Solución: Reinstalar el software que de problemas. En caso extremo formatear el equipo, y recuperar una copia de seguridad.

2. **Problemas con el hardware.** Puede que se produzca un fallo en cualquiera de los equipos utilizados en el desarrollo de la aplicación.
   Efecto: Este riesgo al igual que los problemas de software causará la pérdida de información y de trabajo que habrá que volver a realizar.
   Modo de evitar / mitigar: Buena conservación de los equipos (limpieza, cuidado…).
   Solución: Disponer de otro equipo de similares características a la mayor brevedad posible. O tener los medios o la capacidad para poder reparar el equipo afectado en muy corto plazo.

3. **Dificultades en la implementación por la falta de conocimiento.** Al utilizar nuevas tecnologías no utilizadas anteriormente en el desarrollo del proyecto, no se asegura el suficiente conocimiento necesario para la realización del mismo.
   Efecto: Incremento de las horas de trabajo que causará un retraso en la finalización del mismo.
   Modo de evitar / mitigar: Conseguir una buena documentación complementaria.
   Solución: Emplear horas extras en adquirir conocimientos sobre la materia.

4. **Indisposición o enfermedad.** Sufrir algún tipo de malestar o enfermedad.
   Efecto: Retraso en la finalización del proyecto.
   Solución: En ese periodo si es posible, realizar labores pequeñas y, posteriormente intentar recuperar las horas aumentando la jornada laboral, si esto no fuese posible, habría que retrasar la fecha de finalización del proyecto.

5. **Compromisos personales ajenos al proyecto laboral.** Puede que durante la realización del proyecto se den distintos compromisos como puede ser un imprevisto familiar, entrevistas de trabajo y procesos de selección o un viaje no planificado con anterioridad.
   Efecto: Retraso en la finalización del proyecto.
   Solución: Intentar recuperar posteriormente las horas perdidas en ese periodo.

6. **Mala planificación temporal.**
   Efecto: Retraso adicional en la fecha de entrega del proyecto.
   Modo de evitar / mitigar: Realizar una estimación lo más acertada posible valorando los conocimientos iniciales.
   Solución: Ampliar la jornada de trabajo introduciendo horas extras para recuperar la mala estimación prevista o retrasar la fecha de entrega del proyecto.

1.7.2. Cuantificación de riesgos

La siguiente tabla clasifica los riesgos definidos anteriormente en función de la probabilidad de que sucedan y del impacto que causarían en el desarrollo del proyecto.

Tabla 3. Cuantificación de riesgos

Riesgo	Probabilidad	Impacto
Problemas con el software	Baja	Medio
Problemas con el hardware	Baja	Medio
Dificultades en la implementación por la falta de conocimiento	Alta	Alto
Indisposición o enfermedad	Baja	Medio
Compromisos personales ajenos al proyecto	Media	Medio
Mala planificación temporal	Alta	Medio

1.8. Evaluación económica

Los gastos del proyecto "Aplicación basada en modelos para la generación de Guías de Test integrados en una plataforma de ejecución de Guías de Práctica Clínica", los podemos dividir en tres grupos: salario, gastos de software e inmovilizado de material:

1.8.1. Salario

La estimación temporal del proyecto se ajusta a unas 333 horas. El encargado de la realización del proyecto será un único analista programador, cuyo salario se establecerá en 15 euros por hora. A la estimación temporal, se le podrán añadir un número de horas dedicadas al aprendizaje de las nuevas herramientas utilizadas por parte del trabajador, por lo que el gasto descrito con anterioridad podría verse incrementado. Para ese aprendizaje se creará una bolsa de horas (30h) valorada en 450 euros.

El desglose de horas será el siguiente:

Horas totales → Horas realización proyecto + Bolsa horas aprendizaje = 333h + 30h = **363 horas**

Al sueldo arriba indicado se le tendrá que descontar los porcentajes correspondientes al IRPF y a la cotización para la Seguridad Social por lo que el suelo neto del trabajador será el siguiente:

Salario Neto → 363 horas * 15 euros/hora = 5445 euros brutos – 6% Cotización SS – 15% Cotización IRPF = 5445 euros – 326,70 euros – 816,75 euros = **4301,55euros**

1.8.2. Gastos de software

El software de código abierto Open Source es de libre distribución y no repercute gasto alguno. Por otra parte, los programas de Microsoft de los que se harán uso para el desarrollo del proyecto no supondrán ningún coste añadido ya que han sido adquiridos en la plataforma Microsoft DreamSpark for Academic Institutions de la Facultad de Informática de San Sebastián, gracias al acuerdo establecido entre la Universidad del País Vasco (UPV/EHU) y Microsoft.

(http://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?ws=0060a29a-689b-e011-969d-0030487d8897&vsro=8)

1.8.3. Inmovilizado de material y otros

Para el desarrollo del proyecto necesitaremos disponer de los siguientes dispositivos:

- Ordenador portátil → 945 euros.
- Conexión a Internet → 35 €/mes.
- Memoria USB → 12 euros.
- Impresora → 60 euros.

Ya que la duración del proyecto se estima en cuatro meses, tendremos un gasto fijo correspondiente a la conexión a Internet durante ese tiempo. El gasto de inmovilizado de material, constituido por el ordenador portátil con el que se llevará a cabo el desarrollo de la aplicación, la impresora y la memoria USB, será el siguiente:

La vida útil del ordenador portátil está estimada en 6 años, por lo que la amortización anual es de 945 euros / 6 años = 157,50 euros. Ese ordenador es usado para la realización del proyecto el 60% del tiempo que permanece encendido. El proyecto dura cuatro meses, por lo que la amortización del portátil es:
(157,50 euros * 4 meses * 0,6) / 12 meses = 31,50 euros.

La impresora valorada en 60 euros tiene una vida útil de 5 años, y en consecuencia una amortización anual de 12 euros. El uso de la impresora es de un 40% para la realización del proyecto. Por tanto su amortización es:
(12 euros * 4 meses * 0,4) / 12 meses = 1,60 euros.

La memoria USB, con un coste de 12 euros, tiene una vida útil de 5 años y su uso es exclusivo para el proyecto. Su amortización anual es de 2,40 euros. Su amortización es:
(2,40 euros * 4 meses) / 12 meses = 0,80 euros.

En conclusión, se tendrá el siguiente gasto de material:

Inmovilizado de material y amortización→ (35 euros/mes * 4 meses) + 31,50 euros + 1,60 euros + 0,80 euros = **173,90 euros**

Como otros gastos se contemplan, gasto de luz, folios, tinta de impresora, desplazamientos, teléfono, etc. Dichos gastos ascienden a unos 80 euros.

1.8.4. Beneficios obtenidos

La aplicación basada en modelos para la generación de Guías de Test aportará unos beneficios directos iniciales ya que se trata de una aplicación destinada al sector educativo/sanitario en el ámbito de la enseñanza y la salud. Se espera que tenga una buena aceptación y una demanda incremental la cual reporte unos futuros ingresos. Incluso, si la aplicación fuese colgada en un servidor de uso público, se podrían vender espacios para introducir en ellos banners, anuncios y propaganda con el fin de obtener beneficios.

1.8.5. Coste total del proyecto

El cálculo del coste total del proyecto será el siguiente:

Coste total → = Salario Neto analista/programador + Seguridad Social + IRPF + Inmovilizado de material y amortización + Otros gastos = 4301,55 € + 326,70 € + 816,75 € + 173,90 € + 80 € = **5698,90 euros**

1.8.6. Cálculo de retorno de la inversión

Analizando el coste directo que tendría el proyecto, y contemplando que los beneficios se obtendrían gracias a los derechos de desarrollo por las compras de la aplicación principalmente y en añadido, por la publicidad introducida en la misma, se procederá a calcular en que momento el proyecto pasaría a ser rentable.

Si las ventas de la aplicación ascendiesen a 600 unidades rondando un precio de venta de 18 euros / unidad, se obtendrían unos ingresos valorados en 10800 euros. Así los beneficios serían 10800 € - 9073,90 € = 1726,10 €.

Pero si se jugara con la ley de la oferta y la demanda y el precio de venta disminuyera a los 12 euros / unidad, generándose así unas ventas de 1000 unidades, los ingresos ascenderían hasta los 12000 euros. De este modo los beneficios serían mayores siendo estos de 12000 € - 9073,90 € = 2926,10 €.

Además se podría obtener beneficios por publicidad. Dicha publicidad, sería insertada progresivamente para que no afectase de manera brusca al diseño de la aplicación. De esta manera se conseguirán obtener paulatinamente unos ingresos mayores, amortizando así el gasto de creación inicial.

2. Cuerpo de la memoria

En este apartado quedan documentados los puntos relevantes y decisivos que han sido necesarios para la correcta realización de éste proyecto.

2.1. Introducción

Muchos sistemas de información disponen de mecanismos que intervienen en el estudio de las personas con el propósito de establecer a estas en distintos grupos según diversos criterios. A menudo se hace complejo establecer a cada uno de ellos en un determinado segmento e incluso determinar sus características de forma correcta. Normalmente una forma de estimar estos niveles, es mediante la utilización de Test Adaptativos Informatizados basados en diferentes tipos de preguntas (respuestas múltiples, verdadero-falso, pares pregunta-respuesta, etc.).

Normalmente, test basados en ítems, preguntas de respuesta múltiple, pregunta verdadero-falsa, pares pregunta-respuesta, etc. son la opción más elegida a la hora de estimar el nivel o grupo en el cual una persona o colectivo puede ser incluido.

Una opción de evaluación y agrupación con este tipo de ítems, cada vez más extendida, [1] es la utilización de Test Adaptativos Informatizados (TAI) [2].

Se parte de la idea de crear un prototipo de aplicación capaz de transformar de manera automática Test Informatizados cuyo formato sean estándares utilizados en plataformas de evaluación, como son los formatos estándar Aiken, Gift e IMS QTI, en Guías de Test Informatizadas que formarán parte de la plataforma de ejecución de Guías de Práctica Clínica Aide-GTP. Una vez insertadas en dicha plataforma, estas Guías de Test podrán ayudar tanto a médicos como a pacientes en el estudio y diagnóstico de enfermedades. Esto se consigue gracias a la ingeniería dirigida por modelos (MDE-Model Driven Engineering) empleando altos niveles de abstracción para su desarrollo y la utilización de Modelos Independientes de la Plataforma (PIM).

En el presente proyecto se han diseñado e implementado cuatro módulos para este prototipo de aplicación (Importación/Exportación, Transformación entre metamodelos a MTest, Transformación entre metamodelos a MGPC, Generador de GPC).

Un primer módulo será el encargado de realizar las transformaciones T2M (Text to Model) y M2T (Model to Text) para tareas de "**importación/exportación**" consistentes en, a partir de un documento con un test en formato Aiken o Gift, crear un modelo que represente el mismo test conforme al meta modelo referente a su estándar, MAiken (metamodelo Aiken) o MGift (metamodelo Gift). A partir de un modelo conforme a los metamodelos MAiken o MGift, obtener un documento que represente ese modelo en su formato de test Aiken o Gift.

Un segundo módulo será el encargado de realizar las **transformaciones entre metamodelos a MTest**, dando un paso intermedio antes de pasar al meta modelo de Guías de Práctica Clínica (MGPC). Este paso consiste en la transformación de modelos conformes a metamodelos MAiken o MGift a modelos conformes a metamodelos MTest (meta modelo que será la notación canónica de todo test que se vaya a utilizar y que está basado en el estándar de IMS QTI).

El tercer módulo se ocupará de **transformar modelos conformes a metamodelos MTest en modelos conformes a metamodelos MGPC**. Esta es la principal aportación

y pretende que los test de MTest se puedan ver como Guías de Práctica Clínica Informatizada (GPCI).

Por último, el cuarto módulo **transforma modelos conformes a metamodelos MGPC en Guías de Práctica Clínica (GPC)** en Aide-GTP.

2.1.1. Objetivos del proyecto y justificación de la elección

En cuanto a los objetivos del proyecto se refiere, se podrían separar en dos grandes grupos, por un lado los que se refiere a la propia aplicación, y por otro los referentes al propio alumno que desarrolla el presente proyecto.

Si hablamos del primer grupo, esto es, de los objetivos del proyecto en sí mismo, podemos destacar los siguientes:

- Simplificar la tarea de transformación y generación de Guías de Práctica Clínica
- Generar modelos conformes a estándares para Test Informatizados
- Transformar y adaptar estos modelos para su integración en una plataforma de ejecución de Guías de Práctica Clínica Aide-GTP.

Si por el contrario hablamos de los objetivos del segundo grupo, esto es, los referentes al ámbito personal como alumno del Grado en Ingeniería Informática, se destacan los siguientes:

- Investigar sobre distintas tecnologías para aprender su funcionamiento y sus beneficios.

- Implementar y desarrollar una aplicación como complemento añadido a la formación adquirida durante la carrera universitaria.

- Enfrentarse con la realización de dicha aplicación enfocándola a un ámbito más profesional.

- Administrar todos los recursos disponibles (capital, tiempo, herramientas…) de tal forma que la finalización del proyecto se encuentre dentro del alcance, tiempo y costes definidos y planificados.

La creación partiendo de cero de un nuevo proyecto, es un reto profesional para muchos individuos que trabajan en una empresa, y es una tarea difícil de llevar a cabo para una sola persona. Sin embargo, tras la finalización del proyecto, éste otorga una experiencia muy positiva y satisfactoria para un futuro cercano como profesional cualificado.

Además de toda la experiencia adquirida, éste proyecto sirve para ampliar los conocimientos en los lenguajes con los que se ha trabajado (Java, Xtext, Xtend, ATL, OCL), el entorno de desarrollo empleado (Eclipse), así como los de su arquitectura y tecnología basados en la Ingeniería Dirigida por Modelos. Sin lugar a dudas estos conocimientos serán útiles en una futura vida profesional.

2.2. Estado del arte

No se ha encontrado ninguna herramienta de similares aptitudes o características requeridas por el proyecto, por lo que no se ha procedido a la realización de un estudio de las mismas. Tal situación se debe a la minuciosidad del proyecto, y a lo concreto que debía ser, en el campo de la adaptación de estándares de Test Informatizados.

Este punto contempla y analiza los diferentes lenguajes con los que se desarrollará la aplicación. Finalmente se explican los motivos de la elección del IDE Eclipse así como del uso de los lenguajes Java, Xtext, Xtend, ATL y OCL.

2.2.1. C/C++

Es un lenguaje de programación compuesto por elementos del lenguaje de bajo nivel, así como, elementos del lenguaje de alto nivel. Se amolda a cualquier máquina que soporte el lenguaje de programación C. En principio éste lenguaje se creó para programar sistemas operativos, compiladores, intérpretes de comandos, etc. aunque hoy en día su uso está muy extendido y se puede usar en casi cualquier programa. Su uso más extendido se puede encontrar en entornos UNIX, ya que se diseño en él.

Utiliza patrones para desarrollar clases genéricas, obteniendo de esta manera la base de la biblioteca de patrones estándar. Esta biblioteca incluye patrones para mapas, pilas, contenedores, y muchos otros, además de una gran cantidad de algoritmos para que los usuarios se beneficien de ellos y los adapten a sus necesidades.

Entre sus ventajas cabe destacar su facilidad de aprendizaje, debido al empleo de objetos sencillos. Así como la característica de que dicho lenguaje representa los ejecutables más rápidos que se pueden realizar para una máquina. Por otro lado, sus desventajas más notorias podrían ser el hecho de que la falta de estructuración y de abstracción supone una dificultad a la hora de entender el código escrito por otro programador. Y que el propio lenguaje es poco estricto con las comprobaciones de los tipos de datos.

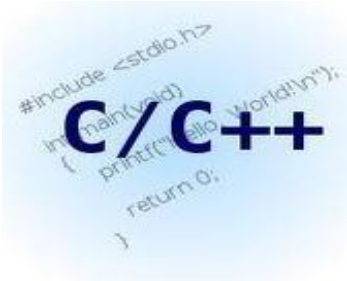


Figura 8. C++

2.2.2. JetBrains Meta Programming System

JetBrains Meta Programming System (MPS) es una herramienta o entorno de desarrollo integrado (IDE) para lenguajes, que permite crear y diseñar lenguajes específicos de dominio o DSL (Domain Specific Languages). Con esta tecnología se pueden superar los límites establecidos por los analizadores de lenguaje que conocemos, y construir nuestro propio lenguaje específico de dominio. Además junto con él se puede definir un generador de código completo para el lenguaje específico de dominio. De este modo se puede obtener código fuente en diferentes lenguajes de programación, como pueden ser Java, XML o C… entre otros.

Figura 9. JetBrains MPS (Meta Programming System)

2.2.3. Xtext

Xtext es un Framework para el desarrollo de lenguajes de programación y lenguajes específicos de dominio o DSL (Domain Specific Languages). Este potente Framework, cubre todos los aspectos de una infraestructura de lenguaje completo, desde analizadores, enlazadores hasta el compilador o intérprete para su perfecta integración en el IDE de desarrollo Eclipse. Gracias al lenguaje Xtext, se permite definir una gramática para especificar un lenguaje.

Xtext está basado en Xbase, lenguaje de programación parcial implementado en Xtext y destinado a ser incorporado y ampliado en otros lenguajes de programación y lenguajes específicos de dominio (DSL).

Posee un potente marco común de trabajo en el que se aloja el conjunto común de sus librerías. Además ofrece una serie de características como: coloreado de sintaxis, autocompletado, validación rápida de la sintaxis, integración con el lenguaje Java y con otras herramientas de Eclipse; que facilitan la programación y diseño.

Figura 10. Xtext

2.2.4. Java

Java es un lenguaje de programación de propósito general, orientado a objetos y basado en clases. Su sintaxis deriva en gran medida de C y C++, pero tiene menos utilidades de bajo nivel que cualquiera de ellos. Las aplicaciones desarrolladas con Java son compiladas para poder ejecutarse en cualquier máquina virtual Java (JVM).

Este lenguaje se creó con diferentes objetivos principales que son: usar el paradigma de la programación orientada a objetos, permitir la ejecución de un mismo programa en múltiples sistemas operativos, incluir soporte para trabajo en red por defecto, ser sencillo de usar y aprovechar características de otros lenguajes orientados a objetos como por ejemplo C++. Una de sus principales características es la independencia de la plataforma que pretende que programas escritos en el lenguaje Java puedan ejecutarse en cualquier tipo de hardware. Este es el significado de ser capaz de escribir un programa una vez y que pueda ejecutarse en cualquier dispositivo, tal como reza el axioma de Java, "write once, run anywhere".

Se considera en esencia un lenguaje orientado a objetos simple, moderno y de propósito general. Como buen lenguaje orientado a objetos, soporta la encapsulación, la herencia y el polimorfismo. En él se presenta incluidos principios de Ingeniería del Software tales como: revisión estricta de los tipos de datos, revisión de los límites de vectores, detección de intentos de usar variables no inicializadas y recolección de basura automática. Sin duda características muy apropiadas y que son de gran utilidad y comodidad. Posee un potente Framework o marco común de trabajo en el que se aloja el conjunto común de sus librerías. Un punto fuerte es que dichas librerías son independientes de la versión del sistema operativo donde se instalen. Así se consigue una fácil transición de código entre distintos lenguajes de programación. Además cuenta con distintos kits de desarrollo de software (SDK) e interesantes marcos de trabajo que facilitan las labores al desarrollador.

Figura 11. Java

2.2.5. Xtend

Xtend es un lenguaje de programación de alto nivel de propósito general, flexible y expresivo para la máquina virtual de Java. Lenguaje especializado en la generación de código a partir de modelos de EMF (Eclipse Modeling Framework). Sintáctica y semánticamente Xtend tiene sus raíces en el lenguaje de programación Java, pero se centra en una sintaxis más concisa y alguna funcionalidad adicional, como la inferencia de tipos, métodos de extensión, y la sobrecarga de operadores. Siendo principalmente un lenguaje orientado a objetos, que también integra características conocidas de la programación funcional. Xtend es compilado a código Java y por lo tanto se integra perfectamente con todas las bibliotecas de Java existentes. Este lenguaje de código abierto se puede compilar y ejecutar independientemente de la plataforma Eclipse.

Figura 12. Xtend

2.2.6. Xpand

Xpand es un lenguaje de programación especializado en la generación de código, basándose en modelos de EMF (Eclipse Modeling Framework). Se le puede considerar como un lenguaje de plantillas estáticamente tipado con características como programación orientada a aspectos, extensiones funcionales o transformación y validación de modelos.

Es utilizado en las transformaciones conocidas como Model to Text (M2T) o Modelo a Texto, las cuáles se centran en la generación de artefactos textuales de modelos. Proporcionar implementaciones del estándar de Eclipse en cuanto a transformaciones modelo a texto o proporcionar una infraestructura común para los lenguajes con los que se trabaja son algunos de los objetivos que pretende aportar.

Figura 13. Xpand

2.2.7. ATL Transformation Language

ATL Transformation Language, también conocido como lenguaje de transformación ATLAS, es un lenguaje usado para la transformación entre modelos (M2M – Model To Model). Centrado en el campo de la Ingeniería dirigida por modelos (MDE – Model Driven Engineering), ATL proporciona un conjunto de herramientas para producir un conjunto de modelos destino a partir de un conjunto de modelos de origen. Es considerado un híbrido entre lenguaje imperativo y declarativo. Su Framework desarrollado sobre la plataforma o IDE de desarrollo Eclipse proporciona una seria de herramientas de desarrollo estándar que tiene como objetivo facilitar el desarrollo de las transformaciones con este lenguaje.

En esencia, un programa de transformación ATL se compone de reglas que definen cómo dos modelos se emparejan y la forma de navegar a través de ellos para así crear, inicializar y adaptar la información procedente del modelo de origen en el modelo de destino.

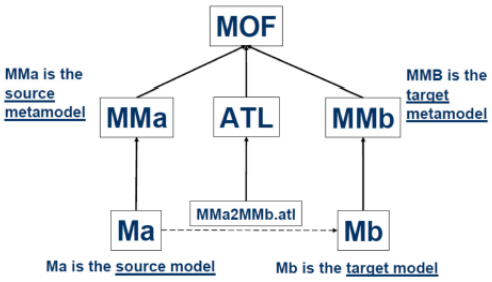


Figura 14. Contexto operacional ATL

Figura 15. ATL Transformation Language

2.2.8. OCL Object Constraint Language

OCL Object Constraint Language, es un lenguaje de especificación usado para la descripción formal de expresiones en modelos UML. Se le puede categorizar como un lenguaje el cual no tiene efectos "colaterales" o de borde, por lo que la verificación de una condición nunca altera a los objetos representados en el modelo. Su principal objetivo consiste en completar las diferentes partes de la notación UML mediante el uso de requerimientos expresados de un modo formal. Su inclusión en el proyecto OCL Eclipse proporciona una herramienta completa para la implementación de la restricción de objetos al lenguaje para modelos basados en EMF. Este núcleo o componente OCL proporciona una API para analizar y evaluar las restricciones OCL y consultas sobre modelos EMF. Además permite integración con metamodelos Ecore y las implementaciones en UML.

Figura 16. OCl Object Constraint Language

2.2.9. Justificación de la elección

Para el presente proyecto fue un requisito establecido el uso del lenguaje Java para su desarrollo. Este lenguaje de programación Java proporciona una fácil migración a los programadores a este nuevo lenguaje, especialmente si se está familiarizado con lenguajes como C o C#. Para este proyecto concreto el desarrollador hubo trabajado con anterioridad con el lenguaje Java y también con C#, por lo que también supuso un agravante a la hora de decidirse por él. Gracias a Java se consigue ahorrar tiempo en la programación ya que cuenta con una librería de clases muy extensa, completa y bien diseñada. Incluso consta de principios de Ingeniería del Software tales como revisión de tipos de datos, revisión de límites en almacenamientos de datos, recolección de basura y otros muchos que son de gran utilidad. Además sus Framework y API´s facilitan la labor al desarrollador. Enlazando Java con el entorno de desarrollo Eclipse se obtiene una grata sensación ya que este entorno divide y diferencia múltiples componentes para el uso el desarrollador, consiguiendo así una mejor claridad a la hora de entender o revisar código. Gracias al proyecto Eclipse Modelling Framework (EMF) de Eclipse se consigue una perfecta integración entre Java y lenguajes como Xtext y Xtend para el correcto desarrollo de un proyecto de software dirigido por modelos. Utilizando la arquitectura MDA (Model Driven Architecture) en el campo de la Ingeniería de Software como piedra angular para el desarrollo de este proyecto, Java, Xtext y Xtend se combinan a la perfección y comparten elementos Ecore y de tecnología incrustado en el Framework EMF. Además para las transformaciones y restricciones en los modelos y diagramas UML, los lenguajes ATL Transformation Language y Object Constraint Language (OCL) encajaban perfectamente con toda la configuración del proyecto y requisitos para el mismo.

En cuanto a las dificultades debidas a no tener experiencia previa con la plataforma y diseño del tipo de aplicaciones necesarias, se solventarán gracias a la fuerte motivación y deseo de aprender con ellas.

2.2.9.1. Herramientas y tecnologías utilizadas

A continuación se describen en detalle las herramientas y tecnologías utilizadas para el desarrollo del proyecto.

Tabla 4. Herramientas y tecnologías utilizadas

IDE	Eclipse versión Indigo Service Release 2
Tecnología	Eclipse Modeling Tools
Framework	Framework de modelado de Eclipse (EMF)
Tecnología	Ingeniería dirigida por modelos (MDE)

IDE (Entorno integrado de desarrollo). Para el desarrollo de la aplicación basada en modelos para la generación de Guías de Test, se ha optado por usar el IDE de desarrollo Eclipse versión Indigo Service Release 2 junto con el paquete Eclipse Modeling Tools y el Framework de modelado de Eclipse o EMF (Eclipse Modeling Framework). Este potente software soporte varios lenguajes de programación y, entre otras, permite a los desarrolladores crear diferentes tipos de aplicaciones.

La selección de ésta herramienta de desarrollo, la ha condicionado las ganas de aprender y profundizar más con ella. Además de por sus muchas aplicaciones y sus múltiples opciones.

Eclipse Modeling Tools. Esta tecnología se conforma de un paquete que contiene un marco de trabajo o Framework y las herramientas necesarias para el tratamiento de modelos. Consta de un editor gráfico Ecore, una utilidad de generación de código y el Framework de modelado EMF. Por lo que se puede decir que se compone de un SDK completo, herramientas de desarrollo y un editor/generador de código fuente.

Framework EMF. El Framework de modelado de Eclipse, o EMF (Eclipse Modeling Framework) es un software o marco de trabajo de modelado y generación de código, utilizado en el Desarrollo de Software dirigido por Modelos o Ingeniería dirigida por Modelos (MDE), para desarrollar herramientas y aplicaciones centradas en modelos de datos y Modelos de Dominio.

Model Driven Engineering (MDE). Se define como Ingeniería dirigida por Modelos, metodología para el desarrollo de software centrada en el diseño y creación de modelos de dominio para cubrir y modelar una necesidad o solución ante un problema para un dominio en particular. Trata de alcanzar el nivel de abstracción en la especificación de programas e incrementar la automatización en su desarrollo. Aporta beneficios como productividad, portabilidad, interoperabilidad, labores de mantenimiento y documentación, calidad aumentada y un alto nivel de reusabilidad.

2.2.9.2. Estándares utilizados

A continuación se describen en detalle los estándares en los que este proyecto se ha basado para su desarrollo

Tabla 5. Estándares utilizados

Estándar de Test	Aiken
Estándar de Test	Gift
Formato estándar	IMS QTI
Formato de representación de Guías	GPC (Guías de Práctica Clínica)

Aiken. El formato Aiken es una forma simple de crear cuestionarios de múltiples opciones usando un formato claro y legible por humanos. (El formato GIFT tiene muchas más opciones y quizá es menos propenso a errores, pero no tiene un aspecto tan simple como el formato Aiken). La pregunta debe estar toda en una sola línea. Cada respuesta debe empezar por un carácter de una sola letra, seguida por un punto '.' o un cierra paréntesis ')', y después un espacio. La línea con la respuesta correcta debe ser la inmediatamente siguiente, comenzando con 'ANSWER: ' (nótese el espacio después de ':') y después la letra apropiada.

https://docs.moodle.org/all/es/Formato_Aiken

```
What is the correct answer to this question?
A. Is it this one?
B. Maybe this answer?
C. Possibly this one?
D. Must be this one!
ANSWER: D
```

Figura 17. Ejemplo de pregunta en Aiken

Gift. El formato Gift permite mediante el uso de un editor de texto crear diversos cuestionarios con preguntas de diferentes tipos como: preguntas de opción múltiple, verdadero-falso, respuesta corta, preguntas numéricas o preguntas de pares de respuesta. El formato ha sido desarrollado en la Comunidad Moodle. https://docs.moodle.org/23/en/GIFT_format

En la siguiente dirección web se especifican todos los completos detalles para cada uno de los tipos.

https://docs.moodle.org/23/en/GIFT_format#Question_format_examples

A continuación se muestran algunos ejemplos de los diferentes tipos de preguntas soportados por este estándar, así como distintos elementos.

```
What two people are entombed in Grant's tomb? {
   ~%-100%No one
   ~%50%Grant
   ~%50%Grant's wife
   ~%-100%Grant's father
}
```

Figura 18. Gift – Multiple Choice Question with multiple answers

```
Who's buried in Grant's tomb?{=Grant =Ulysses S. Grant =Ulysses Grant}
```

Figura 19. Gift – Short Answer Question

```
Match the following countries with their corresponding capitals. {
   =Canada -> Ottawa
   =Italy  -> Rome
   =Japan  -> Tokyo
   =India  -> New Delhi
   }
```

Figura 20. Gift – Matching Question

```
Grant was buried in a tomb in New York City.{T}
```

Figura 21. Gift – True – False Question

```
Moodle costd{~lots of money =nothing ~a small amount} to download from moodle.org.
```

Figura 22. Gift – Missing word Question

```
What is the value of pi (to 3 decimal places)? {#3.14159:0.0005}.
```

Figura 23. Gift – Numerical Question

```
Write a short biography of Dag Hammarskjöld. {}
```

Figura 24. Gift – Essay Question

```
You can use your pencil and paper for these next math questions.
```

Figura 25. Gift – Description

```
// Subheading: Numerical questions below
```

Figura 26. Gift – Line comments

```
::Kanji Origins::
```

Figura 27. Gift – Question names

```
#feedback comment on the wrong answer
```

Figura 28. Gift – Feedback for answers

```
%25%
```

Figura 29. Gift – Percentage answer weights for answers

IMS QTI Question & Test Interoperability. La especificación de preguntas de Test IMS e interoperabilidad, permite el intercambio de ítems, test y resultados, entre diferentes herramientas, bancos de ítems, sistemas de aprendizaje y sistemas de administración de evaluación. Es un estándar muy utilizado en diferentes plataformas de enseñanza, para la evaluación y aprendizaje de las personas. Así como para estudios y análisis relacionado con bancos de ítems. http://www.imsglobal.org/question/

Figura 30. IMS QTI Question & Test Interoperability

Guías de Práctica Clínica GPC. Surge como un proyecto realizado por el grupo de investigación Erabaki de la Universidad del País Vasco (UPV-EHU) en colaboración con BIOEF. Proyecto el cual pretende ayudar al colectivo médico y de la salud en general, a ayudar, diagnosticar y solucionar problemas relacionados con el campo de la medicina. Actualmente existen Guías de Práctica Clínica en formato papel. Se pretende por tanto ofrecer un soporte digital e informatizado para este campo. www.e-guidesmed.ehu.es – guider.ehu.es).

Figura 31. Guías de Práctica Clínica GPC

2.3. Captura de requisitos

En este punto se analizarán los actores que conforman el sistema, así como los casos de uso establecidos para el mismo.

2.3.1. Jerarquía de actores

En esta aplicación, puesto que inicialmente se pretende crear y probar el sistema mediante un prototipo, se distingue un único tipo de usuario, **usuario**.

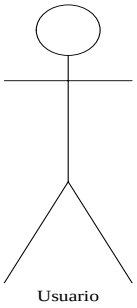


Figura 32. Jerarquía de actores

En un futuro se podría ampliar esta jerarquía de actores incluyendo una nueva figura de actor administrador, encargado de realizar funciones especiales y de nivel superior a las que podría realizar un usuario estándar.

2.3.2. Funcionamiento del sistema

En este apartado se explica el funcionamiento del proyecto general. Se presenta a continuación la arquitectura del mismo con el fin de poder entender mejor su funcionamiento.

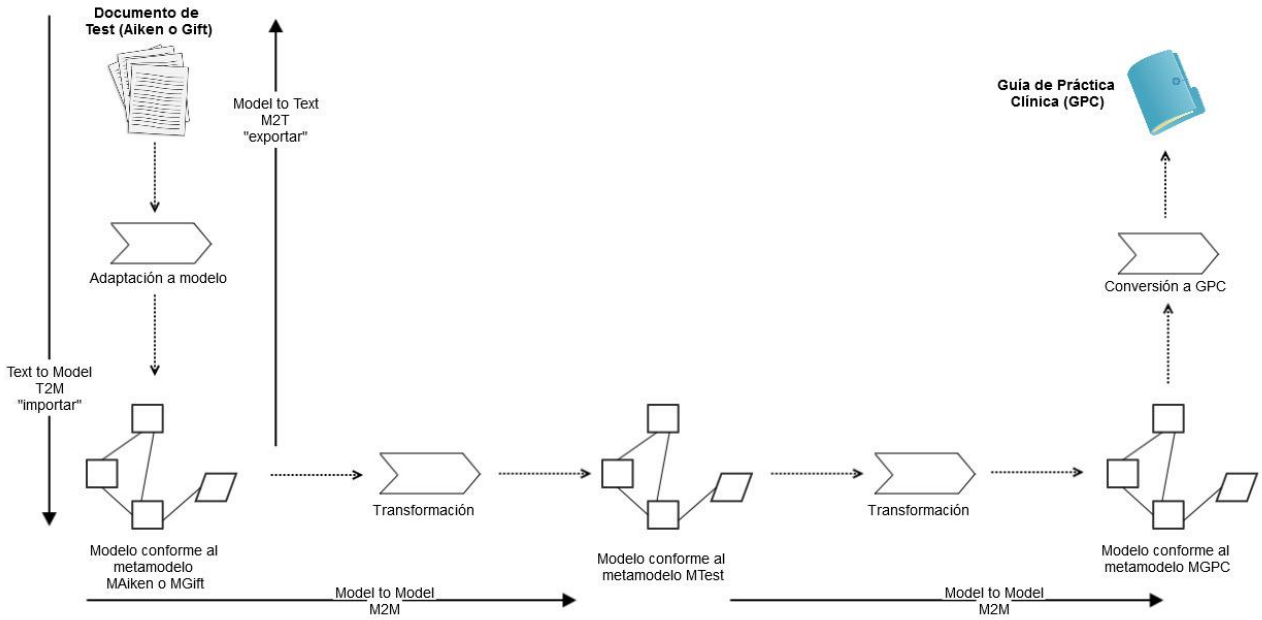


Figura 2. Arquitectura de la Aplicación para la generación de Guías de Test

Un usuario podrá validar un test en cualquiera de los dos formatos estándar Aiken o Gift, para ello dispondrá de un editor de desarrollo para realizar dicha tarea. Es el propio editor quién se encarga de comprobar que los test se corresponden con la gramática especificada. Además, también podrá generar de manera automática un modelo conforme a uno de los metamodelos para los formatos Aiken o Gift. En él editor de desarrollo el usuario podrá crear su propio test conforme a uno de los estándares de test. O bien, si ya disponía de uno, validar si cumple con las restricciones para su estándar. El editor o entorno de desarrollo para el usuario, le indicará los errores y las posibilidades de las que dispone para completar sus test. Una vez finalizado, el sistema generará en la salida un fichero de texto acorde al correspondiente formato que se está validando, de modo que le indicará al usuario que su test ha sido validado y modelado correctamente. Además de poder generar un modelo de test conforme a cualquiera de los dos estándares Aiken o Gift de manera manual (adaptación manual), el usuario podrá generarlo de manera automática. Esto es, con una simple acción, colocando el fichero de origen (de extensión .maiken o .mgift, según el estándar a utilizar) en la carpeta de origen específica para ello, se ejecuta la aplicación y se adapta el test en texto plano (procedente del fichero de texto) en su correspondiente modelo, conforme a su meta modelo, en formato .xmi (XML Metadata Interchange).

Una vez se disponga del modelo Aiken o MGift (de extensión .xmi), se procede a la transformación en un modelo conforme con el estándar para IMS QTI (metamodelo MTest). Para ello se han definido unas reglas de transformación que convierten y adaptan la información procedente de un modelo fuente en un modelo destino. Este proceso se realiza de forma automática gracias a las reglas de transformación definidas con el lenguaje ATL Transformation Language. Obtenido este nuevo modelo, conforme al metamodelo MTest, se procede a una nueva transformación para, partiendo de un modelo conforme a MTest, generar de forma automática un modelo conforme a MGPC. De este modo toda la información la tendremos ya almacenada y distribuida en un modelo acorde al metamodelo estándar utilizado para las Guías de práctica clínica.

Por último un módulo en Eguider es el encargado de realizar la conversión desde un modelo conforme a MGPC a una Guía de Práctica Clínica (GPC).

2.3.3. Diagrama de Casos de uso

En este apartado se recogen los casos de uso de los actores que conforman el sistema.

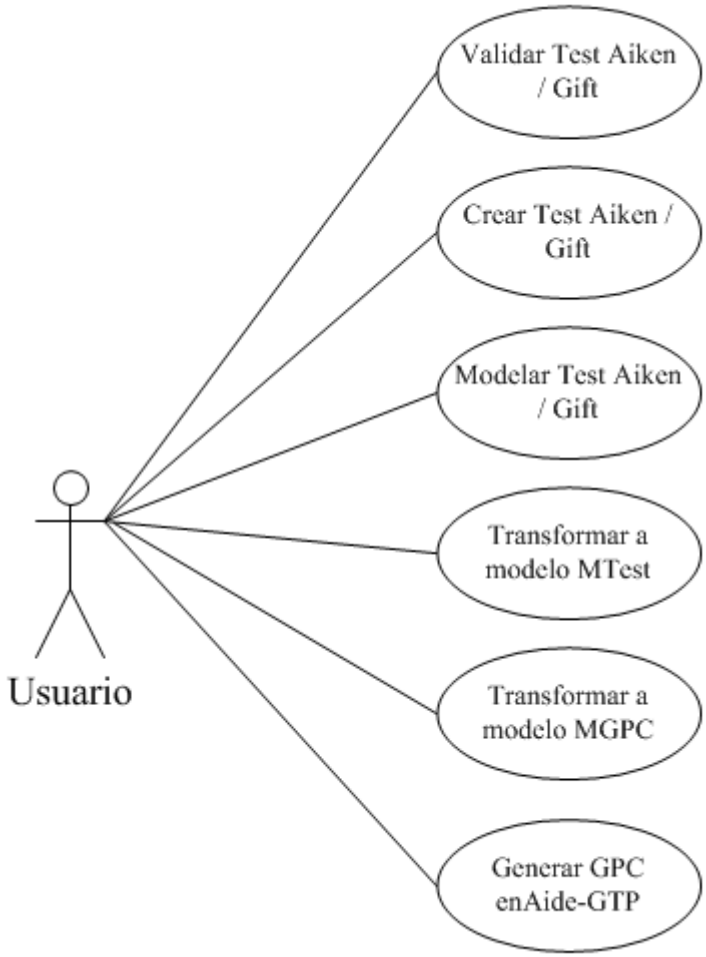
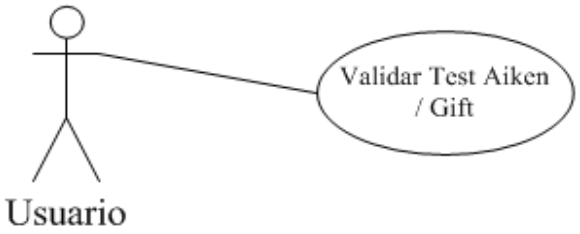


Figura 33. Diagrama de Casos de uso

*****Generar GPC en Aide-GTP no es considerado competencia directa del presente proyecto, pero si forma parte de la visión global del mismo. Es por ello que se ha considerado incluirlo en este diagrama de Casos de Uso.

2.3.4. Casos de Uso extendidos

A continuación se explicarán en detalle cada uno de los casos de uso que conforman el sistema.



Nombre:	Validar Test Aiken / Gift

Descripción:
El usuario a través del editor de desarrollo valida un test conforme a uno de los estándares.

Actores:
Usuario.

Precondiciones:
Ninguna.

Requisitos no funcionales:
Ninguno.

Flujo de eventos:
1.- En el editor de desarrollo el usuario valida un test contra uno de los estándares de Test Aiken o Gift.
2.- El editor de desarrollo le indicará los errores en caso de que hubiera.
3.- Si el test se corresponde con el estándar, el editor de desarrollo gracias a la gramática y de manera automática validará el test.

Poscondiciones:
Ninguna.

Captura de pantalla:

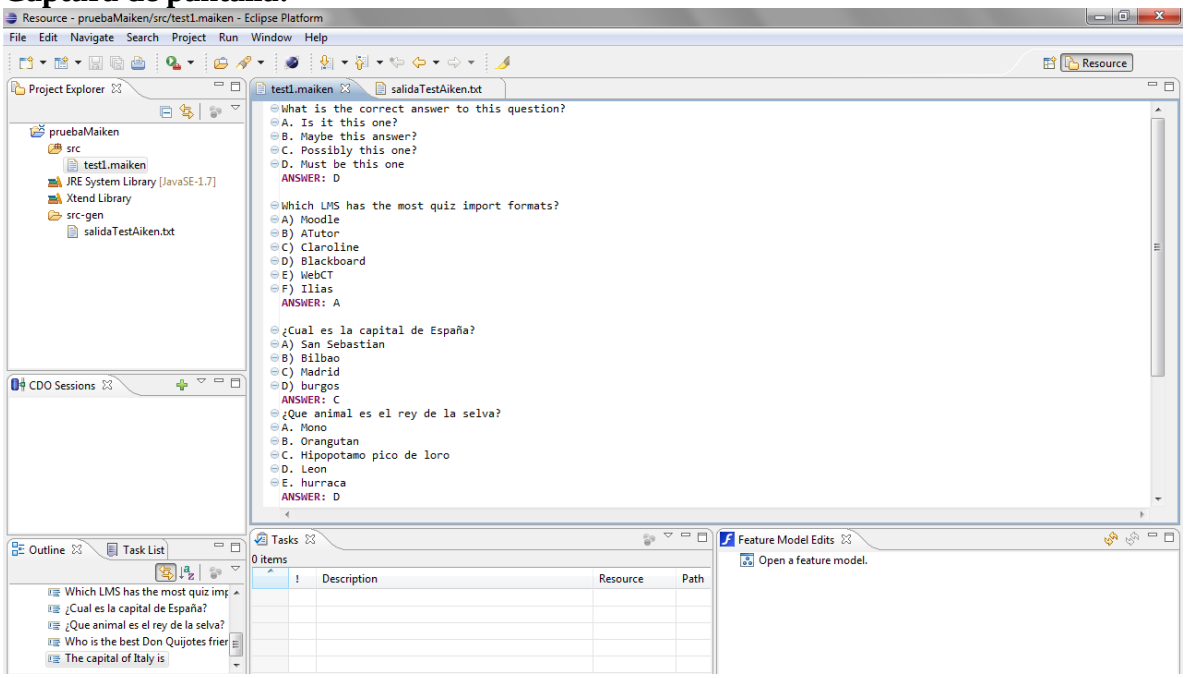


Figura 34. Validar Test Aiken / Gift

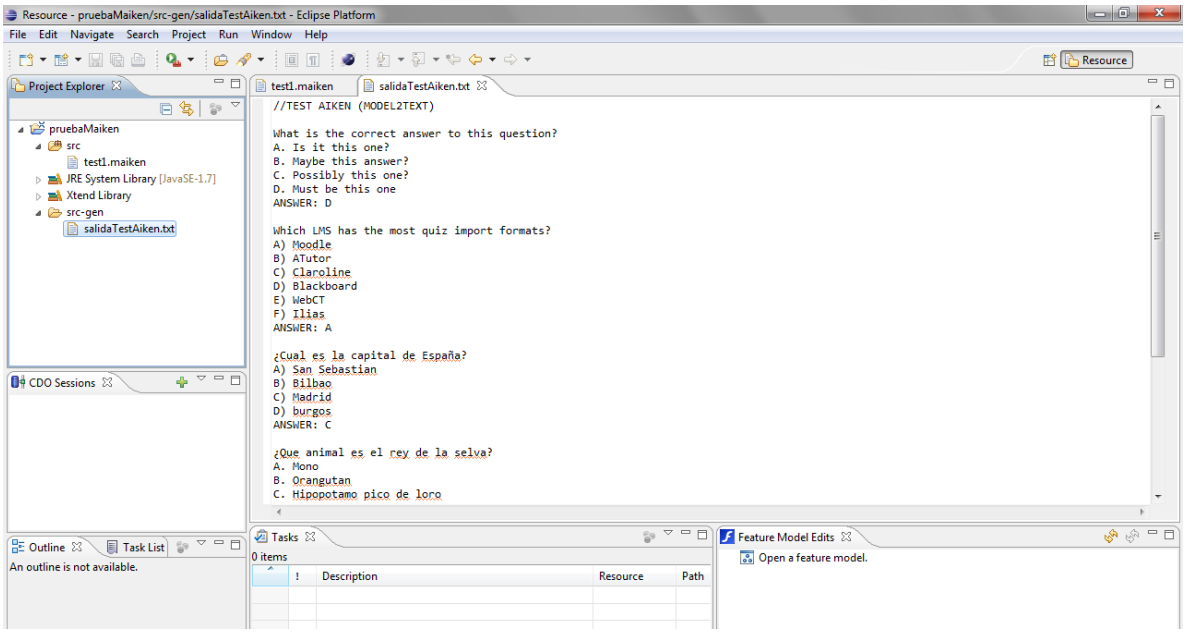
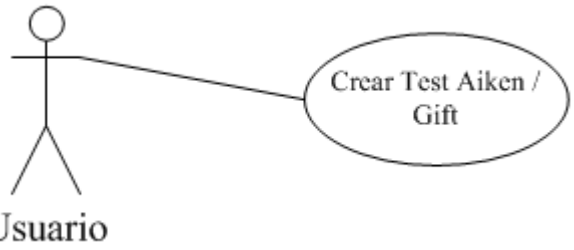


Figura 35. Validar Test Aiken / Gift



Nombre:	Crear Test Aiken / Gift

Descripción:
El usuario a través del editor de desarrollo crea un test conforme a uno de los estándares.

Actores:
Usuario.

Precondiciones:
Ninguna.

Requisitos no funcionales:
Ninguno.

Flujo de eventos:
1.- En el editor de desarrollo el usuario crea un test conforme a uno de los estándares de Test Aiken o Gift.
2.- El editor de desarrollo le guiará y le indicará los errores pertinentes.
3.- Si el test se corresponde con el estándar, el editor de desarrollo gracias a la gramática y de manera automática validará el test y creará un fichero de extensión txt acorde al formato establecido Aiken o Gift.

Poscondiciones:
Ninguna.

Captura de pantalla:

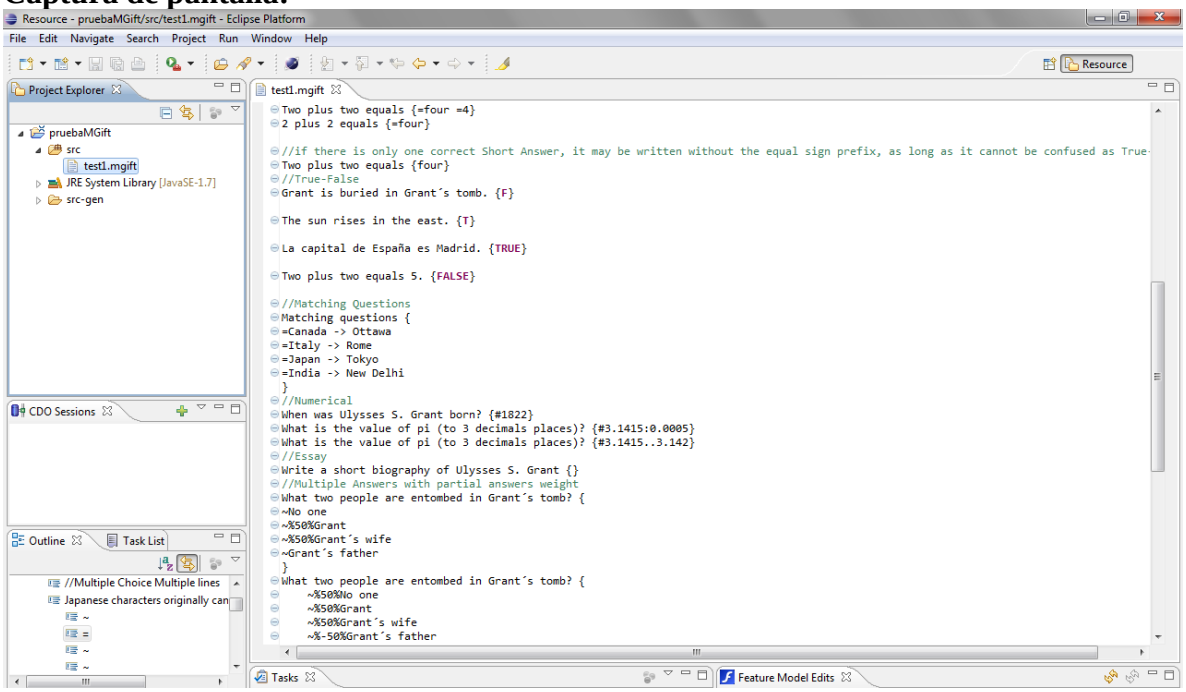


Figura 36. Crear Test Aiken / Gift

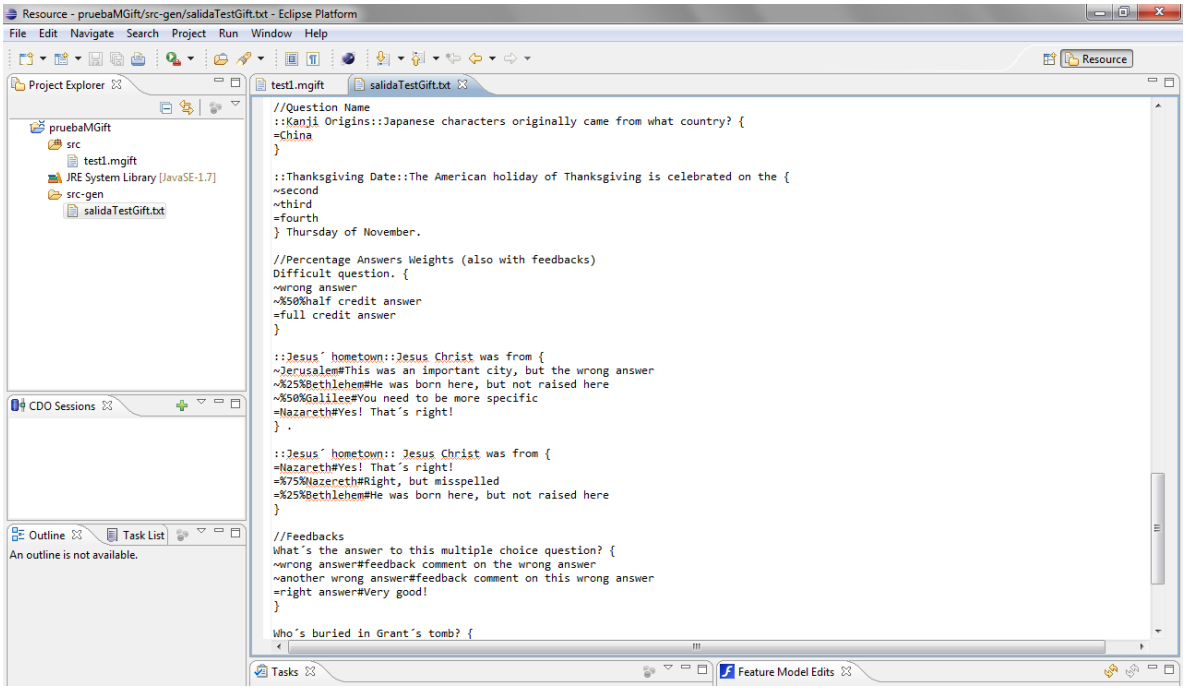
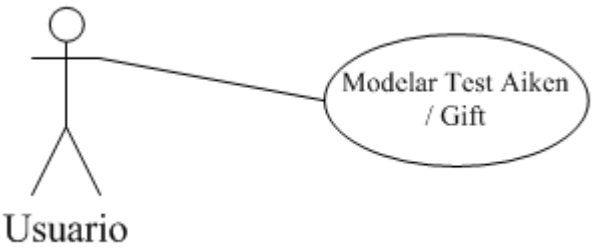


Figura 37. Crear Test Aiken / Gift



Nombre:	Modelar Test Aiken / Gift

Descripción:
El usuario a través del editor de desarrollo modela de manera automática un test conforme a MAiken o a MGift.

Actores:
Usuario.

Precondiciones:
Ninguna.

Requisitos no funcionales:
Ninguno.

Flujo de eventos:
1.- En el editor de desarrollo el usuario copia el test de origen que quiere modelar en la carpeta inputTest y lanza la aplicación.
2.- El programa generará en la carpeta output el test ya modelado conforme a su metamodelo MAiken o MGift y le mostrará al usuario el mensaje "Generado modelo xmi conforme a estándar. Salida en carpeta output."

Poscondiciones:
Ninguna.

Captura de pantalla:

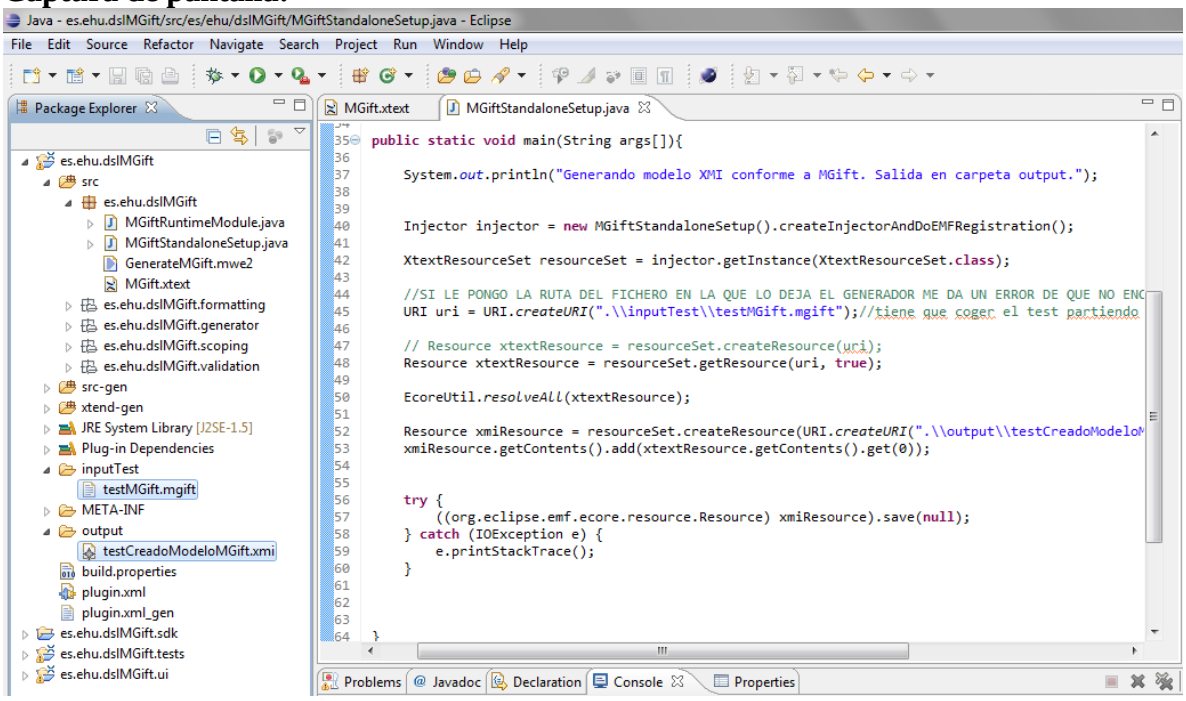
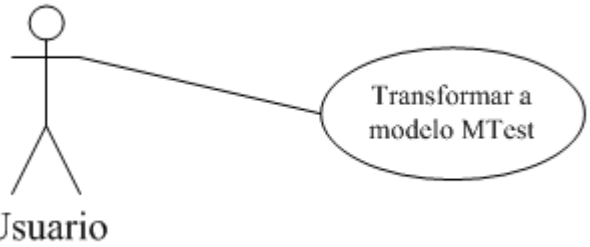


Figura 38. Modelar Test Aiken / Gift



Nombre:	Transformar a modelo MTest

Descripción:
El usuario a través del editor de desarrollo transforma de forma automática un modelo conforme al metamodelo MAiken o MGift en un modelo conforme a MTest.

Actores:
Usuario.

Precondiciones:
Se ha de configurar el panel "Run Configurations" para la regla de transformación.

Requisitos no funcionales:
Ninguno.

Flujo de eventos:
1.- En el editor de desarrollo el usuario copia el modelo del test de origen, conforme al metamodelo MAiken o MGift, en la carpeta model.
2.- Configura la regla de transformación correspondiente ("Run Configurations") indicando cuál es el modelo origen y como se llamará el modelo destino.
3.- El usuario lanza la transformación y se genera de forma automática en la carpeta output el modelo generado conforme al metamodelo MTest.

Poscondiciones:
Ninguna.

Captura de pantalla:

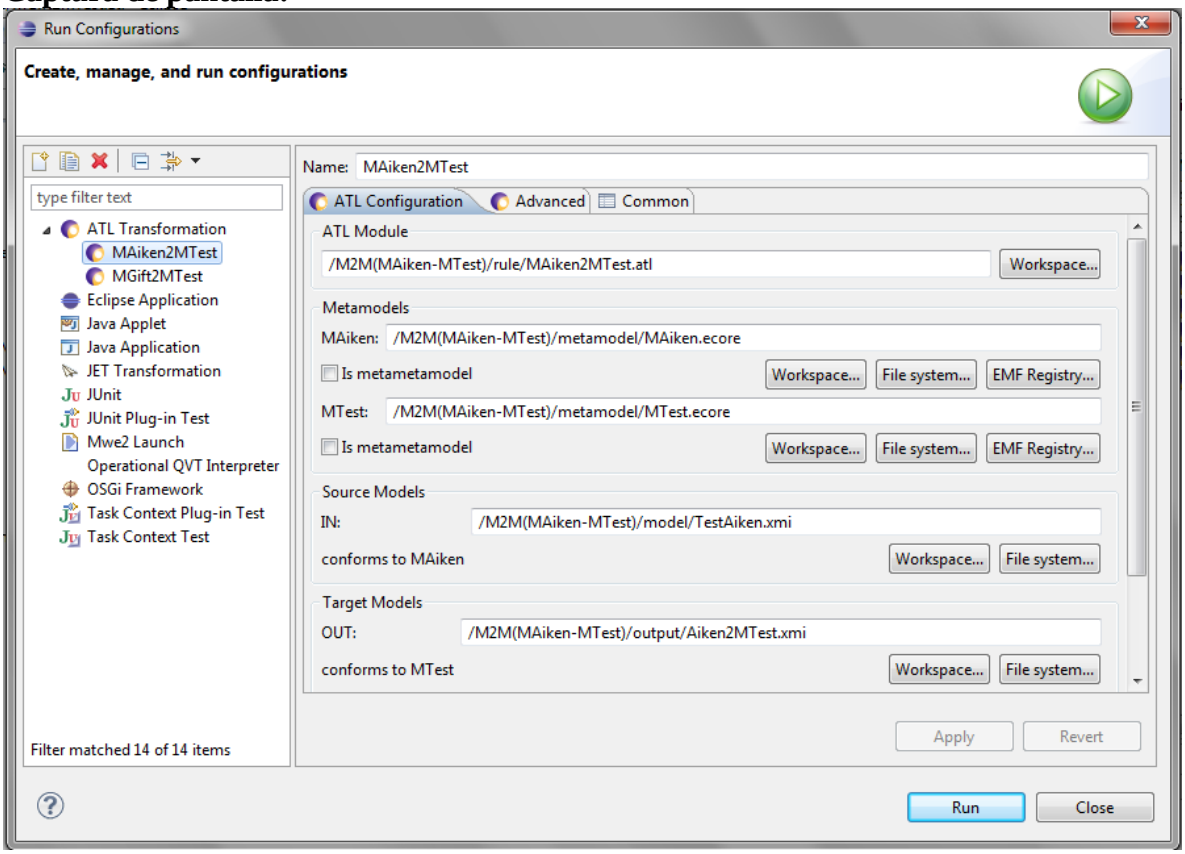


Figura 39. Transformar a modelo MTest

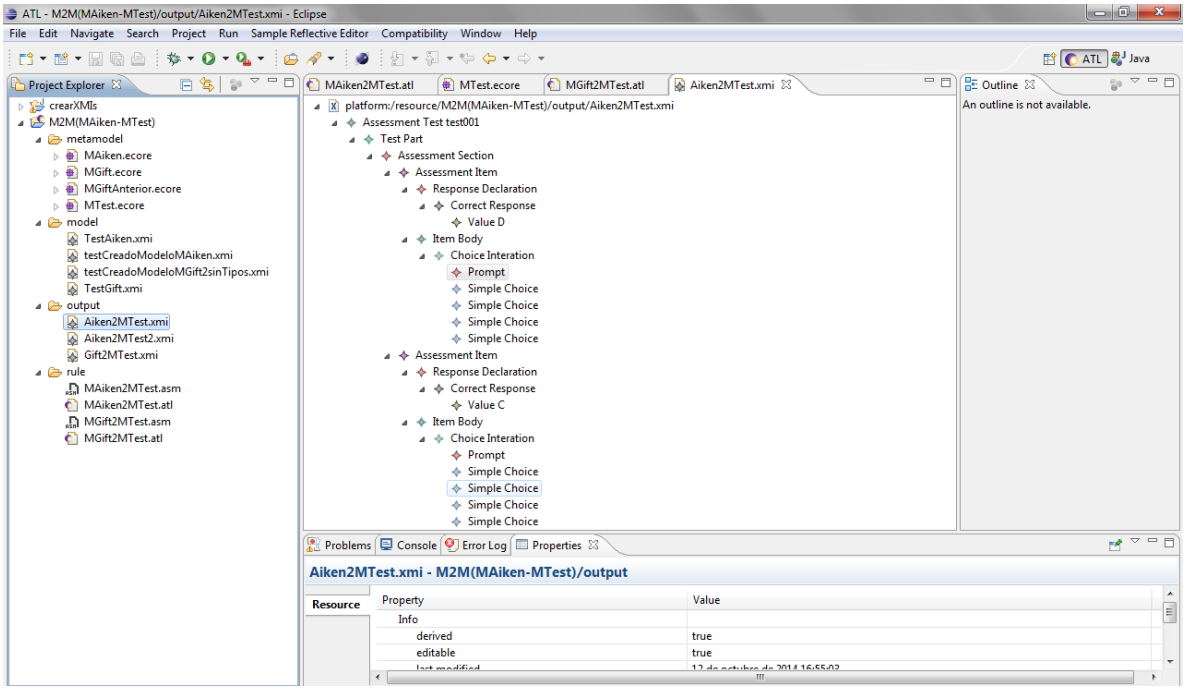
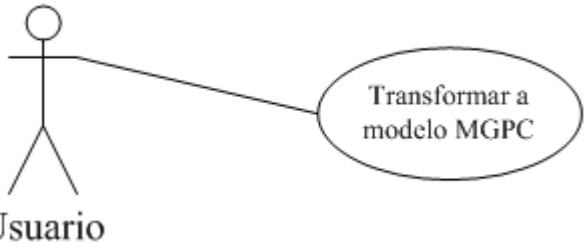


Figura 40. Transformar a modelo MTest



Nombre:	Transformar a modelo MGPC

Descripción:
El usuario a través del editor de desarrollo transforma de forma automática un modelo conforme al metamodelo MTest en un modelo conforme a MGPC.

Actores:
Usuario.

Precondiciones:
Se ha de configurar el panel "Run Configurations" para la regla de transformación.

Requisitos no funcionales:
Ninguno.

Flujo de eventos:
1.- En el editor de desarrollo el usuario copia el modelo del test de origen, conforme al metamodelo MTest, en la carpeta model.
2.- Configura la regla de transformación correspondiente ("Run Configurations") indicando cuál es el modelo origen y como se llamará el modelo destino.
3.- El usuario lanza la transformación y se genera de forma automática en la carpeta output el modelo generado conforme al metamodelo MGPC.

Poscondiciones:
Ninguna.

Captura de pantalla:

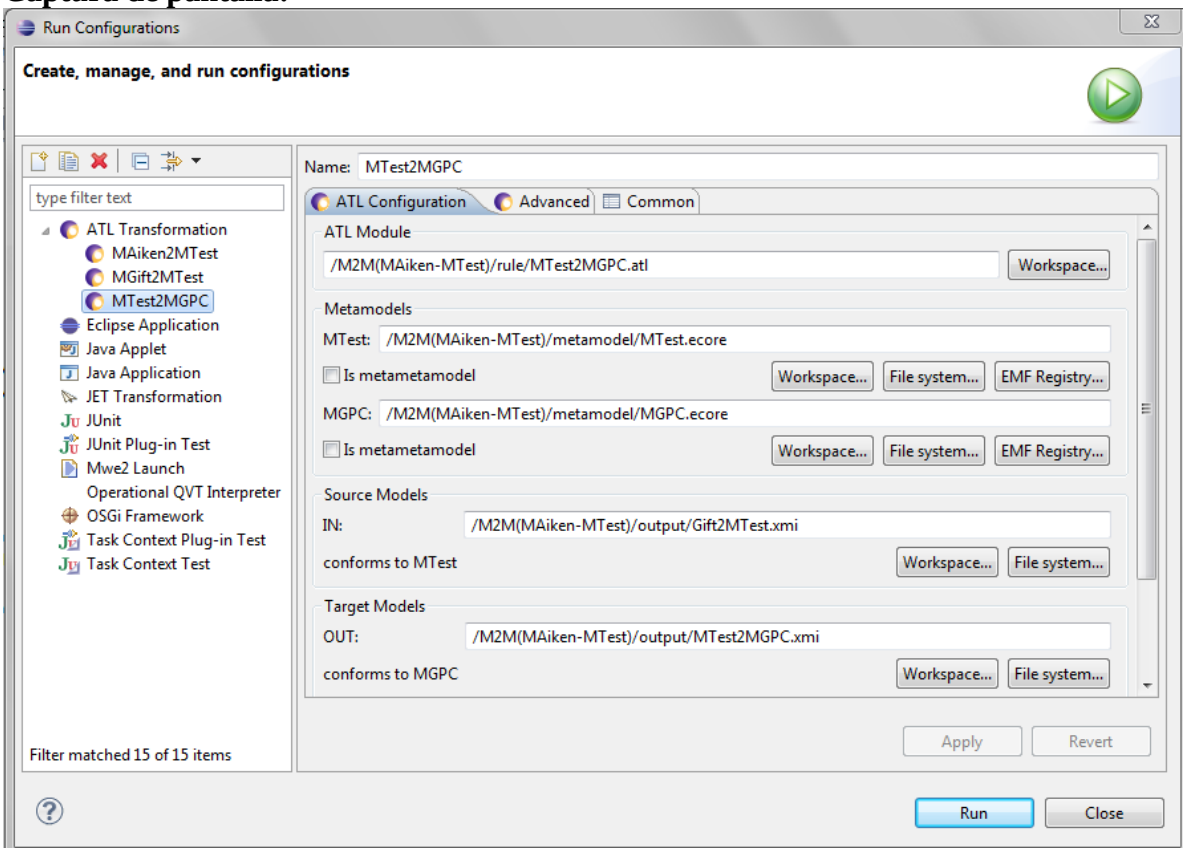


Figura 41. Transformar a modelo MGPC

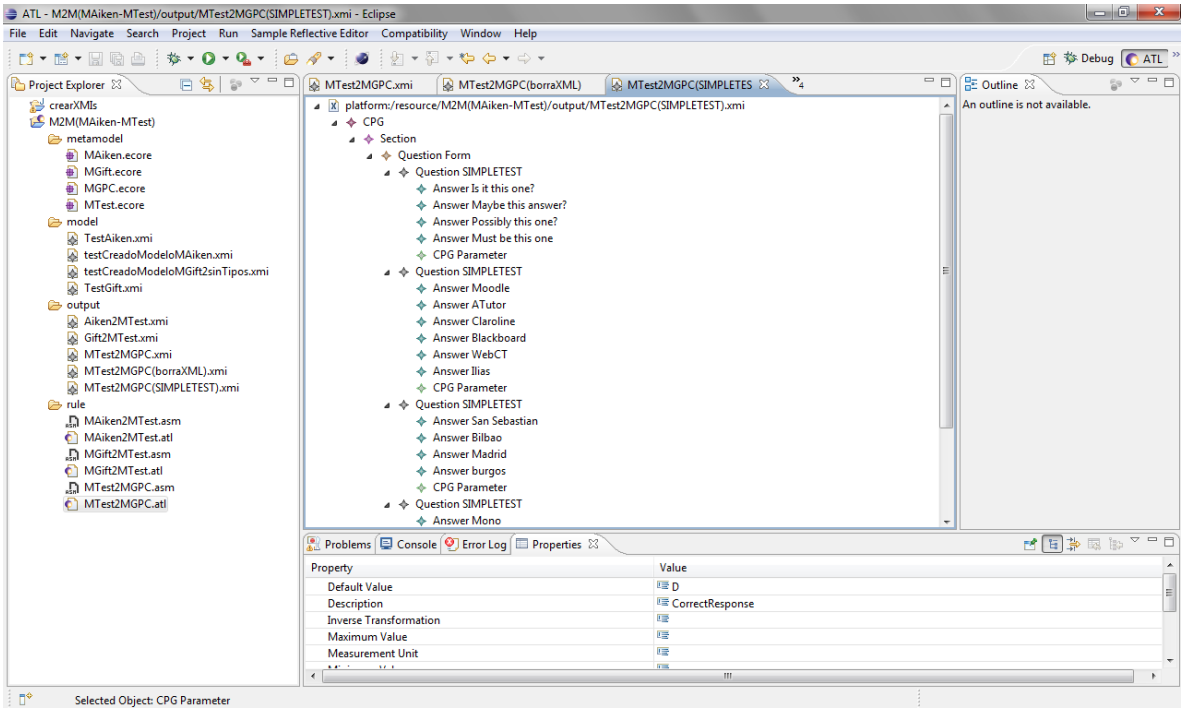


Figura 42. Transformar a modelo MGPC

2.3.5. Requisitos no funcionales

Los requisitos no funcionales sirven para conocer las características del sistema pero sin centrarse en rasgos específicos como descripciones de información generada o funciones realizadas.

Requisitos del sistema para Eclipse versión Indigo Service Release 2:

Procesador de al menos 1,6 GHz tanto para 32bits(x86) como para 64 bits(x64).
1024MB de RAM.

- Windows:
  Microsoft Windows Vista (x86 y x64).
  Microsoft Windows 7 (x86 y x64).
  Microsoft Windows Server (x86 y x64) versión 2003 y versión 2008.
  600mb de espacio disponible en el disco duro.

- Mac OS:
  Mac basado en Intel que ejecuta Mac OS X 10.7.3 (Lion) o posterior.
  Privilegios de administrador para la instalación.
  Explorador de 64 bits.

- Linux:
  Oracle Linux 5.5+
  Oracle Linux 6.x (32 bits)*, 6.x (64 bits)*
  Red Hat Enterprise Linux 5.5+, 6.x (32 bits)*, 6.x (64 bits)*
  Ubuntu Linux 10.04 y superior
  SUSE Linux Enterprise Server 10 SP2, 11.x

Escalabilidad: El código de la aplicación tiene que ser claro y preciso para que cualquier otro programador que necesite modificarlo sea capaz de entender la estructura del programa en el menor tiempo posible, es decir, el sistema debe ser fácilmente ampliable o configurable.

2.4. Análisis y Diseño

En el siguiente apartado se explicará el diseño de los metamodelos necesarios en el sistema así como el funcionamiento de las gramáticas y los editores para la creación y la validación de los test y las transformaciones necesarias para pasar de un modelo a otro.

2.4.1. Metamodelos

Partiendo del análisis y diseño de la aplicación, para el correcto modelado de la información, se presenta seguidamente los metamodelos empleados en el presente proyecto, adjuntando el diseño y la explicación detallada de los mismos. Para el desarrollo de estos metamodelos se ha utilizado el Framework de desarrollo Eclipse Modelling Framework (EMF).

2.4.1.1. Metamodelo MAiken

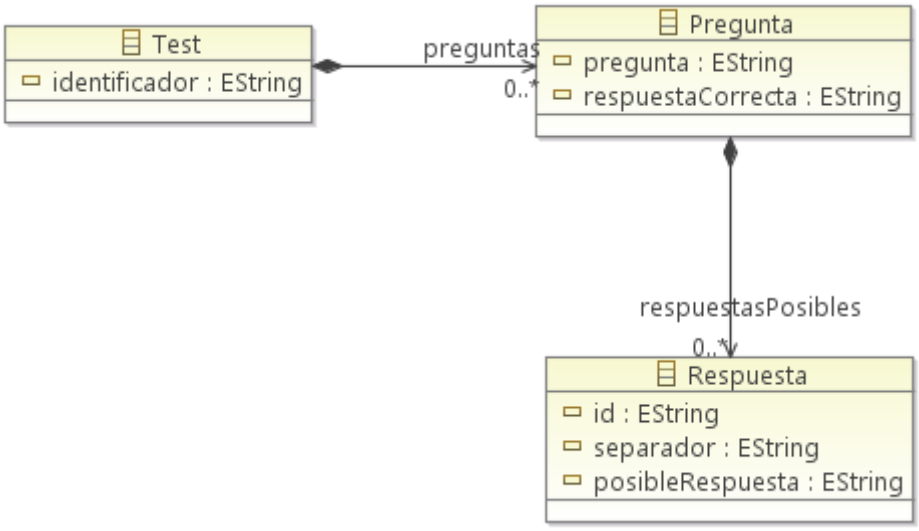


Figura 43. Metamodelo MAiken

Diseñado para poder modelar y gestionar la información referente a un test en formato Aiken. Este metamodelo está soportado por su gramática asociada o DSL (Domain Specific Language), la cual valida los test introducidos en su editor creado para tal fin. Y además guía al usuario en la realización de uno nuevo. A continuación se describen las clases del metamodelo así como sus atributos y relaciones.

La clase Test es la raíz de todo test en formato Aiken. Es el elemento del que parten los demás. Aunque la entrada estándar de un test en Aiken no contemple un valor identificador para un test, se le ha añadido este atributo con el fin de que el usuario pueda añadir a posteriori este valor.

Un Test está formado de cero o varias preguntas. Cada una de estas preguntas almacena la información referente a la propia pregunta y su respuesta correcta en forma de identificador (A, B, C…).

A su vez una Pregunta se compone de posibles respuestas a esa Pregunta. En la clase Respuesta se almacena el valor referente al identificador de la respuesta (A, B, C…), un separador, que será un '.' O un ')' según el estándar Aiken, y el valor de la posible respuesta, esto es la propia posible respuesta a la pregunta.

2.4.1.2. Metamodelo MGift

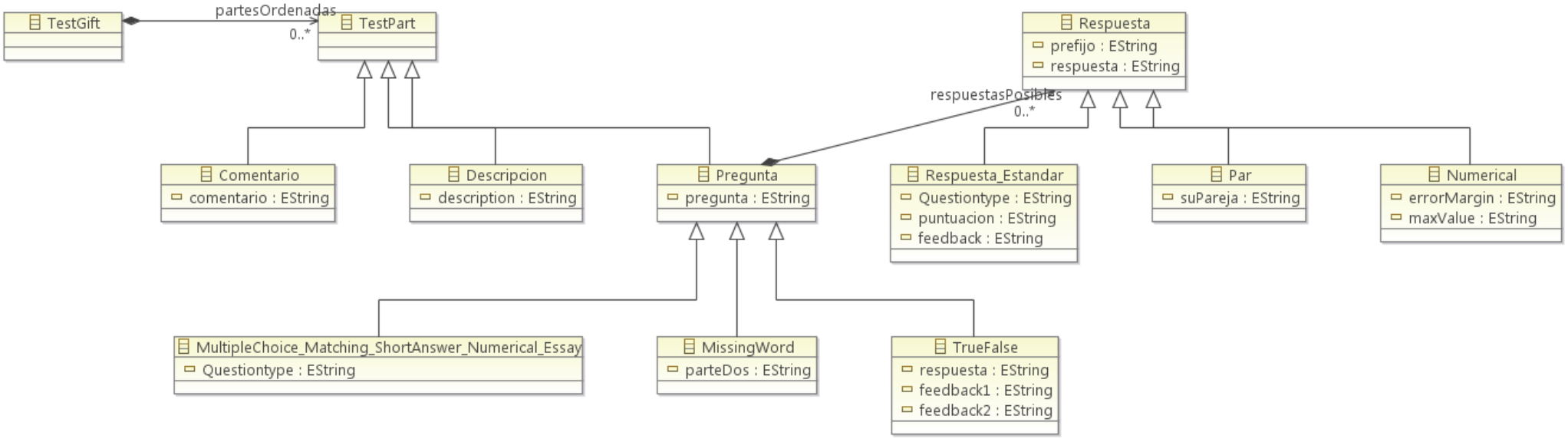


Figura 44. Metamodelo MGift

Este metamodelo ha sido diseñado para poder modelar y gestionar la información referente a un test en formato Gift. Este formato es bastante más complejo que el formato Aiken explicado anteriormente. Al contrario que en Aiken, el estándar Gift soporta diferentes tipos de preguntas así como comentarios para las mismas y puntuaciones. Tanto este estándar de Test como el estándar Aiken son utilizados en la plataforma Moodle como Test Informatizados para el aprendizaje y la enseñanza, así como para encuestas y estudios. Al igual que ocurre con el metamodelo MAiken, este metamodelo MGift está soportado por su gramática o Lenguaje Específico de Dominio (DSL), el cual se encarga de validar los test acordes al estándar Gift. A continuación se describen las clases del metamodelo así como sus atributos y relaciones.

Todo modelo acorde al metamodelo MGift tendrá un elemento raíz de nombre TestGift. De esta clase colgarán las diferentes partes de un test (ordenadas). Estas partes se componen de comentarios, descripciones o preguntas. Un comentario va precedido por // y puede contener información que el usuario considere interesante. Una descripción normalmente suele ir asociada a alguna Pregunta, aunque no necesariamente. Generalmente proporciona información acerca de algún elemento del test.

Para modelar la información referente a una pregunta se distinguen tres clases: MultipleChoice_Matching_ShortAnswer_Numerical_Essay, MissingWord y TrueFalse. Las tres heredan de la clase Pregunta, quien almacena la pregunta en sí. Para la clase MultipleChoice_Matching_ShortAnswer_Numerical_Essay se dispone de un atributo que identifica el tipo de pregunta que almacena (Questiontype): MC (Multiple Choice), NU (Numerical), MA (Matching), SA (Short Answer) o ES (Essay).

Además de esta clase, la clase MissingWord almacena preguntas que se componen de dos partes, o bien, la respuesta a dicha pregunta se encuentra incrustada en medio de una frase. Por ejemplo: *La capital de _____ es Madrid. Posibles respuestas: Francia, España, Portugal, Italia.* La primera parte de la pregunta quedaría almacenada en el atributo pregunta de la clase MissngWord (por herencia de Pregunta), mientras que la segunda parte quedaría almacenada en el atributo parteDos de la misma clase.

Por último para las preguntas, la clase TrueFalse no solo almacena la pregunta sino que también contiene su respuesta y sus posibles feedback para la respuesta correcta o la errónea. Las preguntas se componen de ninguna (como es el caso de TrueFalse) o varias respuestas.

Para cada Respuesta se almacena su propia respuesta y el prefijo (‘=’ ‘~’) que nos da información acerca de una respuesta, como por ejemplo, si es correcta o no. A excepción de las respuestas a las preguntas del Tipo Numerical o Matching, estas se modelan gracias a la clase Respuesta_Estandar. Esta puede almacenar además el feedback para una respuesta, así como su puntuación, en el caso de preguntas que admiten respuestas con puntuación. El tipo de respuesta Numerical es un poco especial, pues ha de permitir guardar un margen de error en caso de que lo hubiera o un valor máximo de tolerancia para dar por válida una respuesta. El tipo Par permite almacenar pares de respuestas que unen sus valores para responder a una pregunta concreta.

2.4.1.3. Metamodelo MTest

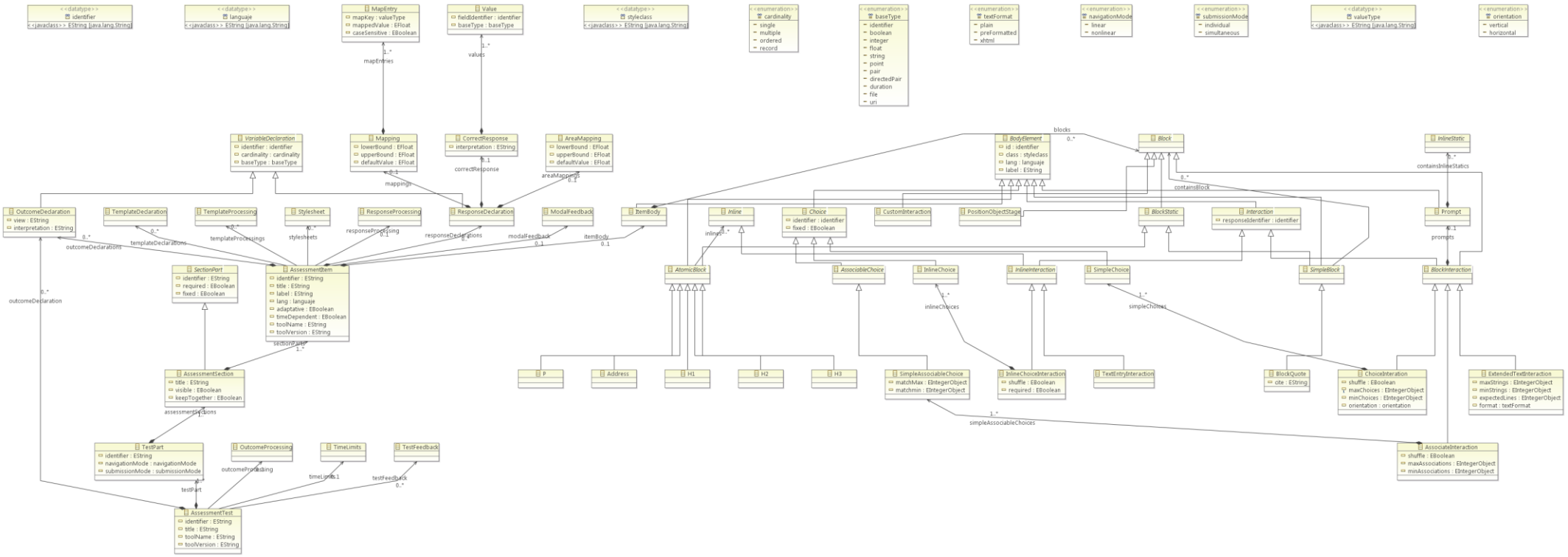


Figura 45. Metamodelo MTest

El metamodelo MTest pretende poder integrar todos los elementos que definen un test. Se trata de un metamodelo independiente de la plataforma o PIM (Platform Independent Model). Basándose en el estándar de test IMS QTI Question & Test Interoperability (estándar utilizado en plataformas de enseñanza, evaluación y aprendizaje, estudios y análisis, etc.), MTest cumple con sus restricciones y normas. El metamodelo o ecore MTest arriba ilustrado no muestra todos los elementos disponibles definidos por el formato estándar de IMS QTI, ya que hay elementos que para la realización del presente proyecto no son necesarios, por lo que se ha adaptado y se han incluido los elementos del formato estándar necesarios para cubrir las necesidades de la aplicación. Además, también han sido integrados elementos de interesante consideración, para la visualización en el conjunto del metamodelo.

La clase AssesmentTest, será el elemento raíz de los modelos conformes a este metamodelo. En ella se almacenará información referente al identificador de Test, el título y otros datos de interés. Un AssesmentTest se compone de diferentes TestPart, que serán las distintas partes de las cuáles un Test dispone. A su vez cada parte se compone de secciones (AssesmentSection), y estas de AssesmentItem. Este elemento será el considerado como descriptor principal o elemento principal para albergar la información referente a un elemento del test. En él o de él se almacenará o saldrá el enunciado de una pregunta, sus posibles respuestas, la descripción de un comentario, etc. Por ejemplo, para un comentario o una descripción, se guardará su tipo en el atributo identifier de la clase AssesmentItem, mientras que su enunciado asociado a esa pregunta se almacena en el atributo label. Para el caso de las preguntas,en el atributo identifier se guarda el tipo de pregunta a la que corresponde: SIMPLETEST (Multiple Choice Aiken estándar), MC (Multiple Choice), MW (Missing Word), NU (Numerical), MA (Matching), SA (Short Answer), TF (True False) o ES (Essay). En función del tipo al que corresponda se irá navegando por el metamodelo para almacenar la información de una forma u otra. En el siguiente enlace se pueden visualizar ejemplos de las diferentes preguntas y su representación para el formato estándar IMS QTI. http://www.imsglobal.org/question/qtiv2p1/imsqti_implv2p1.html

Para el caso de una pregunta de tipo Multiple Choice se modela de la siguiente manera en un modelo conforme a MTest: Dentro del elemento AssesmentItem se encuentra un elemento ItemBody que a su vez contiene al elemento de tipo CoiceInteraction. De él cuelgan los elementos Prompt (donde se almacena la pregunta) y SimpleChoice, donde se encuentran almacenadas las diferentes respuestas para la pregunta con sus correspondientes identificadores. La respuesta correcta para esa pregunta se encuentra guardada en ResponseDeclaration, elemento CorrecResponse, elemento Value, en forma de identificador, el cual se corresponde con el identificador de una de las posibles respuestas dadas. Ver figura 46.

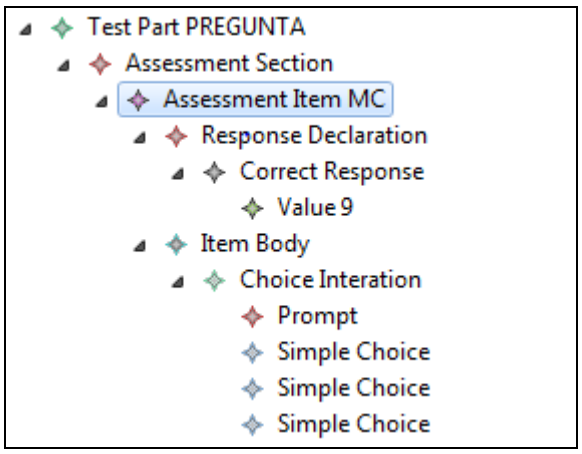


Figura 46. Pregunta Multiple Choice en modelo conforme a MTest

A continuación se puede ver como es representado en formato XML este tipo de pregunta según el estándar IMS QTI.
http://www.imsglobal.org/question/qtiv2p1/examples/items/choice.xml

```
<!--
    This example adapted from the PET Handbook, copyright University of Cambridge ESOL Examinations
-->
<assessmentItem xsi:schemaLocation="http://www.imsglobal.org/xsd/imsqti_v2p1 http://www.imsglobal.org/xsd/qti/qtiv2p1/imsqti_v2p1.xsd" identifier="choice" title="Unattended Luggage"
adaptive="false" timeDependent="false">
  <responseDeclaration identifier="RESPONSE" cardinality="single" baseType="identifier">
    <correctResponse>
      <value>ChoiceA</value>
    </correctResponse>
  </responseDeclaration>
  <outcomeDeclaration identifier="SCORE" cardinality="single" baseType="float">
    <defaultValue>
      <value>0</value>
    </defaultValue>
  </outcomeDeclaration>
  <itemBody>
    <p>Look at the text in the picture.</p>
    <p>
      <img src="images/sign.png" alt="NEVER LEAVE LUGGAGE UNATTENDED"/>
    </p>
    <choiceInteraction responseIdentifier="RESPONSE" shuffle="false" maxChoices="1">
      <prompt>What does it say?</prompt>
      <simpleChoice identifier="ChoiceA">You must stay with your luggage at all times.</simpleChoice>
      <simpleChoice identifier="ChoiceB">Do not let someone else look after your luggage.</simpleChoice>
      <simpleChoice identifier="ChoiceC">Remember your luggage when you leave.</simpleChoice>
    </choiceInteraction>
  </itemBody>
  <responseProcessing template="http://www.imsglobal.org/question/qti_v2p1/rptemplates/match_correct"/>
</assessmentItem>
```

Figura 47. XML Multiple Choice para IMS QTI

En la figura 48 se presenta una captura de pantalla correspondiente a la visualización de una pregunta de tipo Multiple Choice.

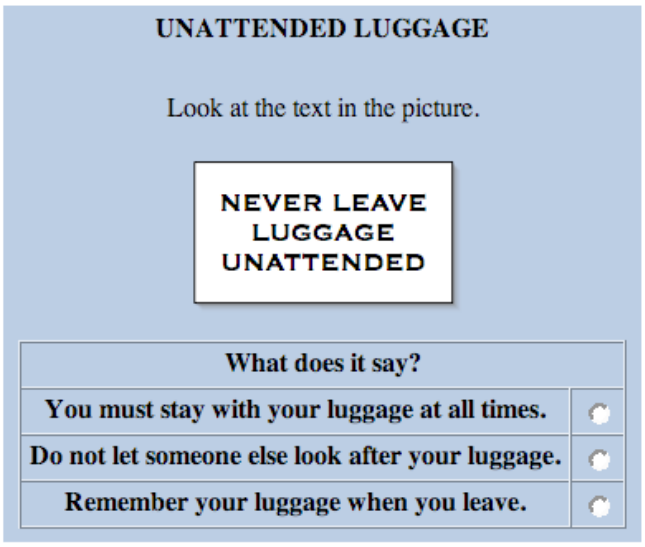


Figura 48. Ejemplo de pregunta Multiple Choice

Para el caso de una pregunta de tipo Missing Word se modela de la siguiente forma en MTest: Partiendo del elemento AssesmentItem se encuentra el elemento ItemBody, y en él se encuentra el elemento BlockQuote, que será quién contendrá en el primero de los elementos de tipo P el enunciado correspondiente a la primera parte de la pregunta. Después se encuentra el elemento InlineChoiceInteraction que dispondrá de las posibles respuestas almacenadas en InliceChoice, así como un elemento que las identifique, un identificador. Finalizadas estas estará un nuevo elemento P que contedrá la siguiente parte o parte final de la pregunta. La respuesta correcta para esa pregunta se encuentra guardada en ResponseDeclaration, elemento CorrecResponse, elemento Value, en forma de identificador el cual se corresponde con el identificador de una de las posibles respuestas dadas.

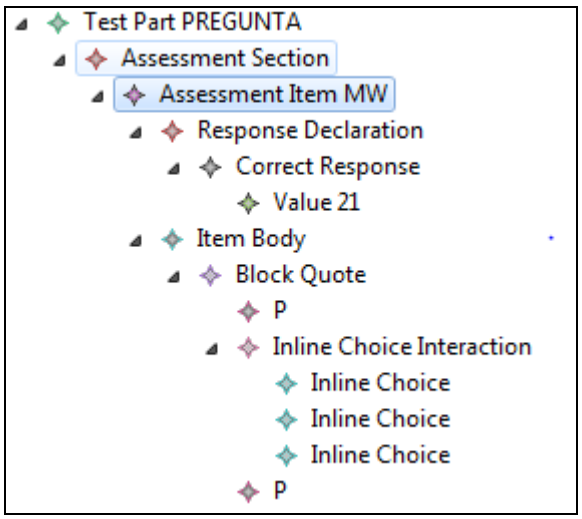


Figura 49. Pregunta Missing Word en modelo conforme a MTest

A continuación se puede ver como es representado en formato XML este tipo de pregunta según el estándar IMS QTI.
http://www.imsglobal.org/question/qtiv2p1/examples/items/inline_choice.xml

```
- <assessmentItem xsi:schemaLocation="http://www.imsglobal.org/xsd/imsqti_v2p1 http://www.imsglobal.org/xsd/qti/qtiv2p1/imsqti_v2p1.xsd" identifier="inlineChoice" title="Richard III
  (Take 2)" adaptive="false" timeDependent="false">
  - <responseDeclaration identifier="RESPONSE" cardinality="single" baseType="identifier">
    - <correctResponse>
        <value>Y</value>
      </correctResponse>
    </responseDeclaration>
    <outcomeDeclaration identifier="SCORE" cardinality="single" baseType="float"/>
  - <itemBody>
    - <p>
        Identify the missing word in this famous quote from Shakespeare's Richard III.
      </p>
    - <blockquote>
      - <p>
          Now is the winter of our discontent
          <br/>
          Made glorious summer by this sun of
        - <inlineChoiceInteraction responseIdentifier="RESPONSE" shuffle="false">
            <inlineChoice identifier="G">Gloucester</inlineChoice>
            <inlineChoice identifier="L">Lancaster</inlineChoice>
            <inlineChoice identifier="Y">York</inlineChoice>
          </inlineChoiceInteraction>
          ;
          <br/>
          And all the clouds that lour'd upon our house
          <br/>
          In the deep bosom of the ocean buried.
        </p>
      </blockquote>
    </itemBody>
    <responseProcessing template="http://www.imsglobal.org/question/qti_v2p1/rptemplates/match_correct"/>
  </assessmentItem>
```

Figura 50. XML Missing Word para IMS QTI

En la figura 51 se presenta una captura de pantalla correspondiente a la visualización de una pregunta de tipo Missing Word.

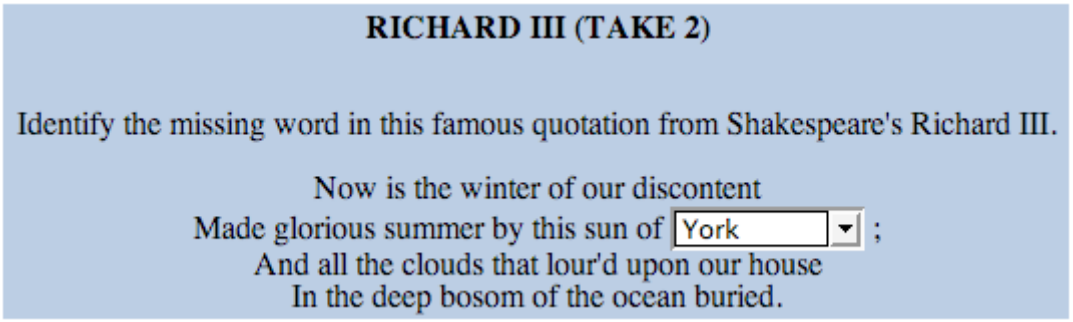


Figura 51. Ejemplo de pregunta Missing Word

En el caso de una pregunta de tipo Numerical para proceder a su modelado se realiza de la siguiente forma: El enunciado correspondiente a la pregunta estará almacenado en el atributo label del elemento P, que irá asociado a un elemento de tipo Block Quote, y este a su vez colgará del elemento ItemBody asociado al AssesmentItem de tipo Numerical.

La respuesta correcta, en este caso no contendrá un identificador sino el propio valor numérico de la respuesta correcta. Elemento ResponseDeclaration, Correct Response, Value, atributo label. Además este tipo de respuestas soportan modificadores a la respuesta correcta. Por ejemplo en la figura 52 se puede observar cómo además del valor correcto para la respuesta (elemento Value 3.1415), el elemento Correct Response almacena otro valor (3.142) que será un valor máximo permitido para dar por válida esa respuesta.

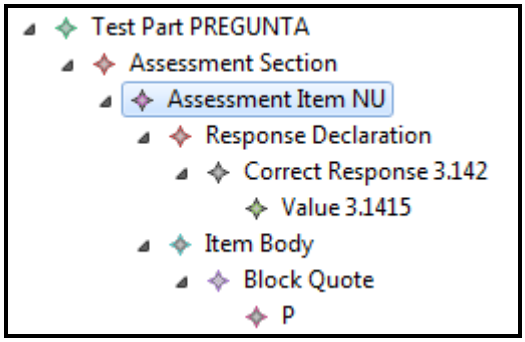


Figura 52. Pregunta Numerical en modelo conforme a MTest

En la ilustración 53 se ilustra una representación de pregunta correspondiente al tipo Numerical Question del formato Gift.

```
What is the value of pi (to 3 decimal places)? {#3.141..3.142}.
```

Figura 53. Ejemplo de pregunta Numerical

A continuación se describe como es modelada una pregunta de tipo Matching en un modelo acorde a MTest. Este tipo de preguntas dispone de pares de valores que hay que unir o emparejar en función a algún criterio. Para ello se dispone de todas las posibles respuestas organizadas a través del elemento ItemBody. Associate Interaction, Simple Associable Choice. Cada una de estos pares está identificado mediante un identificador y alberga su valor en el atributo label. La pregunta o criterio de emparejamiento a estos valores se encuentra en el elemento Prompt, también en su atributo label. En este caso y como hay que unir más dos valores el elemento Response Declaration, Correct Response contendrá varios elementos de tipo Value, cada uno de los cuales guardará e indicará los identificadores de los pares de respuesta correctas, o lo que es lo mismo, cómo quedan emparejados los pares de respuestas.

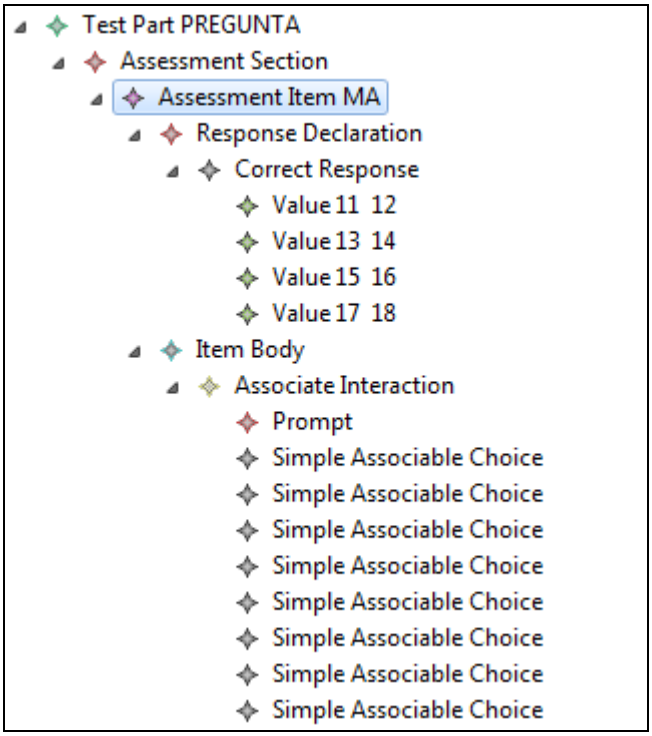


Figura 54. Pregunta Matching en modelo conforme a MTest

A continuación se puede ver como es representado en formato XML este tipo de pregunta según el estándar IMS QTI.
http://www.imsglobal.org/question/qtiv2p1/examples/items/associate.xml

```
- <assessmentItem xsi:schemaLocation="http://www.imsglobal.org/xsd/imsqti_v2p1 http://www.imsglobal.org/xsd/qti/qtiv2p1/imsqti_v2p1.xsd" identifier="associate" title="Shakespearian
  Rivals" adaptive="false" timeDependent="false">
  - <responseDeclaration identifier="RESPONSE" cardinality="multiple" baseType="pair">
    - <correctResponse>
        <value>A P</value>
        <value>C M</value>
        <value>D L</value>
      </correctResponse>
    - <mapping defaultValue="0">
        <mapEntry mapKey="A P" mappedValue="2"/>
        <mapEntry mapKey="C M" mappedValue="1"/>
        <mapEntry mapKey="D L" mappedValue="1"/>
      </mapping>
    </responseDeclaration>
    <outcomeDeclaration identifier="SCORE" cardinality="single" baseType="float"/>
  - <itemBody>
    - <associateInteraction responseIdentifier="RESPONSE" shuffle="true" maxAssociations="3">
      - <prompt>
          Hidden in this list of characters from famous Shakespeare plays are three pairs of rivals. Can you match each character to his adversary?
        </prompt>
        <simpleAssociableChoice identifier="A" matchMax="1">Antonio</simpleAssociableChoice>
        <simpleAssociableChoice identifier="C" matchMax="1">Capulet</simpleAssociableChoice>
        <simpleAssociableChoice identifier="D" matchMax="1">Demetrius</simpleAssociableChoice>
        <simpleAssociableChoice identifier="L" matchMax="1">Lysander</simpleAssociableChoice>
        <simpleAssociableChoice identifier="M" matchMax="1">Montague</simpleAssociableChoice>
        <simpleAssociableChoice identifier="P" matchMax="1">Prospero</simpleAssociableChoice>
      </associateInteraction>
    </itemBody>
    <responseProcessing template="http://www.imsglobal.org/question/qti_v2p1/rptemplates/map_response"/>
  </assessmentItem>
```

Figura 55. XML Matching para IMS QTI

En la figura 56 se presenta una captura de pantalla correspondiente a la visualización de una pregunta de tipo Matching.

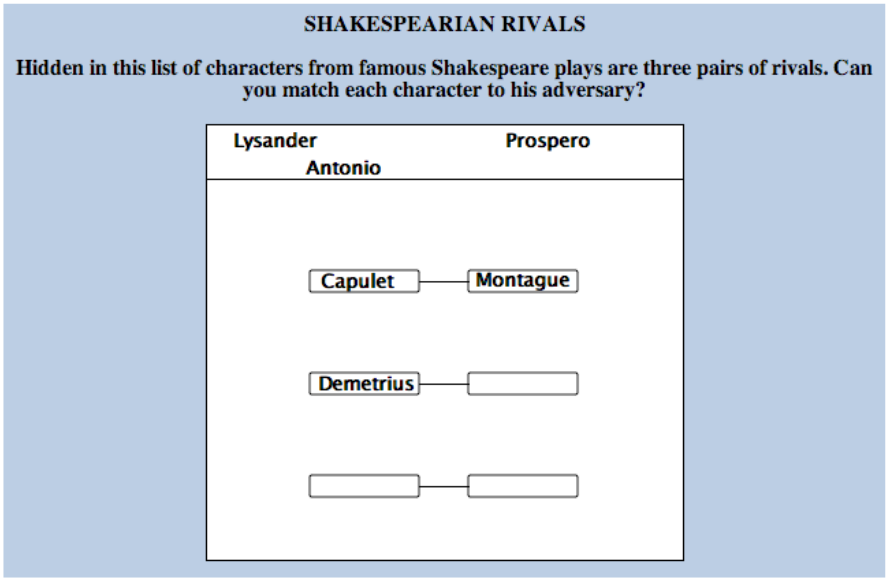


Figura 56. Ejemplo de pregunta Matching

Las preguntas de tipo Short Answer para un modelo conforme a MTest quedan de la siguiente manera: El elemento P anidado en Block Quote e Item Body contiene el enunciado de la pregunta. Este tipo de preguntas contempla una o varias repuestas posibles por lo que el elemento Assesment Item tendrá un Response Declaration por cada respuesta válida. En el ejemplo de la figura 57 se observa que para cada Response Declaration, Correct Response hay un elemento Value (uno con valor '*4*' y otro con valor '*four*'), ambos valores indicados son correctos para su pregunta asociada.

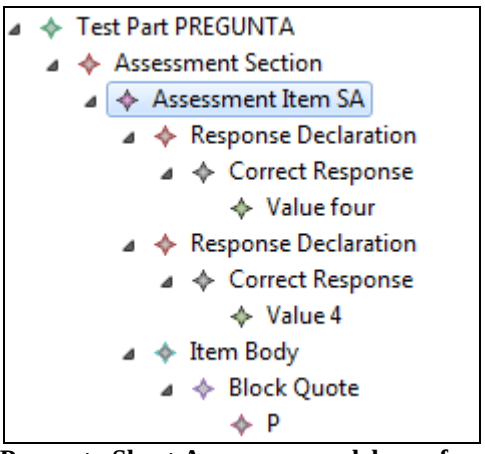


Figura 57. Pregunta Short Answer en modelo conforme a MTest

A continuación se puede ver como es representado en formato XML este tipo de pregunta según el estándar IMS QTI.
http://www.imsglobal.org/question/qtiv2p1/examples/items/text_entry.xml

```
- <assessmentItem xsi:schemaLocation="http://www.imsglobal.org/xsd/imsqti_v2p1 http://www.imsglobal.org/xsd/qti/qtiv2p1/imsqti_v2p1.xsd" identifier="textEntry" title="Richard III (Take 3)" adaptive="false" timeDependent="false">
  - <responseDeclaration identifier="RESPONSE" cardinality="single" baseType="string">
    - <correctResponse>
        <value>York</value>
      </correctResponse>
    - <mapping defaultValue="0">
        <mapEntry mapKey="York" mappedValue="1"/>
        <mapEntry mapKey="york" mappedValue="0.5"/>
      </mapping>
    </responseDeclaration>
    <outcomeDeclaration identifier="SCORE" cardinality="single" baseType="float"/>
  - <itemBody>
    - <p>
        Identify the missing word in this famous quote from Shakespeare's Richard III.
      </p>
    - <blockquote>
      - <p>
          Now is the winter of our discontent
          <br/>
          Made glorious summer by this sun of
          <textEntryInteraction responseIdentifier="RESPONSE" expectedLength="15"/>
          ;
          <br/>
          And all the clouds that lour'd upon our house
          <br/>
          In the deep bosom of the ocean buried.
        </p>
      </blockquote>
    </itemBody>
    <responseProcessing template="http://www.imsglobal.org/question/qti_v2p1/rptemplates/map_response"/>
  </assessmentItem>
```

Figura 58. XML Short Answer para IMS QTI

En la figura 59 se presenta una captura de pantalla correspondiente a la visualización de una pregunta de tipo Matching.

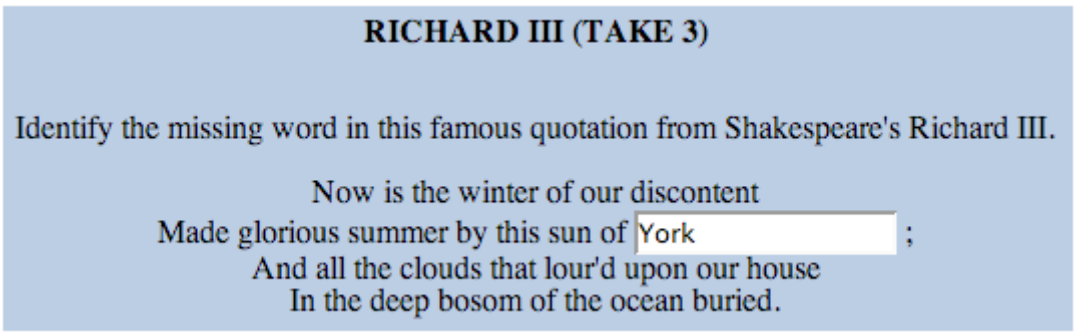


Figura 59. Ejemplo de pregunta Short Answer

En el caso de una pregunta de tipo True False para proceder a su modelado se realiza de la siguiente forma: El enunciado correspondiente a la pregunta estará almacenado en el atributo label del elemento Prompt, que irá asociado a un elemento de tipo Extended Text Interaction, y este a su vez colgará del elemento ItemBody asociado al AssesmentItem de tipoTrue False.

La respuesta correcta, en este caso contendrá el valor correspondiente a *True* o *False* en función de si el enunciado es verdadero o falso. Elemento ResponseDeclaration, Correct Response, Value, atributo label.

Además este tipo de respuestas soportan feedbacks asociados, tanto si la respuesta ha sido correcta como si se ha dado una respuesta errónea. En la figura 60 se puede apreciar como el elemento Correct Response dispone de estos feedback.

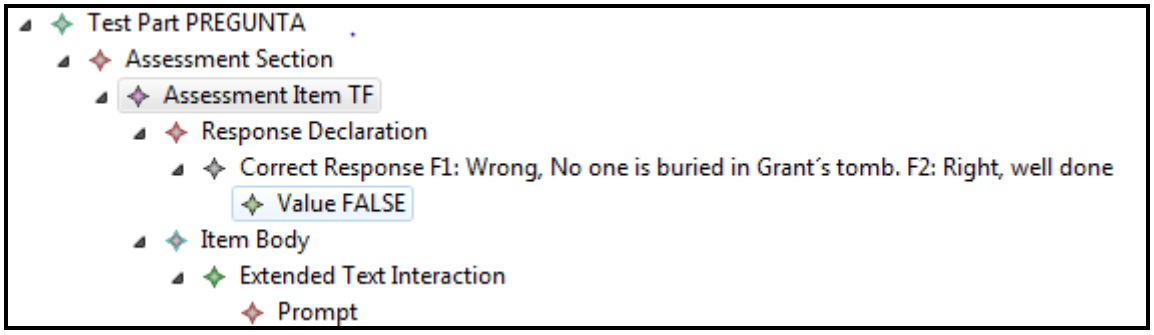


Figura 60. Pregunta True False en modelo conforme a MTest

En la figura 61 se muestra una representación de pregunta correspondiente al tipo True False del formato Gift.

```
Grant was buried in a tomb in New York City.{T}
```

Figura 61. Ejemplo de pregunta True False

Una pregunta de tipo Essay modelada en un modelo conforme a MTest no dispone de respuesta o respuestas ya que consiste en un enunciado y un bloque de texto para que sea el usuario quien desarrolle su respuesta. Así, su pregunta se almacena a través del elemento Item Body, Extended Text Interaction, atributo label del elemento Prompt (ver figura 62).

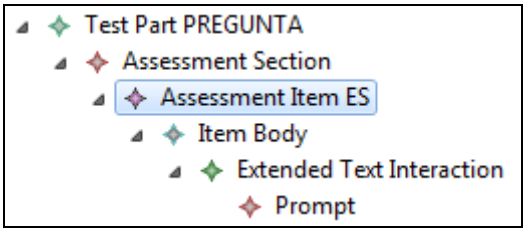


Figura 62. Pregunta Essay en modelo conforme a MTest

Seguidamente se puede ver como es representado en formato XML este tipo de pregunta según el estándar IMS QTI.
http://www.imsglobal.org/question/qtiv2p1/imsqti_implv2p1.html

```
    This example adapted from the PET Handbook, copyright University of Cambridge ESOL Examinations
  -->
- <assessmentItem xsi:schemaLocation="http://www.imsglobal.org/xsd/imsqti_v2p1 http://www.imsglobal.org/xsd/qti/qtiv2p1/imsqti_v2p1.xsd" identifier="extendedText" title="Writing a Postcard" adaptive="false" timeDependent="false">
    <responseDeclaration identifier="RESPONSE" cardinality="single" baseType="string"/>
    <outcomeDeclaration identifier="SCORE" cardinality="single" baseType="float"/>
  - <itemBody>
    - <p>
        Read this postcard from your English pen-friend, Sam.
      </p>
    - <div>
      - <object type="image/png" data="images/postcard.png">
        - <blockquote class="postcard">
          - <p>
              Here is a postcard of my town. Please send me
              <br/>
              a postcard from your town. What size is your
              <br/>
              town? What is the nicest part of your town?
              <br/>
              Where do you go in the evenings?
              <br/>
              Sam.
            </p>
          </blockquote>
        </object>
      </div>
    - <extendedTextInteraction responseIdentifier="RESPONSE" expectedLength="200">
      - <prompt>
          Write Sam a postcard. Answer the questions. Write 25-35 words.
        </prompt>
      </extendedTextInteraction>
    </itemBody>
  </assessmentItem>
```

Figura 63. XML Essay para IMS QTI

En la figura 64 se presenta una captura de pantalla correspondiente a la visualización de una pregunta de tipo Essay.

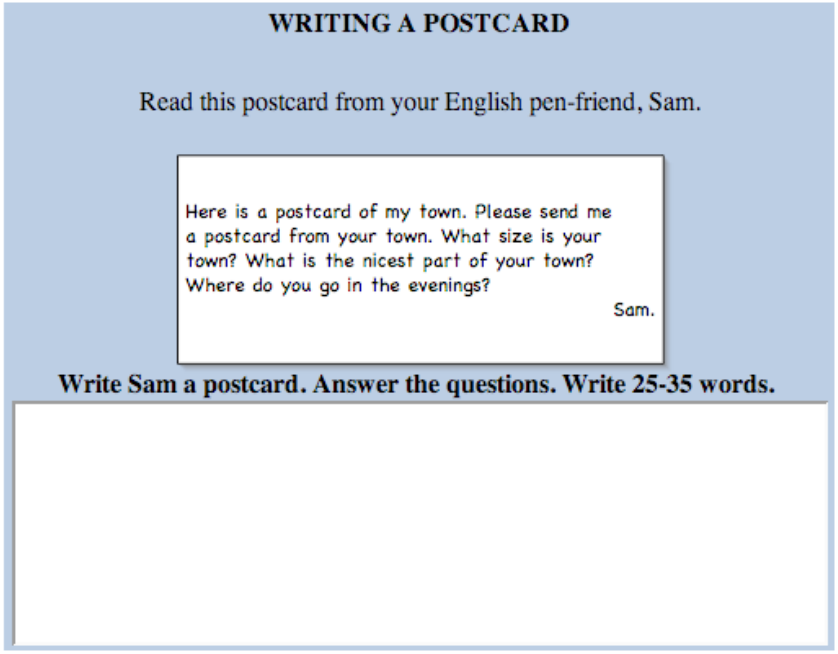


Figura 64. Ejemplo de pregunta Essay

Por último la representación de una pregunta de tipo SIMPLETEST es modelada de igual forma que una pregunta de tipo Multiple Choice. Ya que ambas están basadas en preguntas de respuesta múltiple. Lo que se pretende con este tipo de preguntas y partes de tipo SIMPLETEST, es identificar aquellos modelos de MTest que provienen de un modelo MAiken proveniente a su vez de un estándar simple de test conforme al formato Aiken.

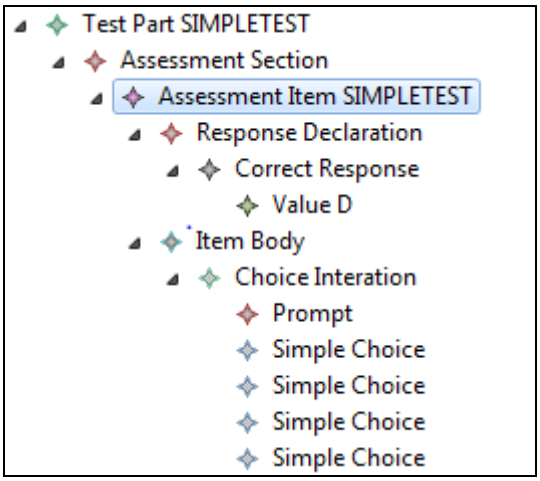


Figura 65. Pregunta Simple Test en modelo conforme a MTest

2.4.1.4. Metamodelo MGPC

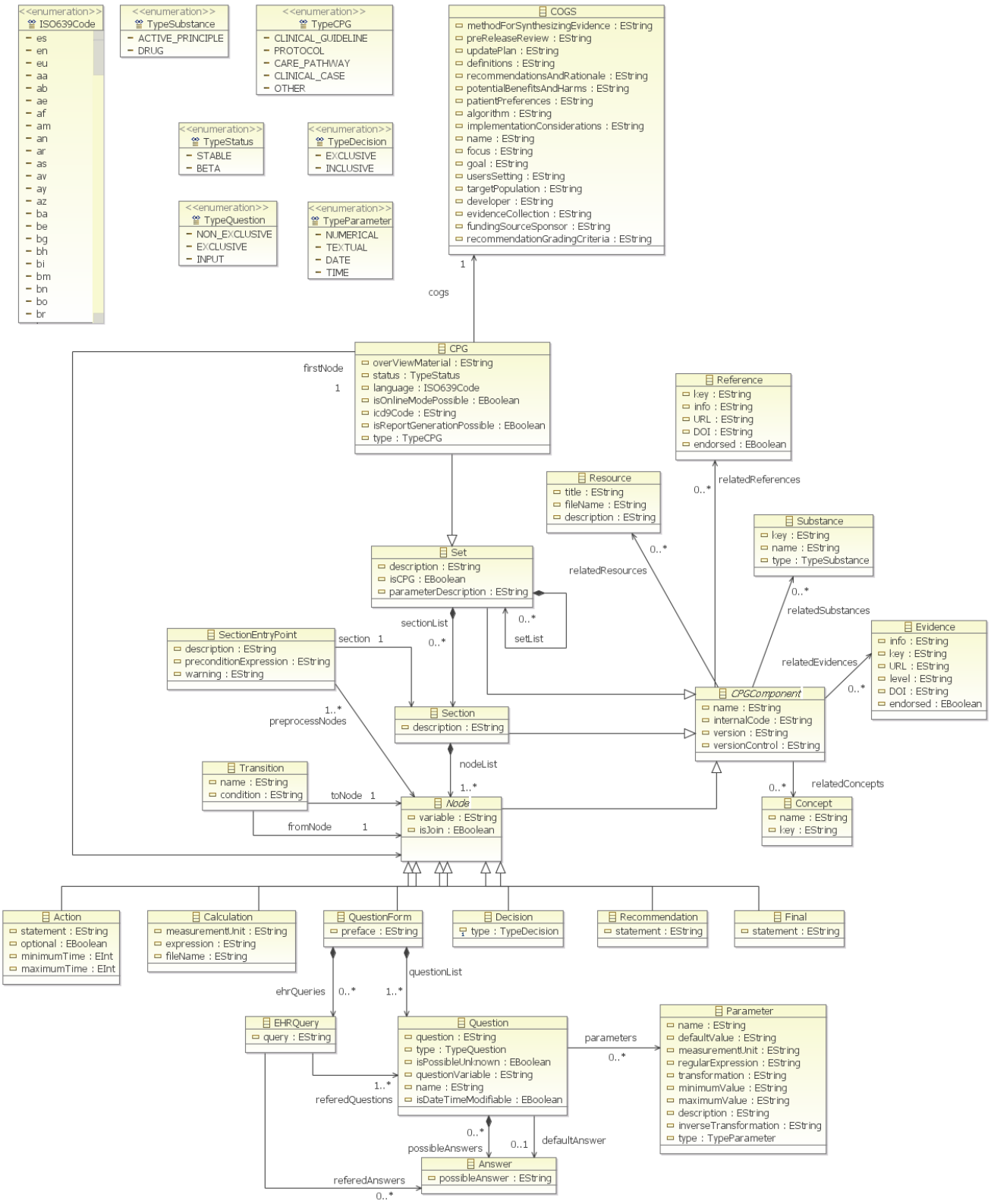


Figura 66. Metamodelo MGPC

Como transformación final, una vez obtenido un modelo conforme a Mtest, se quiere transformar la información contenida en él y traspasarla a un modelo conforme al metamodelo MGPC, usado para la representación de Guías de Práctica Clínica (GPC). Se pretende conseguir que los Test de MTest puedan verse como una Guía de Práctica Clínica Informatizada.

Una de las tareas simples identificada para las Guías de Práctica Clínica es la tarea de Recopilación de Datos. Consiste en conseguir información del paciente al que se le está aplicando la Guía de Práctica Clínica. Para llevar a cabo esta tarea de recopilación de datos se definió y para ello se utilizan, los nodos de tipo QuestionForm. Los nodos QuestionForm pueden contener un número indefinido de preguntas para dar soporte a la tarea de recopilación de datos. Es en estos nodos o elementos, donde la transformación desde modelos conformes a MTest se va a centrar, para crear los nuevos elementos y modelar correctamente la información en un modelo conforme a MGPC. Por tanto para el modelado de la información se partirá del elemento raíz CPG el cuál contendrá las diferentes secciones (Section) que tenía el test. Cada una de estas secciones contendrá un nodo (QuestionForm en este caso), del cual colgarán las diferentes preguntas (Question), donde se almacenará la información correspondiente al tipo de pregunta (atributo name), el enunciado de la pregunta (atributo question), sus correspondientes posibles respuestas y en caso de ser necesario la respuesta correcta. Además de poder guardar para una pregunta, gracias al elemento Parameter o CPG Parameter más información acerca de la misma y sus peculiaridades.

A continuación se puede ver en la figura 67 un ejemplo de una interfaz para una aplicación web en la cual se muestran diferentes tipos de preguntas para una Guía de Práctica Clínica.

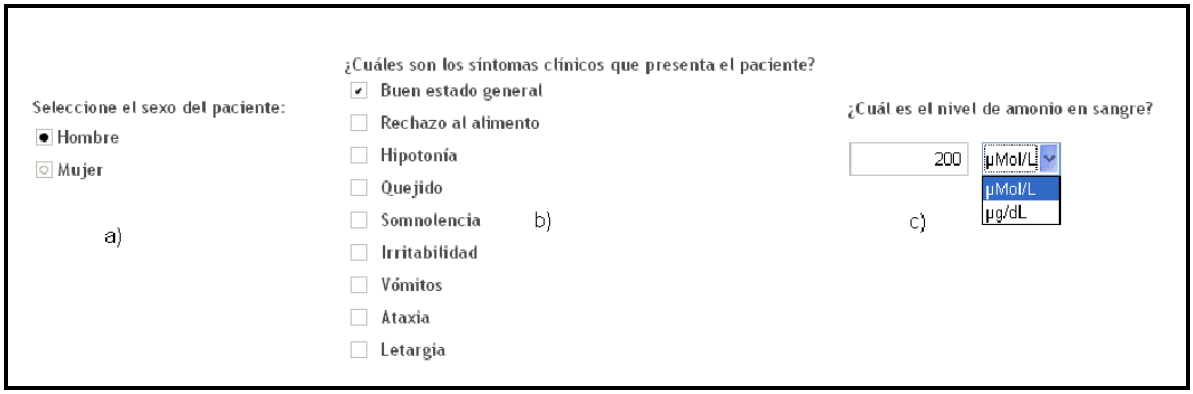
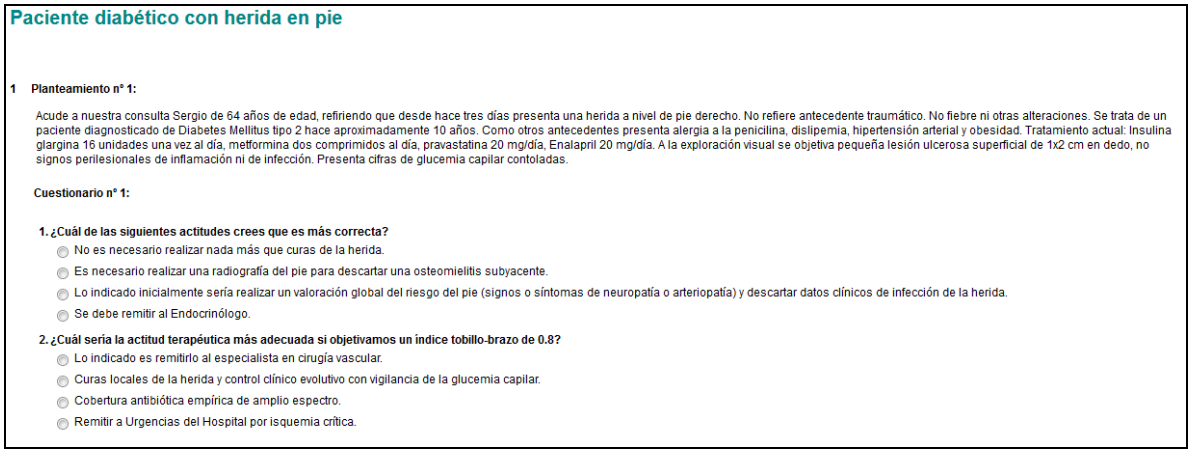


Figura 67. Posible interfaz en una aplicación web para preguntas de una GPC

Además también se podrían utilizar las preguntas modeladas para la realización de casos clínicos y poder estudiar los resultados dados. Un ejemlo se muestra en la figura 68.



Paciente diabético con herida en pie

1 Planteamiento nº 1:

Acude a nuestra consulta Sergio de 64 años de edad, refiriendo que desde hace tres días presenta una herida a nivel de pie derecho. No refiere antecedente traumático. No fiebre ni otras alteraciones. Se trata de un paciente diagnosticado de Diabetes Mellitus tipo 2 hace aproximadamente 10 años. Como otros antecedentes presenta alergia a la penicilina, dislipemia, hipertensión arterial y obesidad. Tratamiento actual: Insulina glargina 16 unidades una vez al día, metformina dos comprimidos al día, pravastatina 20 mg/día, Enalapril 20 mg/día. A la exploración visual se objetiva pequeña lesión ulcerosa superficial de 1x2 cm en dedo, no signos perilesionales de inflamación ni de infección. Presenta cifras de glucemia capilar contoladas.

Cuestionario nº 1:

1. ¿Cuál de las siguientes actitudes crees que es más correcta?
- No es necesario realizar nada más que curas de la herida.
- Es necesario realizar una radiografía del pie para descartar una osteomielitis subyacente.
- Lo indicado inicialmente sería realizar un valoración global del riesgo del pie (signos o síntomas de neuropatía o arteriopatía) y descartar datos clínicos de infección de la herida.
- Se debe remitir al Endocrinólogo.

2. ¿Cuál sería la actitud terapéutica más adecuada si objetivamos un índice tobillo-brazo de 0.8?
- Lo indicado es remitirlo al especialista en cirugía vascular.
- Curas locales de la herida y control clínico evolutivo con vigilancia de la glucemia capilar.
- Cobertura antibiótica empírica de amplio espectro.
- Remitir a Urgencias del Hospital por isquemia crítica.

Figura 68. Planteamiento de un Caso Clínico

2.4.2. Domain Specific Languages (DSL) – Gramáticas

La funcionalidad, los metamodelos y las formas de interacción de los casos de uso de la aplicación, han sido descritas con anterioridad. Ahora, se expondrán los Lenguajes Específicos de Dominio que se centrarán en explicar cómo se consigue crear un entorno de desarrollo para el usuario capaz de permitirle crear un test y/o validar uno de manera automática y acorde a uno de los formatos estándar de test Aiken o Gift.

2.4.2.1. DSL Aiken

Este Lenguaje Específico de Dominio se centra en atender las restricciones establecidas por el formato de test Aiken. A continuación se presenta un ejemplo de un test en formato Aiken con diferentes preguntas.

```
What is the correct answer to this question?
A. Is it this one?
B. Maybe this answer?
C. Possibly this one?
D. Must be this one
ANSWER: D

Which LMS has the most quiz import formats?
A) Moodle
B) ATutor
C) Claroline
D) Blackboard
E) WebCT
F) Ilias
ANSWER: A

¿Cual es la capital de España?
A) San Sebastian
B) Bilbao
C) Madrid
D) burgos
ANSWER: C
¿Que animal es el rey de la selva?
A. Mono
B. Orangutan
C. Hipopotamo pico de loro
D. Leon
E. hurraca
ANSWER: D

Who is the best Don Quijotes friend?
X) Roman
Y) Tomas
Z) Sancho
ANSWER: Z

```

Figura 69. Ejemplo de test en formato Aiken

Para disponer de un editor el cuál permita al usuario validar y crear test conformes al estándar Aiken se diseñó y definió la siguiente gramática, (ver figura 70). Esta gramática lleva asociada un metamodelo (MAiken.ecore), además de los paquetes de código es.ehu.dslMaiken-src (el cuál contiene los ficheros main, la propia gramática, un fichero xtend para transformaciones M2T, el generador de MWE2 Workflow y un paquete de validaciones), es.eshu.dslMaiken-src-gen (que contiene los códigos generados, el metamodelo asociado, el fichero genmodel y el fichero xmi acorde al metamodelo) y el paquete es.eshu.dslMaiken-xtend-gen.

```
1  grammar es.ehu.dslMAiken.MAiken with org.eclipse.xtext.common.Terminals
2
3  generate mAiken "http://www.ehu.es/dslMAiken/MAiken"
4
5  Test:
6      ('Identificador' identificador= STRING)?
7      preguntas += Pregunta +
8  ;
9
10 terminal ID_MAIKEN:
11     ('A'..'Z')
12     ;
13 terminal MAIKEN_ANSWER:
14     ('a'..'z' |'A'..'Z' |' ' | '\t' | '?' | '¿' | 'ñ')*
15     ;
16 terminal SEPARADOR_MAIKEN:
17     ('. '|') ')
18     ;
19
20 SentenceValue hidden():
21   (INT | MAIKEN_ANSWER)* LINEBREAK
22   ;
23
24 terminal LINEBREAK: ('\n' | '\r')*
25
26 ;
27
28 Pregunta:
29     pregunta = SentenceValue
30     respuestasPosibles += Respuesta +
31     'ANSWER: ' respuestaCorrecta = ID_MAIKEN LINEBREAK
32 ;
33
34 Respuesta:
35     id = ID_MAIKEN separador = SEPARADOR_MAIKEN posibleRespuesta = MAIKEN_ANSWER LINEBREAK
36 ;
37
```

Figura 70. Maiken.xtext

Esta gramática restringe los elementos, valores y datos que un usuario puede introducir para poder crear o validar un test conforme al estándar Aiken. Esto se consigue mediante las diferentes reglas y terminales definidos en la gramática. Así por ejemplo podemos observar que hay un elemento raíz de nombre Test el cual puede tener una o más preguntas. Cada una de estas ha de cumplir con las condiciones restringidas por los terminales. Además estas preguntas tendrán una o varias respuestas que han de seguir el formalismo indicado.

En la siguiente figura se puede ver el editor de desarrollo validando un test en formato Aiken.

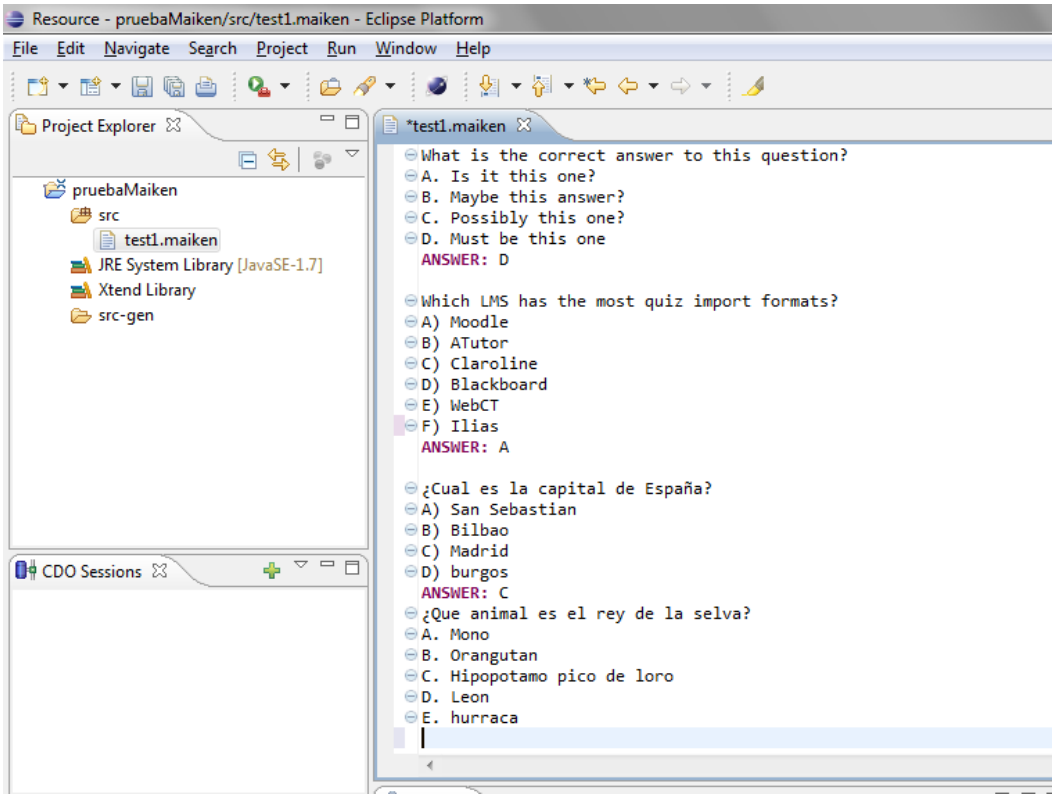


Figura 71. Editor de desarrollo para test Aiken

2.4.2.2. DSL Gift

Al igual que el anterior, este Lenguaje Específico de Dominio se centra en atender las restricciones establecidas por el formato de test Gift. A continuación se presenta un ejemplo de un test en formato Gift con diferentes tipos de preguntas.

```
¿Cual es la capital de España? {
~San Sebastian
~Bilbao
=Madrid
~Toledo
~Ninguna de las anteriores
}

//Missing Word
Grant is {~buried =entombed ~living } in Grant´s tomb.

//Missing Word Multiple lines
The American holiday of Thanksgiving is celebrated on the {
~second
~third
=fourth
} Thursday of November 6th.

//Short Answer
Who is buried in Grant´s tomb {=no one =nobody}

Two plus two equals {=four =4}

2 plus 2 equals {=four}

//if there is only one correct Short Answer, it may be written
Two plus two equals {four}

//True-False
Grant is buried in Grant´s tomb. {F}

The sun rises in the east. {T}

La capital de España es Madrid. {TRUE}

Two plus two equals 5. {FALSE}

//Matching Questions
Matching questions {
```

Figura 72. Ejemplo de test en formato Gift

Para disponer de un editor el cuál permita al usuario validar y crear test conformes al estándar Gift se diseñó y definió la siguiente gramática, (ver figura 73). Esta gramática lleva asociada un metamodelo (MGift.ecore), además de los paquetes de código es.ehu.dslMGift-src (el cuál contiene los ficheros main, la propia gramática, un fichero xtend para transformaciones M2T, el generador de MWE2 Workflow y un paquete de validaciones), es.eshu.dslMGift-src-gen (que contiene los códigos generados, el metamodelo asociado, el fichero genmodel y el fichero xmi acorde al metamodelo) y el paquete es.eshu.dslMGift-xtend-gen.

```
grammar es.ehu.dslMGift.MGift
import "http://www.eclipse.org/emf/2002/Ecore" as ecore
generate mGift "http://www.ehu.es/dslMGift/MGift"

TestGift:
	partesOrdenadas += TestPart +;

TestPart:
	Comentario | Pregunta | Descripcion;

terminal SL_COMMENT  : '//' !('\n'|'\r')* ('\r'? '\n')? ;

Comentario:
	comentario = SL_COMMENT ;

Descripcion:
	description = SentenceValue;

Pregunta:
	MultipleChoice_Matching_ShortAnswer_Numerical_Essay | MissingWord |
TrueFalse;

MultipleChoice_Matching_ShortAnswer_Numerical_Essay:
	Questiontype = (QuestionType)?
	pregunta = QuestionValue
	(LINEBREAK)?
	respuestasPosibles += (Respuesta)*
	'}'
	LINEBREAK;

MissingWord:

	pregunta = QuestionValue
	(LINEBREAK)?
	respuestasPosibles += (Respuesta)+
	'} ' parteDos = ParteDosValue
	LINEBREAK;

ParteDosValue hidden():
	(ENTEROS | MGIFT_ANSWER | '.')+;

TrueFalse:
	pregunta = QuestionValue
	respuestasPosibles += (Respuesta)*
	respuesta = ('TRUE' | 'FALSE' | 'T' | 'F')
	(('#'feedback1 = MGIFT_ANSWER '.''#'feedback2= MGIFT_ANSWER'.') |
'#'feedback1 = MGIFT_ANSWER)?
	'}'
	LINEBREAK;

Respuesta:
	Respuesta_Estandar | Par | Numerical;
```

```
Respuesta_Estandar:
	Questiontype = (QuestionType)?
	(prefijo = ('~' | '='))? ('%'puntuacion = NegativeWeightsValue'%')?
respuesta = (MGIFT_ANSWER | ENTEROS) ('#'feedback=MGIFT_ANSWER ('.')?)?
(LINEBREAK)?;

Par:
	prefijo = ('=') respuesta = MGIFT_ANSWER '->' suPareja = MGIFT_ANSWER
(LINEBREAK)?;

Numerical:
	prefijo = ('#') respuesta = NumericalValue ((':'
errorMargin=NumericalValue) | ('..' maxValue=NumericalValue))?;

terminal MGIFT_ANSWER:
	('a'..'z' | 'A'..'Z' | ' ' | '\t' | '?' | '¿' | '!' | 'ñ' | '(' | ')'
| '´' | '[' | ']' | ',')*;

NegativeWeightsValue hidden():
	(ENTEROS | '-')+;

QuestionType hidden(): '@';

 terminal ENTEROS:
	('0'..'9')*;

NumericalValue hidden():
	(ENTEROS | '.')+ ;

QuestionValue hidden():
	(ENTEROS | MGIFT_ANSWER | '.' | ':')* QUESTION_END;

terminal QUESTION_END: '{';

SentenceValue hidden():
	(ENTEROS | MGIFT_ANSWER | '.')* LINEBREAK;

terminal LINEBREAK: ('\n' | '\r')* ;
```

Figura 73. MGift.xtext

Se puede apreciar a simple vista que esté DSL es bastante más complejo que el desarrollado para el formato Aiken. Esta gramática restringe los elementos, valores y datos que un usuario puede introducir para poder crear o validar un test conforme al estándar Gift. Esto se consigue mediante las diferentes reglas y terminales definidos en la gramática. De considerable dificultad dadas las restricciones establecidas por la entrada de un estándar acorde al formato Gift, hubo que crear diferentes elementos, reglas de herencia y terminales para poder cubrir todas las necesidades y pensar en el desarrollo más correcto para que no se solapasen diferentes opciones a la hora de crear el test.

Así por ejemplo podemos observar que hay un elemento raíz de nombre TestGift el cual puede tener una o más partes ordenadas. Cada una de estas ha de cumplir con las condiciones restringidas por los terminales y podrán ser de uno de los tres tipos indicados.

Hay preguntas con varias respuestas, preguntas sin respuestas, diferentes prefijos que indican si una respuesta es o no correcta, feedbacks de apoyo a las preguntas, comentarios, etc.

En la siguiente figura se puede ver el editor de desarrollo validando un test en formato Gift.

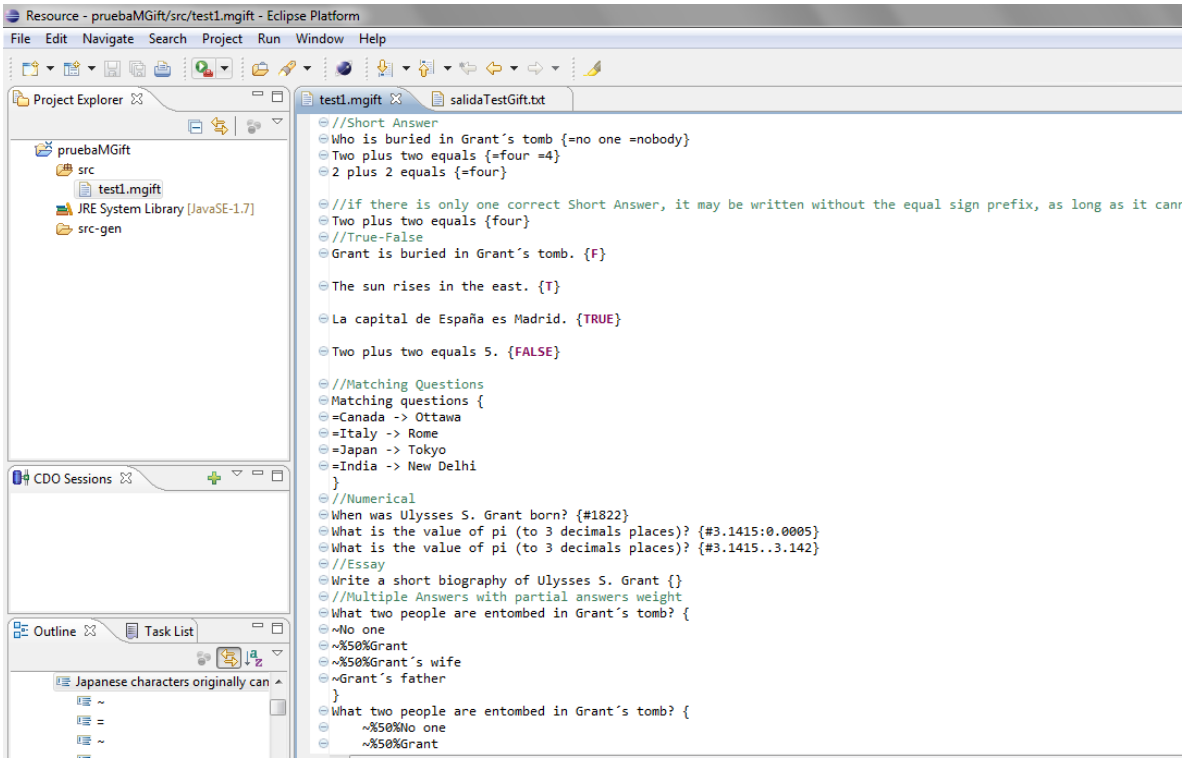


Figura 74. Editor de desarrollo para test Gift

2.4.3. Transformaciones entre metamodelos

En este apartado se muestran las tablas de adaptación correspondientes a las transformaciones de un modelo origen conforme a un metamodelo, en un modelo destino conforme a otro metamodelo.

2.4.3.1. MAiken2MTest

La primera de las tablas de transformación en mostrar será la que transforma modelos conformes al metamodelo MAiken en modelos conformes a metamodelos MTest. Esta tabla sirve de guía para comprender y desarrollar la regla de transformación correspondiente (MAiken2MTest.atl) escrita en el lenguaje ATL Transformation Language.

Tabla 6. MAiken2MTest

Metamodelo MAiken		**Metamodelo MTest**	
Metaclase	**Atributo**	**Metaclase**	**Atributo**
Test	identificador	AssesmentTest	identifier
Pregunta	pregunta	AssesmentItem -> ItemBody -> ChoiceInteraction->Prompt	label
	respuestaCorrecta	AssesmentItem -> ResponseDeclaration -> CorrectResponse ->Value	fieldIdentifier
Respuesta	id	AssesmentItem -> ItemBody -> ChoiceInteraction -> SimpleChoice	identifier
	separador	-	-
	posibleRespuesta	AssesmentItem -> ItemBody -> ChoiceInteraction -> SimpleChoice	label

2.4.3.2. MGift2MTest

La siguiente de las tablas de transformación en mostrar será la que transforma modelos conformes al metamodelo MGift en modelos conformes a metamodelos MTest. Esta tabla sirve de guía para comprender y desarrollar la regla de transformación correspondiente (MGift2MTest.atl) escrita en el lenguaje ATL Transformation Language.

Tabla 7. MGift2MTest

Metamodelo MGift		**Metamodelo MTest**	
Metaclase	**Atributo**	**Metaclase**	**Atributo**
TestGift	-	AssesmentTest	-
Comentario	comentario	AssesmentItem	label
Descripcion	description	AssesmentItem	label
MutipleChoice_Matching_ ShortAnswer_Numerical_Essay (para el MultipleChoice)	pregunta	AssesmentItem -> ItemBody -> ChoiceInteraction -> Prompt	label
MutipleChoice_Matching_ ShortAnswer_Numerical_Essay (para el Numerical)	pregunta	AssesmentItem -> ItemBody -> blockQuote -> P	label
MutipleChoice_Matching_ ShortAnswer_Numerical_Essay (para el ShortAnswer)	pregunta	AssesmentItem -> ItemBody -> blockQuote-> P	label
MutipleChoice_Matching_ ShortAnswer_Numerical_Essay (para el Matching)	pregunta	AssesmentItem -> ItemBody -> associateInteraction -> Prompt	label
MutipleChoice_Matching_ ShortAnswer_Numerical_Essay (para el Essay)	pregunta	AssesmentItem -> ItemBody -> BlockInteraction -> extendedTextInteraction -> Prompt	label
MissingWord	pregunta	AssesmentItem -> ItemBody->blockQuote -> P	label
	parteDos	AssesmentItem -> ItemBody -> blockQuote -> P	
TrueFalse	pregunta	ExtendedTextInteraction -> Prompt	label
	respuesta	CorrectResponse -> Value	fieldIdentifier
	feedback1	CorrectResponse	interpretation
	feedback2		
Respuesta_Estandar (para el MultipleChoice) *	prefijo	-	-
	respuesta	SimpleChoice y la correcta en responseDeclaration -> CorrectResponse -> Value	Label y fieldIdentifier
	puntuacion	CorrectResponse	interpretation
	feedback		

Respuesta_Estandar (para el MissingWord)	prefijo	-	-
	respuesta	ItemBody->Block -> BlockStatic -> AtomicBlock -> Inline -> InlineInteraction -> InlineChoiceInteraction -> InlineChoice	Label y fieldIdentifier
	puntuacion	CorrectResponse	interpretation
	feedback		
Respuesta_Estandar (para el ShortAnswer)	prefijo	-	-
	respuesta	CorrectResponse -> Value	fieldIdentifier
	puntuacion	CorrectResponse	interpretation
	feedback		
Par **	prefijo	-	-
	respuesta	SimpleAssociableChoice	label
	suPareja	SimpleAssociableChoice	label
Numerical	prefijo	-	-
	respuesta	CorrectResponse->Value	fieldIdentifier
	errorMargin	CorrectResponse	interpretation
	maxValue		

En el elemento TestPart, atributo identifier se guardará el tipo de parte al que corresponde. Esto es, si es un comentario, una descripción o una pregunta. Las partes que sean de tipo Comentario o Descripción guardarán su información directamente en el elemento AssesmentItem, mientras que para las partes de tipo Pregunta se seguirá navegando por los elementos del metamodelo MTest.

* No se ha de guardar el prefijo. Una respuesta errónea tiene el prefijo ~ y una correcta el prefijo =. Asignación automática de un identificador que relacione la respuesta correcta con el identificador de las posibles respuestas.

** Para el elemento Par, se ha de conseguir que se correspondan los valores de CorrectResponse -> Value -> FieldIdentifier con dos de los elementos SimpleAssociableChoice -> identifier, para los cuales se habrá generado de forma automática ese valor identificador.

2.4.3.3. MTest2MGPC

La última de las tablas de transformación es la que transforma modelos conformes al metamodelo MTest en modelos conformes a metamodelos MGPC. Esta tabla sirve de guía para comprender y desarrollar su regla de transformación correspondiente (MTest2MGPC.atl) escrita en el lenguaje ATL Transformation Language.

Tabla 8. MTest2MGPC

Metamodelo MTest		**Metamodelo MGPC**	
Metaclase	**Atributo**	**Metaclase**	**Atributo**
AssesmentTest	identifier	CPG	description
TestPart	identifier	Section	description
Simpletest, Comentario, Descripcion y MultipleChoice			
AssementSection	-	QuestionForm	-
AssesmentItem	identifier	Question	name
	label		question
AssesmentItem -> itemBody -> ChoiceInteraction -> Prompt	label		question
AssesmentItem -> itemBody -> ChoiceInteraction -> SimpleChoice	label y identifier	Answer	possibleAnswer
AssesmentItem -> ResponseDeclaration -> CorrectResponse -> Value	label	CPGParameter	description y defaultValue
Numerical			
AssesmentItem	identifier	Question	name
	label		question
AssesmentItem -> ItemBody-> BlockQuote -> P	label	Question	question
AssesmentItem -> ResponseDeclaration -> CorrectResponse	interpretation	CPGParameter	name
AssesmentItem -> ResponseDeclaration -> CorrectResponse -> Value	fieldIdentifier	Answer	possibleAnswer
		Question	defaultAnswer
Matching			
AssesmentItem	identifier	Question	name
	label		question
AssesmentItem -> ItemBody -> AssociateInteraction -> Prompt	label	Question	question
AssesmentItem -> ItemBody -> AssociateInteraction -> SimpleAssociableChoice	identifier	Answer	possibleAnswer
	label		
AssesmentItem -> ResponseDeclaration -> CorrectResponse -> Value	fieldIdentifier	CPGParameter	defaultValue
			description

ShortAnswer			
AssesmentItem	identifier	Question	name
	label		question
AssesmentItem -> ItemBody-> BlockQuote -> P	label	Question	question
AssesmentItem -> ResponseDeclaration	-	CPGParameter	-
AssesmentItem -> ResponseDeclaration -> CorrectResponse -> Value	fieldIdentifier	CPGParameter	defaultValue
			description
Essay			
AssesmentItem	identifier	Question	name
	label		question
AssesmentItem -> ItemBody -> ExtendedTextInteraction -> Prompt	label	Question	question
MissingWord			
AssesmentItem	identifier	Question	name
	label		question
AssesmentItem -> ItemBody -> BlockQuote -> P	label	Question	question
AssesmentItem -> ItemBody -> InlineChoiceInteraction -> InlineChoice	identifier	Answer	possibleAnswer
	label		
AssesmentItem -> ResponseDeclaration -> CorrectResponse -> Value	fieldIdentifier	CPGParameter	defaultValue
			description
True False			
AssesmentItem	identifier	Question	name
	label		question
AssesmentItem -> ItemBody -> ExtendedTextInteraction -> Prompt	label	Question	question
AssesmentItem -> ResponseDeclaration -> CorrectResponse	interpretation	CPGParameter	description
			name
AssesmentItem -> ResponseDeclaration -> CorrectResponse -> Value	fieldIdentifier	Answer	possibleAnswer

En estas transformaciones, se modelan los elementos partiendo de un elemento raíz CPG el cual tendrá una o varias secciones (Section) en función de los elementos TestPart que tuviera el modelo de origen. Así por ejemplo para el caso de TestPart de tipo SimpleTest en el modelo de origen, se tendrá únicamente una sección y un nodo QuestionForm para el modelo destino. Los diferentes AssesmentItem del modelo de origen quedarán relacionados dentro del elemento QuestionForm en forma de Question en el modelo destino.

En el caso de que el elemento de tipo TestPart no sea SimpleTest, se transformará cada parte (TestPart) del modelo de origen, en un elemento Section del modelo destino. Cada uno de estos tendrá asociado su nodo QuestionForm y en el elemento Question se indicará el tipo al que corresponde y se procederá a su transformación según la tabla arriba indicada.

2.5. Desarrollo del proyecto

Este punto consistirá en explicar el trabajo desarrollado así como la forma en el que se ha hecho. Para ello nos apoyaremos en el entorno de desarrollo elegido, su funcionamiento y método de trabajo. A su vez se explicarán las decisiones tomadas, en referencia al diseño, así como a la implementación, sin olvidar las complicaciones surgidas durante el desarrollo del proyecto.

2.5.1. Entorno de desarrollo Eclipse versión Indigo Service Release 2

Este entorno ofrece distintas posibilidades de visualización de elementos a la hora de trabajar con ellos. Así, se puede ver el diseño y apariencia que tendrá un metamodelo en modo arborescente, en modo gráfico o de diagrama, en modo texto, en modo de edición de código. La siguiente imagen corresponde a una captura del entorno de desarrollo general.

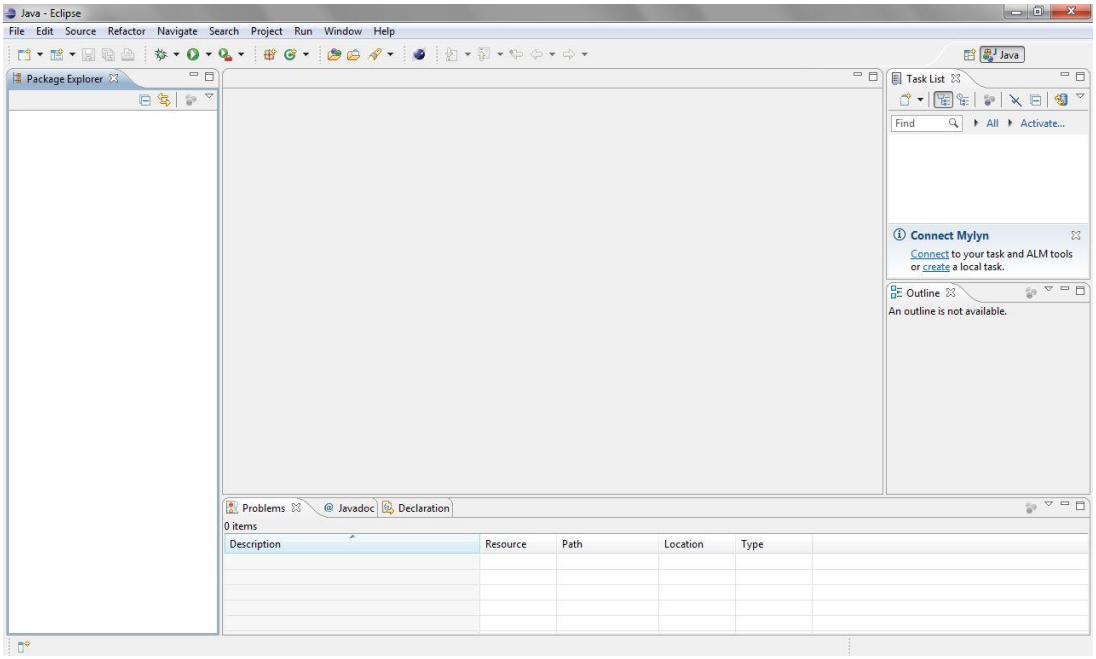


Figura 75. Entorno de desarrollo Eclipse

Se pueden diferenciar distintas partes dentro del entorno de edición. Se dispone de una ventana central donde se podrá visualizar metamodelos en modo diagrama o en modo arborescente, el código de los programas en diferentes lenguajes, las reglas de transformación, etc. Es la parte principal de edición. En el lateral izquierdo se dispone de un "Package Explorer" el cual nos permite navegar por los elementos de los que se dispone e incluso crear nuevos. Es la forma de organizar los elementos, explorador de proyectos y archivos. La parte derecha de la pantalla ocupa el lugar encargado de mostrar propiedades de control, o de elementos, así como un Outline. Para terminar en el centro inferior de la pantalla, se observa el cuadro de errores, problemas, propiedades y consola que nos va informando de errores de compilación y posibles problemas y en el que podremos ver y editor propiedades de los elementos.

La aplicación está contenida bajo cuatro proyectos separados: MAiken, MGift, MTest y M2M. El primero, MAiken, contiene la gramática asociada a Aiken, el metamodelo para MAiken, su diagrama, el editor de desarrollo para el usuario (validación y creación de test en formato Aiken) y la aplicación que genera de forma automática un modelo conforme a MAiken teniendo como entrada un documento de texto en el formato estándar de test Aiken.

El segundo proyecto, MGift contiene la gramática asociada a Gift, el metamodelo para MGift, su diagrama, el editor de desarrollo para el usuario (validación y creación de test en formato Gift) y la aplicación que genera de forma automática un modelo conforme a MGift teniendo como entrada un documento de texto en el formato estándar de test Gift.

El tercero de los proyectos se usó para desarrollar el metamodelo MTest acorde a las restricciones del estándar IMS QTI. Contiene por tanto el metamodelo asociado a MTest y su diagrama.

El último de los proyectos M2M aúna todas las reglas de transformación entre los modelos. Contiene las reglas de transformación MAiken2MTest, MGift2MTest y MTest2MGPC, escritas en el lenguaje ATL y alojadas en la carpeta rule. Además contiene la carpeta metamodel, donde se encuentran los metamodelos MAiken.ecore, MGift.ecore, MTest.ecore y MGPC.ecore. La carpeta model, donde se encuentran los diferentes modelos de origen en formato xmi, uno para Aiken, otro para Gift y otro para MTest. Por último la carpeta output sirve para alojar los modelos xmi generados gracias a las reglas de transformación.

2.5.2. Complicaciones

Durante el desarrollo del presente proyecto, han surgido varias complicaciones. Si bien, la más importante podría considerarse la adaptación al uso de las tecnologías y herramientas utilizadas para el desarrollo del mismo, en comparación con lo aprendido durante la carrera, también ha ocupado un papel determinante el costoso trabajo de búsqueda y recopilación de información para los estándares de test utilizados, así como la complejidad en cuanto al entendimiento de los objetivos del proyecto.

Las principales eventualidades surgidas fueron causadas por un conocimiento inicial básico de las tecnologías y estándares con los que se iba a trabajar.

Al margen de estas complicaciones, afortunadamente no han surgido problemas de hardware o de incompatibilidades entre software y máquina, por lo que problemas de este tipo no supusieron un retraso en el proyecto. No obstante, se realizaron copias de seguridad, almacenadas en dispositivos externos, durante la realización de todo el proyecto con el fin de poder mitigar el impacto (si ocurriese) y reducir así el tiempo de exposición provocado por distintas eventualidades.

2.5.3. Orden de ejecución

Se presenta, a continuación, el orden de ejecución lógico seguido por la aplicación gracias a la visualización de un diagrama de transición, para cubrir los objetivos del proyecto desde el inicio al fin.

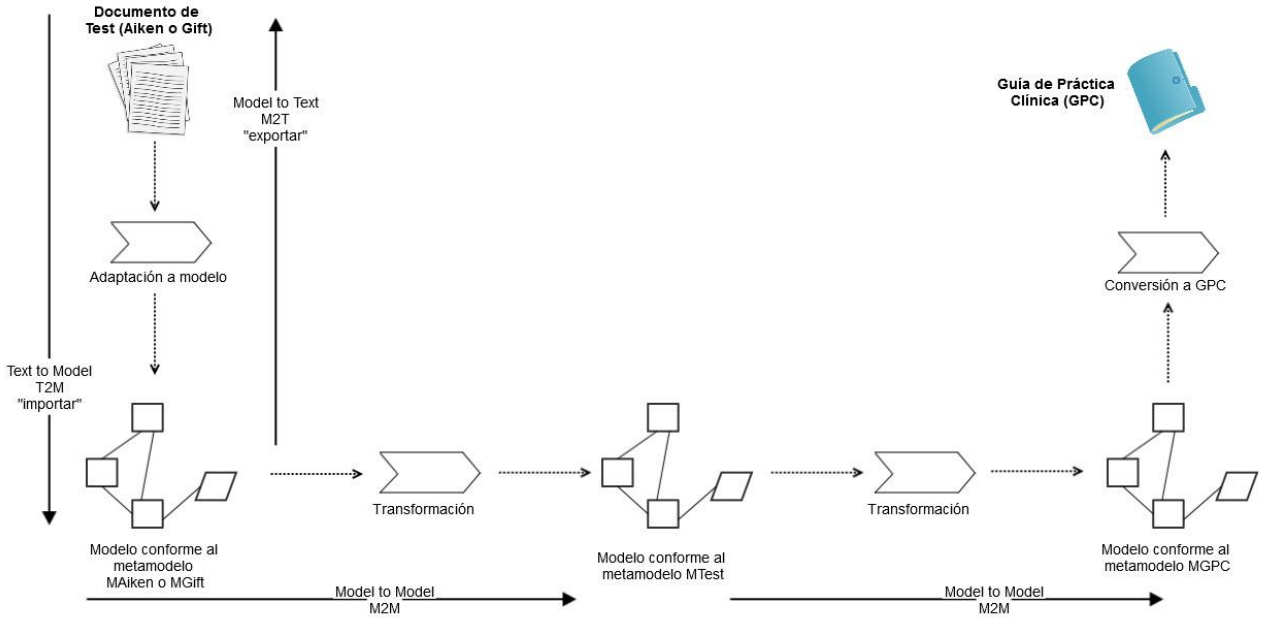


Figura 3. Arquitectura de la Aplicación para la generación de Guías de Test

2.5.4. Desarrollo del CU Validar Test Aiken/Gift

En este caso de uso se le proporciona al usuario un editor de desarrollo mediante el cual puede validar un test contra el estándar de test Aiken o Gift. La gramática o DSL Maiken.xtext, MGift.xtext (ver figuras 70 y 73), tiene asociada un metamodelo en el cual se almacena la información validada por la gramática. Para crear la gramática en el proyecto de Eciplse que contiene este caso de uso se ha usado un proyecto de tipo Xtext el cual generará de manera automática el código necesario asociado a la gramática y al metamodelo.

```
grammar es.ehu.dslMAiken.MAiken with org.eclipse.xtext.common.Terminals

generate mAiken "http://www.ehu.es/dslMAiken/MAiken"

Test:
    ('Identificador' identificador= STRING)?
    preguntas += Pregunta +
;

terminal ID_MAIKEN:
    ('A'..'Z')
    ;
terminal MAIKEN_ANSWER:
    ('a'..'z' |'A'..'Z' |' ' | '\t' | '?' | '¿' | 'ñ')*
    ;
terminal SEPARADOR_MAIKEN:
    ('. '|') ')
    ;

SentenceValue hidden():
  (INT | MAIKEN_ANSWER)* LINEBREAK
  ;

terminal LINEBREAK: ('\n' | '\r')*

;

Pregunta:
    pregunta = SentenceValue
    respuestasPosibles += Respuesta +
    'ANSWER: ' respuestaCorrecta = ID_MAIKEN LINEBREAK
;

Respuesta:
    id = ID_MAIKEN separador = SEPARADOR_MAIKEN posibleRespuesta = MAIKEN_ANSWER LINEBREAK
;

```

Figura 70. Maiken.xtext

```
grammar es.ehu.dslMGift.MGift
import "http://www.eclipse.org/emf/2002/Ecore" as ecore
generate mGift "http://www.ehu.es/dslMGift/MGift"

TestGift:
      partesOrdenadas += TestPart +;

TestPart:
      Comentario | Pregunta | Descripcion;
```

```
terminal SL_COMMENT  : '//' !('\n'|'\r')* ('\r'? '\n')? ;

Comentario:
	comentario = SL_COMMENT ;

Descripcion:
	description = SentenceValue;

Pregunta:
	MultipleChoice_Matching_ShortAnswer_Numerical_Essay | MissingWord |
TrueFalse;

MultipleChoice_Matching_ShortAnswer_Numerical_Essay:
	Questiontype = (QuestionType)?
	pregunta = QuestionValue
	(LINEBREAK)?
	respuestasPosibles += (Respuesta)*
	'}'
	LINEBREAK;

MissingWord:

	pregunta = QuestionValue
	(LINEBREAK)?
	respuestasPosibles += (Respuesta)+
	'} ' parteDos = ParteDosValue
	LINEBREAK;

ParteDosValue hidden():
	(ENTEROS | MGIFT_ANSWER | '.')+;

TrueFalse:
	pregunta = QuestionValue
	respuestasPosibles += (Respuesta)*
	respuesta = ('TRUE' | 'FALSE' | 'T' | 'F')
	(('#'feedback1 = MGIFT_ANSWER '.''#'feedback2= MGIFT_ANSWER'.') |
'#'feedback1 = MGIFT_ANSWER)?
	'}'
	LINEBREAK;

Respuesta:
	Respuesta_Estandar | Par | Numerical;




Respuesta_Estandar:
	Questiontype = (QuestionType)?
	(prefijo = ('~' | '='))? ('%'puntuacion = NegativeWeightsValue'%')?
respuesta = (MGIFT_ANSWER | ENTEROS) ('#'feedback=MGIFT_ANSWER ('.')?)?
(LINEBREAK)?;

Par:
	prefijo = ('=') respuesta = MGIFT_ANSWER '->' suPareja = MGIFT_ANSWER
(LINEBREAK)?;
```

```
Numerical:
	prefijo = ('#') respuesta = NumericalValue ((':'
errorMargin=NumericalValue) | ('..' maxValue=NumericalValue))?;

terminal MGIFT_ANSWER:
	('a'..'z' | 'A'..'Z' | ' ' | '\t' | '?' | '¿' | '!' | 'ñ' | '(' | ')'
| '´' | '[' | ']' | ',')*;

NegativeWeightsValue hidden():
	(ENTEROS | '-')+;

QuestionType hidden(): '@';

 terminal ENTEROS:
	('0'..'9')*;

NumericalValue hidden():
	(ENTEROS | '.')+ ;

QuestionValue hidden():
	(ENTEROS | MGIFT_ANSWER | '.' | ':')* QUESTION_END;

terminal QUESTION_END: '{';

SentenceValue hidden():
	(ENTEROS | MGIFT_ANSWER | '.')* LINEBREAK;

terminal LINEBREAK: ('\n' | '\r')* ;
```

Figura 73. MGift.xtext

Si un test es válido, de forma automática se genera en segundo plano un modelo, y gracias a la transformación de modelo a texto con la plantilla MaikenGenerator.xtend, MGiftGenerator.xtend se genera en la carpeta src-gen el mismo test en un fichero de extensión txt. Si un test no cumple con algún criterio o restricción establecida, el editor de forma automática lo valida señala y sugiere el error al usuario.

```
package es.ehu.dslMAiken.generator

import org.eclipse.emf.ecore.resource.Resource
import org.eclipse.xtext.generator.IGenerator
import org.eclipse.xtext.generator.IFileSystemAccess
import es.ehu.dslMAiken.mAiken.Pregunta
import es.ehu.dslMAiken.mAiken.Respuesta

class MAikenGenerator implements IGenerator {

override void doGenerate(Resource resource, IFileSystemAccess fsa) {

	fsa.generateFile("salidaTestAiken.txt", resource.compile);

	}
```

```
	def Iterable<Pregunta> getList(Resource r){
		return r.allContents.toList().filter(typeof(Pregunta));
	}

		def Iterable<Respuesta> getRespuestas(Pregunta p){
		return p.respuestasPosibles;
	}

	def compile (Resource r) '''
	//TEST AIKEN (MODEL2TEXT)

	«FOR e:r.getList»
		«e.pregunta»
		«FOR i:e.getRespuestas»
		«i.id»«i.separador»«i.posibleRespuesta»
		«ENDFOR»
		ANSWER: «e.respuestaCorrecta»

	«ENDFOR»
	'''

}
```

Figura 76. MAikenGenerator.xtend

Estos ficheros de extensión xtend actúan como una plantilla que va recorriendo y cargando la información procedente de un modelo para después generar en este caso un fichero de texto con los criterios establecidos para el formato estándar de test en concreto, Aiken o Gift.

```
package es.ehu.dslMGift.generator

import org.eclipse.emf.ecore.resource.Resource
import org.eclipse.xtext.generator.IGenerator
import org.eclipse.xtext.generator.IFileSystemAccess
import es.ehu.dslMGift.mGift.TestPart
import es.ehu.dslMGift.mGift.Pregunta
import es.ehu.dslMGift.mGift.Respuesta
import es.ehu.dslMGift.mGift.TrueFalse
import es.ehu.dslMGift.mGift.Comentario
import es.ehu.dslMGift.mGift.Descripcion
import
es.ehu.dslMGift.mGift.MultipleChoice_Matching_ShortAnswer_Numerical_Essay
import es.ehu.dslMGift.mGift.MissingWord
import es.ehu.dslMGift.mGift.Respuesta_Estandar
import es.ehu.dslMGift.mGift.Par
import es.ehu.dslMGift.mGift.Numerical

class MGiftGenerator implements IGenerator {

	override void doGenerate(Resource resource, IFileSystemAccess fsa) {

fsa.generateFile("salidaTestGift.txt", resource.compile);
	}
```

```
	def Iterable<TestPart> getListParts(Resource r){
		return r.allContents.toList().filter(typeof(TestPart));
	}

		def Iterable<Respuesta> getRespuestas(Pregunta p){
		return p.respuestasPosibles;
	}

	def compile (Resource r) '''
	««««//TEST GIFT (MODEL2TEXT)
	«FOR e:r.getListParts»
		«IF e instanceof Comentario»
			«e.dameComentario.comentario»
		«ELSEIF e instanceof Descripcion»
			«e.dameDescripcion.description»
		«ELSEIF e instanceof Pregunta»
			«e.damePregunta.pregunta»
			«IF e.damePregunta instanceof
MultipleChoice_Matching_ShortAnswer_Numerical_Essay»
				«IF e.damePregunta.getRespuestas !==null»
				«FOR j:e.damePregunta.getRespuestas»
				«IF j instanceof Respuesta_Estandar»
					«IF j.dameRespEstandar.feedback !== null»
						«IF j.dameRespEstandar.puntuacion
!==null»

	«j.dameRespEstandar.prefijo»%«j.dameRespEstandar.puntuacion»%«j.dameRe
spEstandar.respuesta»#«j.dameRespEstandar.feedback»
						«ELSE»

	«j.dameRespEstandar.prefijo»«j.dameRespEstandar.respuesta»#«j.dameResp
Estandar.feedback»
						«ENDIF»
					«ELSE»
						«IF j.dameRespEstandar.puntuacion
!==null»

	«j.dameRespEstandar.prefijo»%«j.dameRespEstandar.puntuacion»%«j.dameRe
spEstandar.respuesta»
						«ELSE»

	«j.dameRespEstandar.prefijo»«j.dameRespEstandar.respuesta»
						«ENDIF»
					«ENDIF»
				«ELSEIF j instanceof Numerical»««««//tengo que ver
si tiene margin o max value
					«IF j.dameNumerical.errorMargin !== null»

	«j.dameNumerical.prefijo»«j.dameNumerical.respuesta»:«j.dameNumerical.
errorMargin»
					«ELSEIF j.dameNumerical.maxValue !== null»

	«j.dameNumerical.prefijo»«j.dameNumerical.respuesta»..«j.dameNumerical
.maxValue»
					«ELSE»
```

```
      «j.dameNumerical.prefijo»«j.dameNumerical.respuesta»
                              «ENDIF»
                        «ELSE»
                              «j.damePar.prefijo»«j.damePar.respuesta»-
>«j.damePar.suPareja»
                        «ENDIF»««««//será par lo de arriba
                        «ENDFOR»
                        «ENDIF»}

                  «ELSEIF e.damePregunta instanceof MissingWord»
                        «IF e.damePregunta.getRespuestas !== null»
                        «FOR j:e.damePregunta.getRespuestas»««««una Missing
Word solo tendrá respuestas Respuesta_Estandar
                        «IF j instanceof Respuesta_Estandar»
                              «IF j.dameRespEstandar.feedback !== null»
                                    «IF j.dameRespEstandar.puntuacion
!==null»

      «j.dameRespEstandar.prefijo»%«j.dameRespEstandar.puntuacion»%«j.dameRe
spEstandar.respuesta»#«j.dameRespEstandar.feedback»
                                    «ELSE»

      «j.dameRespEstandar.prefijo»«j.dameRespEstandar.respuesta»#«j.dameResp
Estandar.feedback»
                                    «ENDIF»
                              «ELSE»
                                    «IF j.dameRespEstandar.puntuacion
!==null»

      «j.dameRespEstandar.prefijo»%«j.dameRespEstandar.puntuacion»%«j.dameRe
spEstandar.respuesta»
                                    «ELSE»

      «j.dameRespEstandar.prefijo»«j.dameRespEstandar.respuesta»
                                    «ENDIF»
                              «ENDIF»
                        «ENDIF»
                        «ENDFOR»} «e.damePregunta.dameMissing.parteDos»

                        «ENDIF»
                  «ELSE»««««//será true false, no tiene respuestas
                        «IF e.damePregunta.dameTF.feedback1 !== null»
                              «IF e.damePregunta.dameTF.feedback2 !==null»

      «e.damePregunta.dameTF.respuesta»#«e.damePregunta.dameTF.feedback1».#«
e.damePregunta.dameTF.feedback2»}

                              «ELSE»

      «e.damePregunta.dameTF.respuesta»#«e.damePregunta.dameTF.feedback1»}

                              «ENDIF»
                        «ELSE»
                        «e.damePregunta.dameTF.respuesta»}
```

```
                        «ENDIF»
                    «ENDIF»
                «ENDIF»
        «ENDFOR»
        '''
        //filtrado para Test Part
        def Comentario dameComentario(TestPart tp) {
                return tp as Comentario;
        }
        def Descripcion dameDescripcion(TestPart tp) {
                return tp as Descripcion;
        }
        def Pregunta damePregunta(TestPart tp) {
                return tp as Pregunta;
        }
        //filtro para tipo Pregunta
        def TrueFalse dameTF (Pregunta p){
                return p as TrueFalse
        }
        def MissingWord dameMissing (Pregunta p){
                return p as MissingWord
        }
        def MultipleChoice_Matching_ShortAnswer_Numerical_Essay dameVarios
(Pregunta p){
                return p as MultipleChoice_Matching_ShortAnswer_Numerical_Essay
        }

        //filtro para tipo Respuesta
        def Respuesta_Estandar dameRespEstandar(Respuesta res) {
                return res as Respuesta_Estandar;
        }
        def Par damePar(Respuesta res) {
                return res as Par;
        }
        def Numerical dameNumerical(Respuesta res) {
                return res as Numerical;
        }

}
```

Figura 77. MGiftGenerator.xtend

Se muestra a continuación una captura de pantalla donde un test ha sido validado correctamente y se genera el fichero de extensión txt con la misma información. Se cierra así el ciclo de transformación Text2Model y Model2Text.

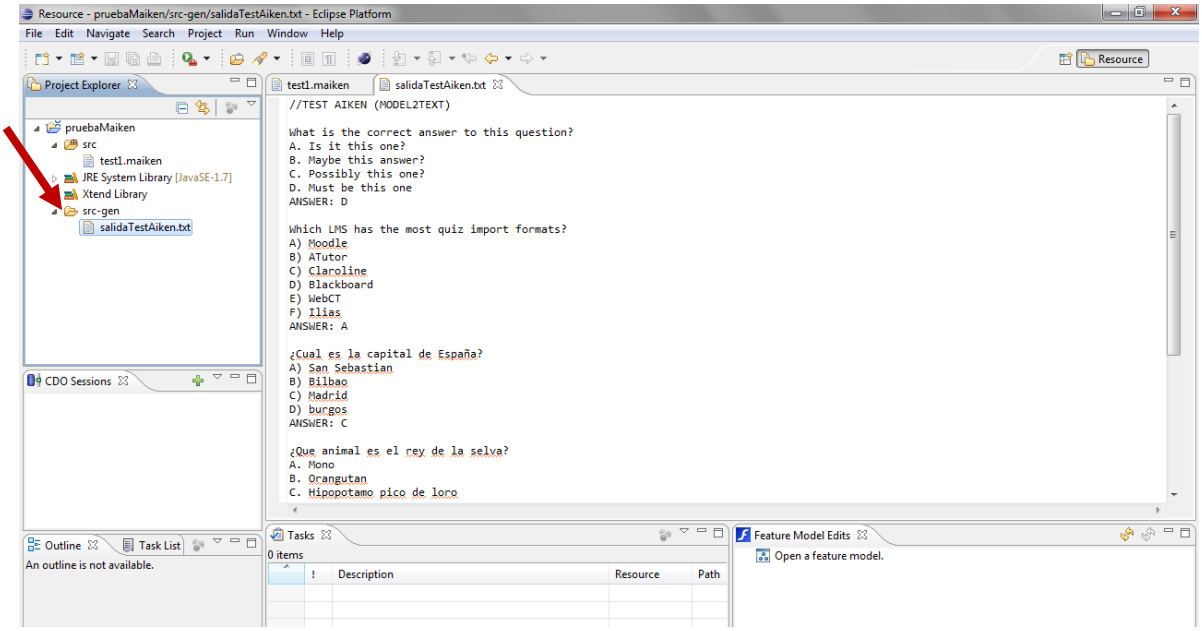


Figura 35. Validar Test Aiken / Gift

2.5.5. Desarrollo del CU Crear Test Aiken/Gift

Caso de uso basado en los mismos elementos utilizados por el caso de uso anterior. Utiliza las mismas gramáticas para, mientras el usuario va creando el test, validarlo y modelarlo en segundo plano. Así mismo el editor de desarrollo va guiando y sugiriendo al usuario valores a introducir que cumplen con el estándar. Si el test es creado correctamente se valida y se genera el fichero txt correspondiente en la carpeta src-gen que contiene la transformación Model2Text proveniente del modelo cargado en segundo plano gracias a la transformación Text2Model obtenida por la gramática y su metamodelo asociado.

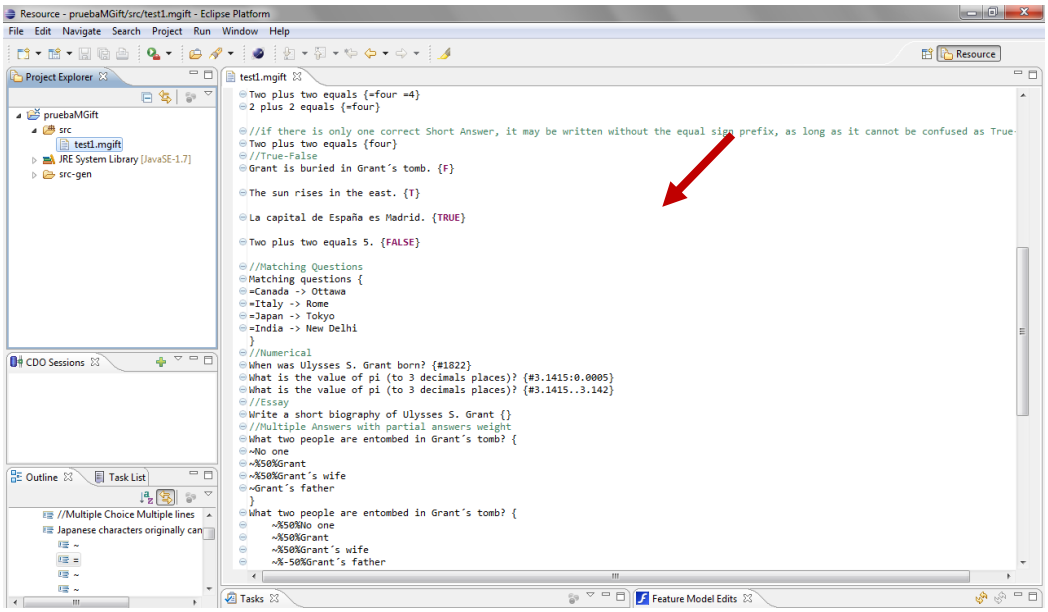


Figura 36. Crear Test Aiken / Gift

2.5.6. Desarrollo del CU Modelar Test Aiken/Gift

Este caso de uso permite al usuario crear un modelo conforme al metamodelo MAiken o MGift, indicando como entrada u origen el test que quiere modelar, colocándolo previamente en la carpeta inputTest.

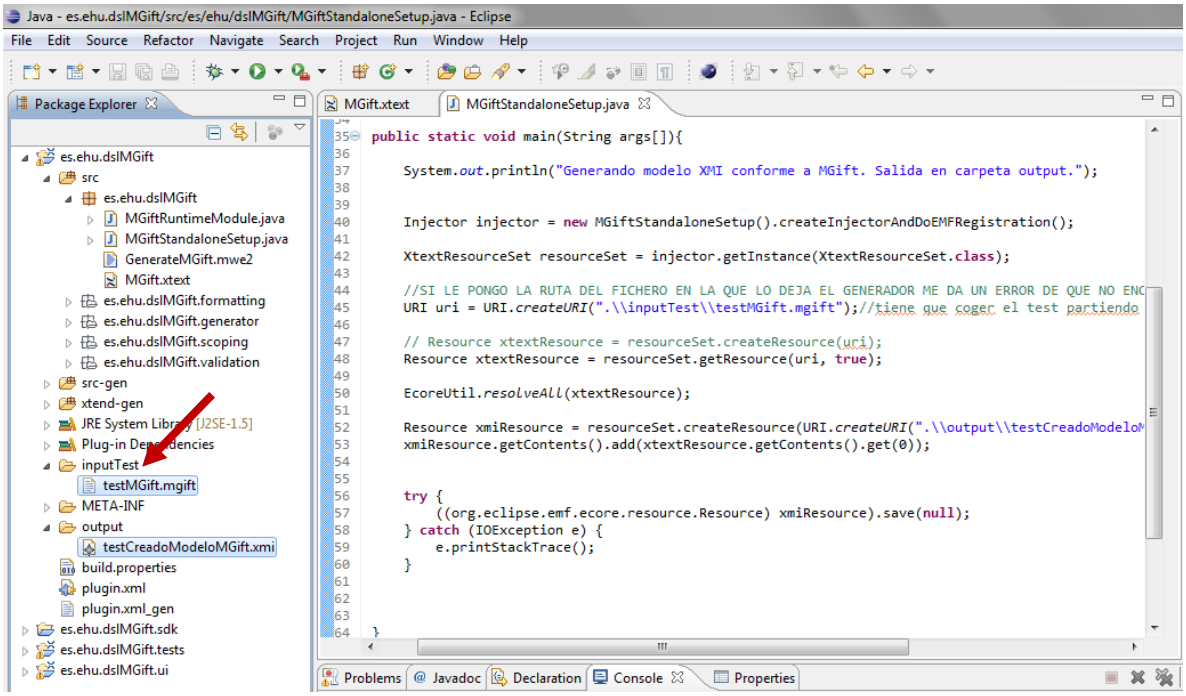


Figura 38. Modelar Test Aiken / Gift

Una vez colocado, se lanzaría la aplicación (MAikenStandaloneSetup.java o MGiftStandaloneSetup.java). Ver figura 78.

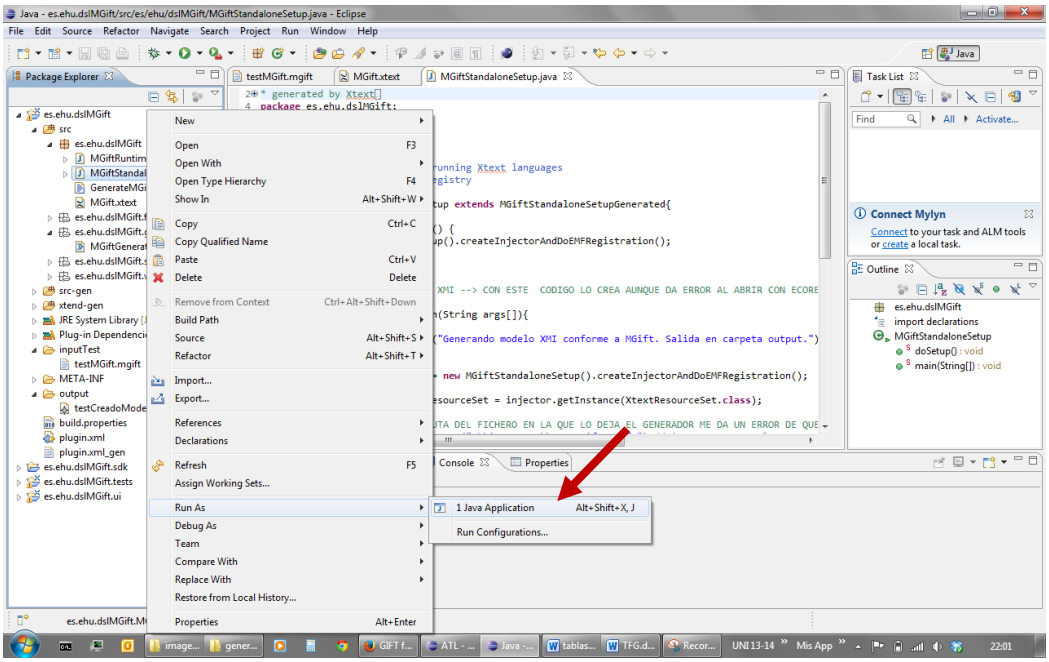


Figura 78. StandaloneSetup.java

Lanzada la aplicación se generaría de forma automática el modelo conforme al metamodelo MAiken o MTest, según el proyecto que hayamos utilizado.

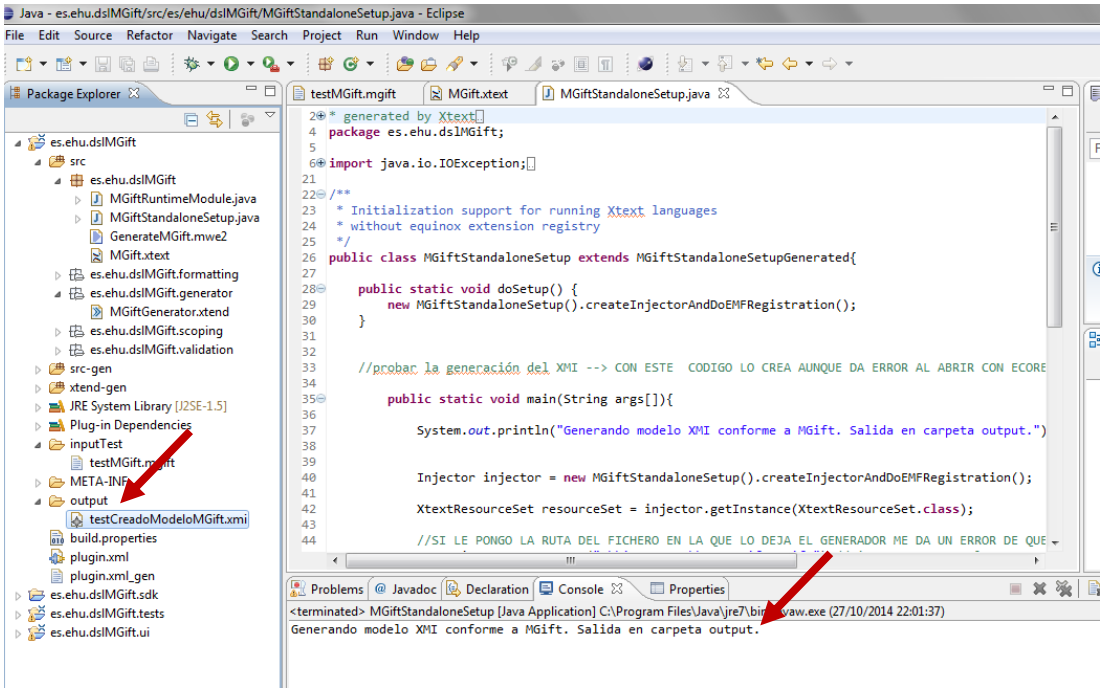


Figura 79. Modelar test

El siguiente código es el encargado de generar el fichero xmi.

```
package es.ehu.dslMGift;

import java.io.IOException;

/**
 * Initialization support for running Xtext languages
 * without equinox extension registry
 */
public class MGiftStandaloneSetup extends MGiftStandaloneSetupGenerated{

    public static void doSetup() {
        new MGiftStandaloneSetup().createInjectorAndDoEMFRegistration();
    }


        public static void main(String args[]){

            System.out.println("Generando modelo XMI conforme a MGift. Salida en carpeta output.");

            Injector injector = new MGiftStandaloneSetup().createInjectorAndDoEMFRegistration();
            XtextResourceSet resourceSet = injector.getInstance(XtextResourceSet.class);

            URI uri = URI.createURI(".\\inputTest\\testMGift.mgift");

            Resource xtextResource = resourceSet.getResource(uri, true);

            EcoreUtil.resolveAll(xtextResource);

            Resource xmiResource = resourceSet.createResource(URI.createURI(".\\output\\testCreadoModeloMGift.xmi"));
            xmiResource.getContents().add(xtextResource.getContents().get(0));

            try {
                ((org.eclipse.emf.ecore.resource.Resource) xmiResource).save(null);
            } catch (IOException e) {
                e.printStackTrace();
            }

        }

}
```

Figura 80. Código generador de xmi

2.5.7. Desarrollo de CU Transformar a modelo MTest

El siguiente caso de uso ha sido desarrollado mediante el asistente para la creación de un proyecto ATL. En él se organiza la siguiente estructura de carpetas: metamodel, model, output y rule. Ver figura 40. La carpeta metamodel contiene lo metamodelos para los cuales las transformaciones son ejecutadas. La carpeta model contendrá los modelos xmi de origen para las transformaciones. En la carpeta output se generarán los modelos xmi destino al ejecutarse una regla de transformación. La carpeta rule contiene las reglas de transformación oportunas.

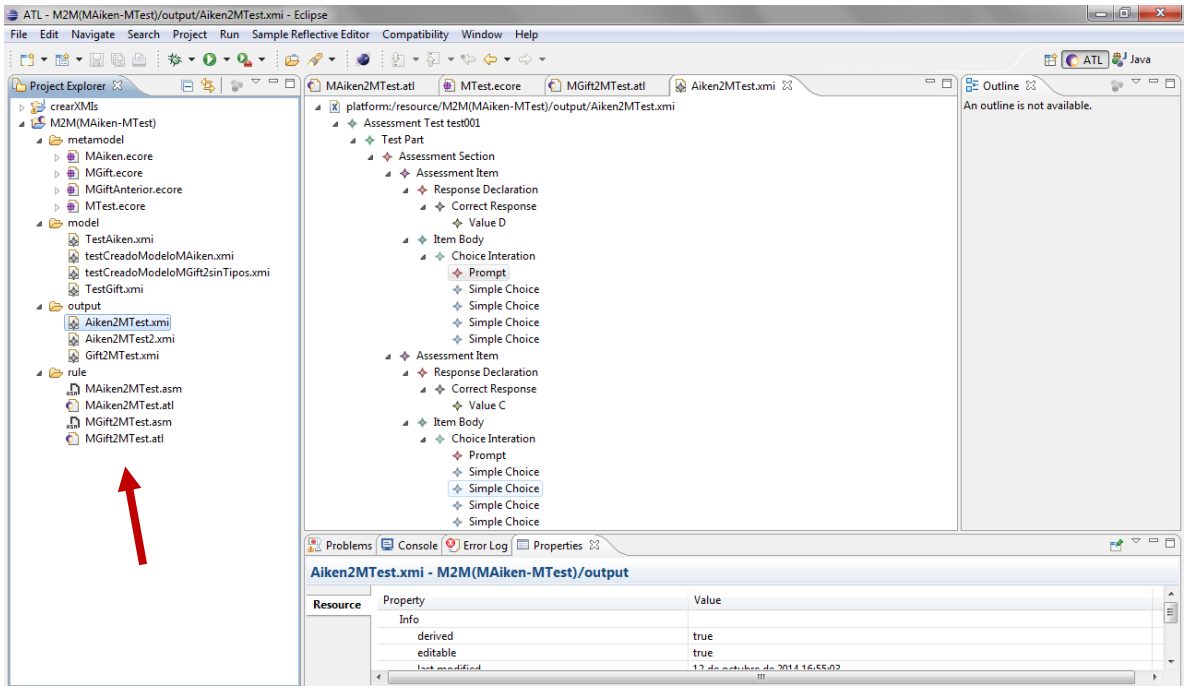


Figura 40. Transformar a modelo MTest

El usuario introduce el modelo de test origen en la carpeta model y configura la regla de transformación correspondiente, (“Run Configurations”) indicando cuál es el modelo origen y como se llamará el modelo destino. Ver figura 39.

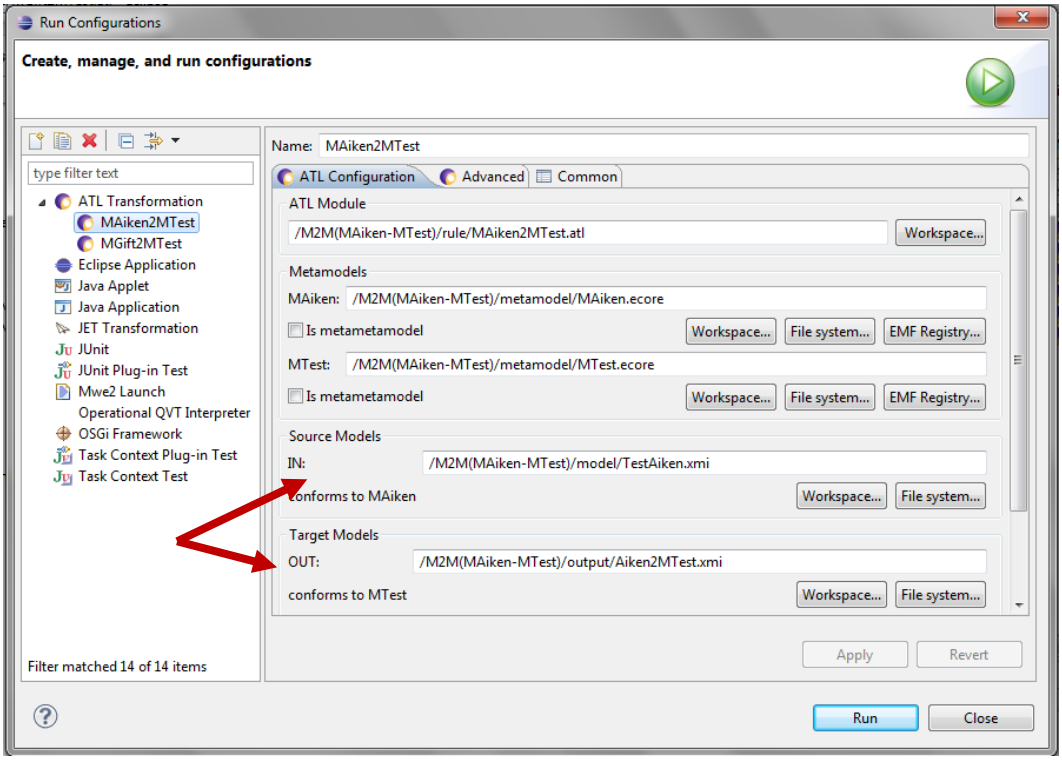


Figura 39. Transformar a modelo MTest

Una vez realizada la configuración se ejecuta la regla de transformación (ver Figura 81) y se genera de forma automática en la carpeta output el modelo generado conforme al metamodelo MTest.

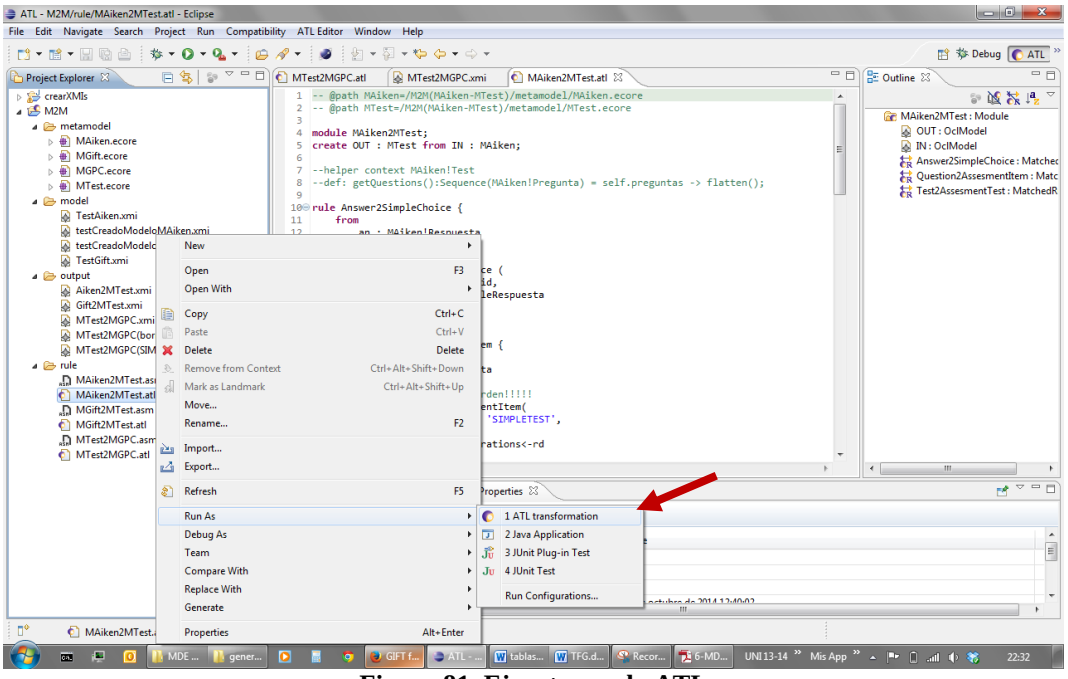


Figura 81. Ejecutar regla ATL

Este Caso de Uso comprende dos reglas de transformación MAiken2MTest.atl y MGift2MTest.atl. Estas reglas de transformación generan modelos destino acordes a un metamodelo específico, gracias a un modelo origen. Por ejemplo la regla MAiken2MTest.atl, transforma la información que recoge de un modelo acorde al metamodelo MAiken en un modelo acorde al metamodelo MTest. En concreto esta regla de transformación se compone de tres reglas, que se encargan de hacer el mapeo de elementos de un modelo origen en otros elementos de un modelo destino, la regla Test2AssementTest indica cómo se convierte un elemento Test del modelo conforme a MAiken en un elemento de tipo AssesmentTest del modelo conforme a MTest. Lo mismo ocurre con las reglas Question2AssesmentItem y Answer2SimpleChoice.

```
-- @path MAiken=/M2M(MAiken-MTest)/metamodel/MAiken.ecore
-- @path MTest=/M2M(MAiken-MTest)/metamodel/MTest.ecore

module MAiken2MTest;
create OUT : MTest from IN : MAiken;

rule Answer2SimpleChoice {
	from
		an : MAiken!Respuesta
	to

		si : MTest!SimpleChoice (
			identifier <- an.id,
			label <- an.posibleRespuesta
		)
}

    rule Question2AssesmentItem {
        from
            t : MAiken!Pregunta
        to
                    ai: MTest!AssessmentItem(
                          identifier <- 'SIMPLETEST',
                    itemBody<-ib,
                    responseDeclarations<-rd
                    ),

                    ci: MTest!ChoiceInteration(
                          prompts <- pr,
                          simpleChoices <- t.respuestasPosibles --

                    ),

                    pr : MTest!Prompt (
                    label <- t.pregunta
                    ),

                    ib: MTest!ItemBody(
                          blocks<-ci

                    ),
```

```
            rd: MTest!ResponseDeclaration(
                  correctResponse<-cr
                  ),
            cr: MTest!CorrectResponse(
                  values<-v
                  ),
            v: MTest!Value(
                  fieldIdentifier <- t.respuestaCorrecta
                  )

    }
    rule Test2AssesmentTest {
        from
            t : MAiken!Test
        to

             a : MTest!AssessmentTest (

                identifier <- t.identificador,
                testPart <- tp
            ),

            as : MTest!AssessmentSection (
                title <- '',
                visible <- true,
                keepTogether <- true,
                sectionParts <- t.preguntas
            ),

            tp :  MTest!TestPart (
                identifier <- 'SIMPLETEST',
                navigationMode <- #linear,
                submissionMode <- #individual,
                assessmentSections <- as
            )
    }
```

Figura 82. Maiken2MTest.atl

La regla MGift2MTest.atl, transforma la información que recoge de un modelo acorde al metamodelo MGift en un modelo acorde al metamodelo MTest. Esta regla es bastante más compleja que la anterior y se compone de treinta reglas de transformación para transformar y filtrar correctamente todos los elementos. Ver figura 83. Se usan condiciones en las clausulas from de cada regla para filtrar los elementos a tratar.

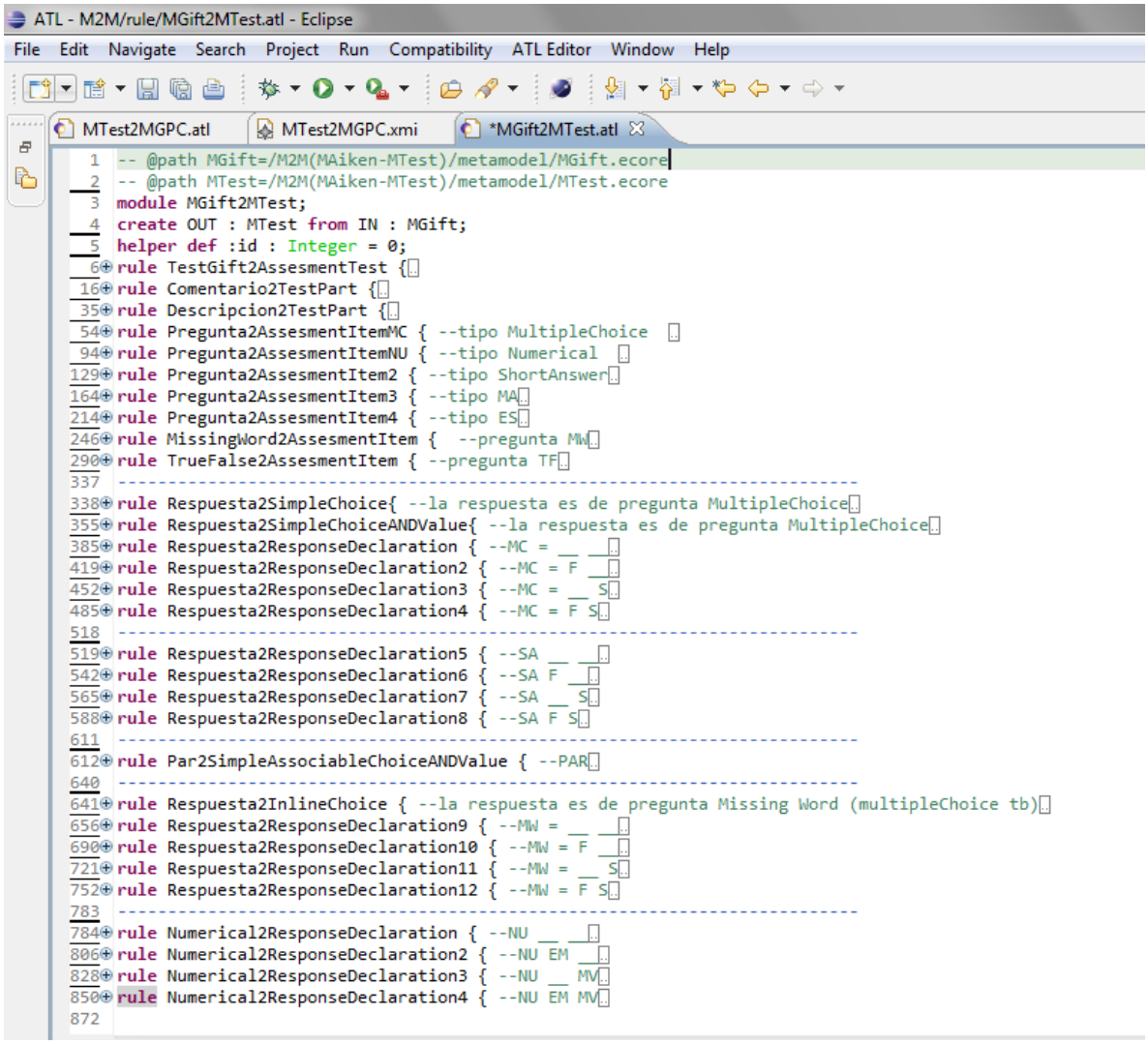


Figura 83. MGift2MTest.atl

A destacar en estas reglas la forma de anidar los elementos. En el fragmento de código que se muestra a continuación se encuentra la regla que transforma una Pregunta en un elemento AssesmentItem, haciendo un filtrado y solo recogiendo aquellos que cumplen la restricción (Questiontype = 'MC'). Se ve cómo se crea un elemento TestPart que lleva anidado un elemento AssesmentSection, que a su vez lleva anidado un elemento AssesmentItem. Este contendrá anidadas tantas responseDeclarations como respuestasPosibles tenga el elemento p, y además para la regla que transforme el tipo de respuestas en otro elemento, este o estos quedarán también bien anidados.

```
rule Pregunta2AssesmentItemMC { --tipo MultipleChoice
	from
		p : MGift!MultipleChoice_Matching_ShortAnswer_Numerical_Essay
		(p.Questiontype = 'MC')
	to
		tp : MTest!TestPart (
			identifier <- 'PREGUNTA',
			assessmentSections <- as
```

```
		),
		as : MTest!AssessmentSection (
			sectionParts <- ai
		),
		ai : MTest!AssessmentItem (
			identifier <- 'MC',
			itemBody <- ib,
			responseDeclarations <- p.respuestasPosibles,
			responseDeclarations <- p.respuestasPosibles->
			collect(e|thisModule.resolveTemp(e,'responseDeclaration'))
		),
		ib : MTest!ItemBody (
			blocks <- ci
		),
		ci : MTest!ChoiceInteration (
			prompts <- pr,
			simpleChoices <- p.respuestasPosibles
		),
		pr : MTest!Prompt (
			label <- p.pregunta
		)
}
```

También es interesante mencionar como gracias al uso de un helper, se genera de forma dinámica un identificador en forma de número, el cual se va asignando a las preguntas con respuesta múltiple, y a su vez y sólo para la respuesta correcta se le introduce también el valor como método para guardar cual es de las respuestas la correcta. En la cláusula do del siguiente código se puede ver un ejemplo.

```
helper def :id : Integer = 0;

rule Respuesta2SimpleChoiceANDValue{ --la respuesta es de pregunta MultipleChoice
	from
		re : MGift!Respuesta_Estandar (re.Questiontype = 'MC' and re.prefijo =
'~'
			and not re.puntuacion.oclIsUndefined())
	to
		sc : MTest!SimpleChoice (
			label <- re.respuesta
		),
		responseDeclaration : MTest!ResponseDeclaration (
			correctResponse <- cr
		),

		cr : MTest!CorrectResponse (
			values <- va,
			interpretation <- re.puntuacion
		),

		va : MTest!Value (
		)
		do{	thisModule.id <- thisModule.id + 1;
			sc.identifier <- thisModule.id.toString();
			va.fieldIdentifier <- thisModule.id.toString();
			}
}
```

2.5.8. Desarrollo del CU Transformar a modelo MGPC

Al igual que el anterior, el siguiente caso de uso ha sido desarrollado mediante el asistente para la creación de un proyecto ATL, siguiendo la misma estructura de carpetas. Al igual que para el caso de uso Trnasformar a modelo MTest, el usuario introduce el modelo de test origen en la carpeta model y configura la regla de transformación correspondiente, ("Run Configurations") indicando cuál es el modelo origen y como se llamará el modelo destino. Ver figura 41.

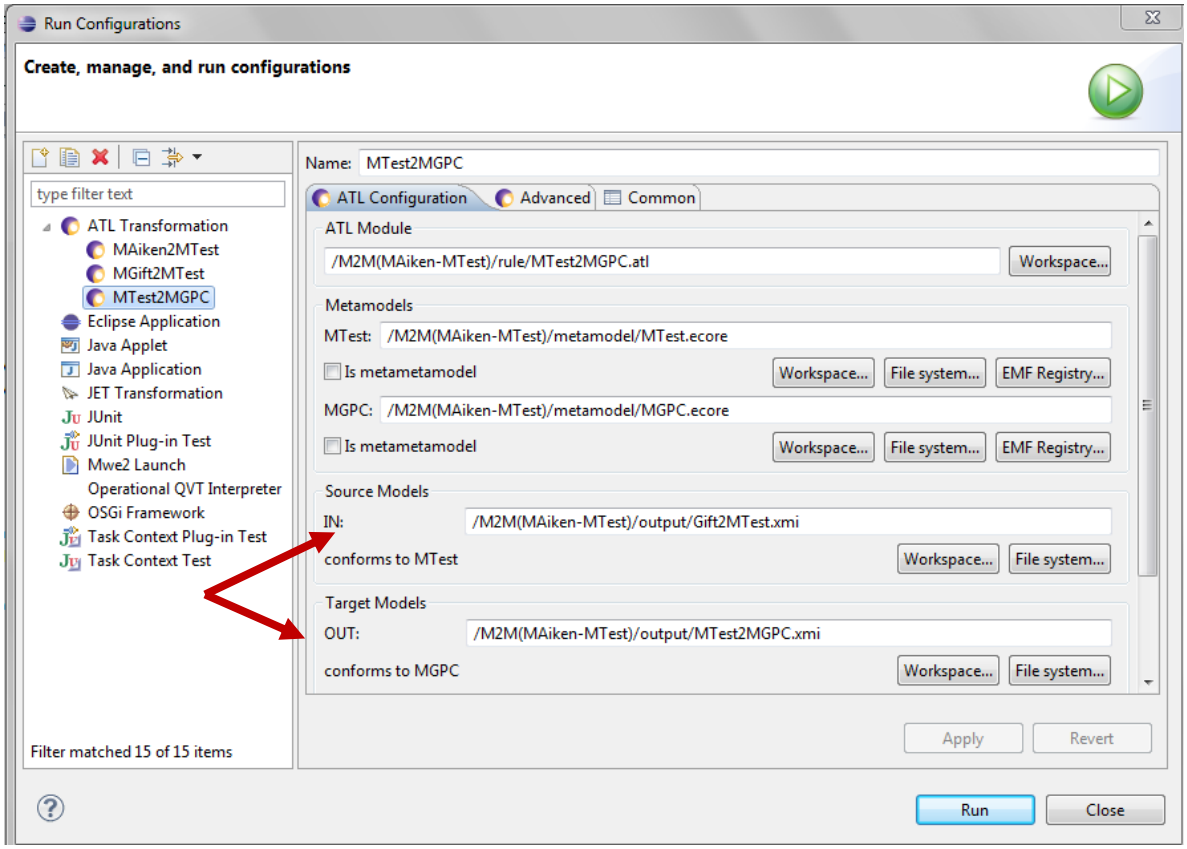


Figura 41. Transformar a modelo MGPC

Una vez realizada la configuración se ejecuta la regla de transformación (ver Figura 81) y se genera de forma automática en la carpeta output el modelo generado conforme al metamodelo MTest.

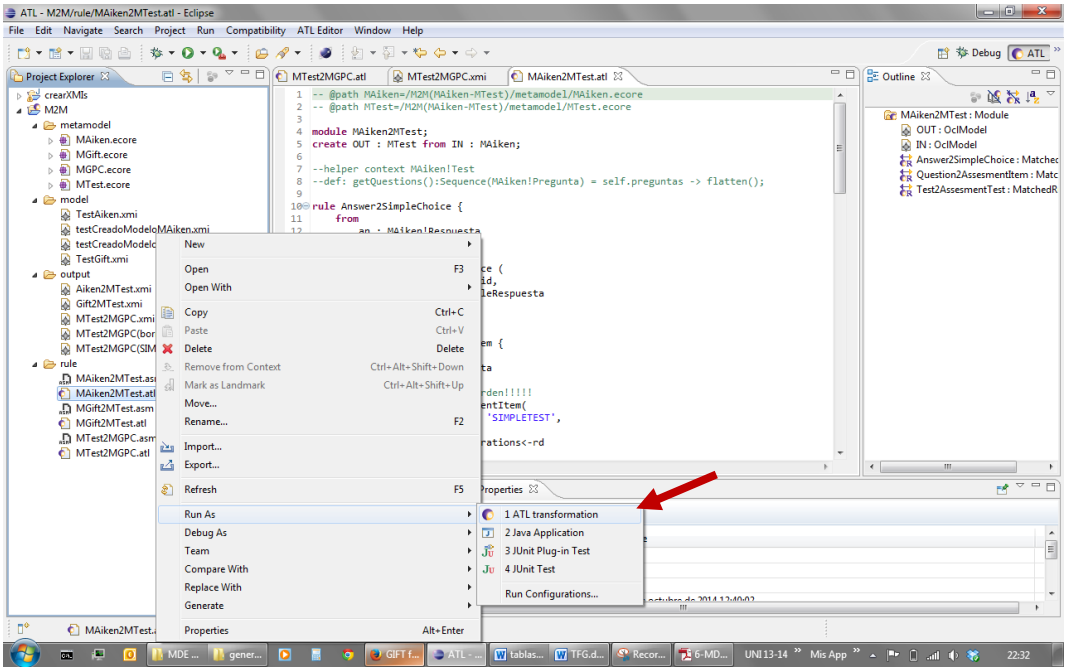


Figura 81. Ejecutar regla ATL

La regla MTest2MGPC.atl, transforma la información que recoge de un modelo acorde al metamodelo MTest en un modelo acorde al metamodelo MGPC. Esta regla se compone de multiples reglas de transformación para transformar y filtrar correctamente todos los elementos. Ver figura 84. Se usan condiciones en las clausulas from de cada regla para filtrar los elementos a tratar. En esta regla de transformación en concreto se han utilizad diversos helpers como ayuda para recorrer los elementos del modelo conforme al metamodelo MTest y recoger exactamente los necesarios, ya clasificados por el tipo de elemento.

```
-- @path MGPC=/M2M(MAiken-MTest)/metamodel/MGPC.ecore
-- @path MTest=/M2M(MAiken-MTest)/metamodel/MTest.ecore

module MTest2MGPC;
create OUT : MGPC from IN : MTest;

helper def :values2param : Boolean = true;
helper def :responseDeclaration2param : Boolean = true;

rule AssesmentTest2CPG {
rule TestPart2Section {
rule AssesmentSection2QuestionForm {
rule AssesmentItem2Question {
rule responseDeclaration2Parameters {
--HELPER QUE DADO UN AssesmentItem TE DEVUELVE EL ÚNICO PROMPT
helper context MTest!AssessmentItem def: getPrompt(): MTest!ChoiceInteration
=
      self.itemBody.blocks -> collect (p | p.prompts) -> first();
--HELPER QUE RECORRA EL ASSESMENTITEM->ITEMBODY->BLOCKS Y QUE COJA EL
--CHOICEINTERACTION
```

```
helper context MTest!AssessmentItem def: getChoiceInteraction():
MTest!ChoiceInteration =self.itemBody.blocks -> first();
rule SimpleChoice2Answer {
rule AssesmentItem2Question2 {--DESCRIPCION O COMENTARIO
rule AssesmentItem2Question3 {--MC
helper context MTest!AssessmentItem def: getBL(): MTest!BlockQuote =
	self.itemBody.blocks -> first ();

helper context MTest!BlockQuote def: getP(): MTest!P =
	self.containsBlock -> first();

helper context MTest!AssessmentItem def: get1Value(): MTest!Value =
	self.responseDeclarations -> first().correctResponse.values ->
first();

rule AssesmentItem2Question4 {--NU

helper context MTest!AssessmentItem def: getPrompt2(): MTest!Prompt =
	self.itemBody.blocks -> collect (p | p.prompts) -> first();

helper context MTest!AssessmentItem def: getAssociateInteraction():
MTest!AssociateInteraction =
	self.itemBody.blocks -> first();

rule AssesmentItem2Question5 {--MA
rule SimpleAssociableChoicesChoice2Answer {
rule Values2Parameter {
rule AssesmentItem2Question6 {--SA

helper context MTest!AssessmentItem def: getProm(): MTest!Prompt =
self.itemBody.blocks -> collect (p | p.prompts) -> first();
rule AssesmentItem2Question7 {--ES
rule AssesmentItem2Question8 {--MW
helper context MTest!BlockQuote def: getPss(): Sequence (MTest!P) =
self.containsBlock;

helper context MTest!BlockQuote def: getInlineChoiceInteraction():
MTest!InlineChoiceInteraction =
self.containsBlock -> select (p |
p.oclIsTypeOf(MTest!InlineChoiceInteraction)) -> first();

rule InlinceChoice2Answer {
rule AssesmentItem2Question9 {--TF
```

Figura 84. MTest2MGPC.atl

A destacar del código el siguiente fragmento, algo diferente a las demás reglas ya que en esta se tuvieron que usar variables locales en la clausula using, como paso intermedio para obtener valores gracias a los helpers getBL() y getPss().

```
rule AssesmentItem2Question8 {--MW
	from
		ai : MTest!AssessmentItem (ai.identifier = 'MW')

	using {
```

```
                    Ppart1 : String = ai.getBL().getPss() -> first().label;
                    Ppart2: String = ai.getBL().getPss() -> last().label;
                    --definición de variables locales;
        }
        to
                qu : MGPC!Question (
                    name <- ai.identifier,
                    question <- Ppart1 + ' _____ ' + Ppart2,
                    possibleAnswers <-
ai.getBL().getInlineChoiceInteraction().inlineChoices,
                    parameters <- ai.responseDeclarations
                )
                do {
                    thisModule.responseDeclaration2param = true;
                    thisModule.values2param = false;
                }
}
```

A continuación, en la figura 85 se puede ver una captura de pantalla de la generación de un modelo conforme a MGPC de manera automática. A la izquierda de la figura se ve modelado un test simple con un único tipo de preguntas, mientras que a la derecha se observa un modelo el cual dispone de diferentes secciones y tipos de preguntas en sus nodos QuestionForm.

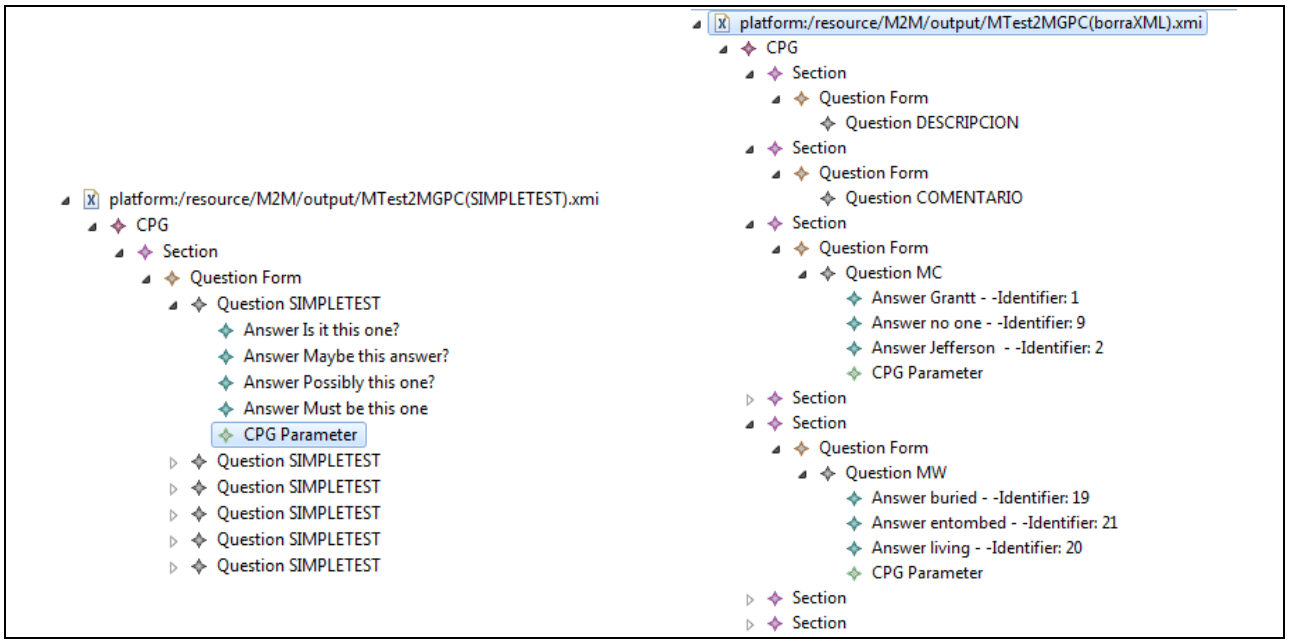


Figura 85. Aspecto de modelos generados conformes a MGPC

Además de con la vista anterior proporcionada por el IDE de desarrollo, también se pueden ver estos modelos como una vista de texto plano en XML. Ya que al fin y al cabo un fichero xmi es un fichero XML Metadata Interchange y por tanto se puede ver como un fichero XML. Ver figura 86. Se han resaltado algunos elementos para de un primer vistazo tener una visión más efectiva de su modelado.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<xmi:XMI xmi:version="2.0" xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gpc="http://erabaki.ehu.es/gpc">
  <gpc:CPG>
    <sectionList description="DESCRIPCION">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="Descripcion de la primera pregunta"
name="DESCRIPCION"/>
      </nodeList>
    </sectionList>
    <sectionList description="COMENTARIO">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="//MULTIPLE CHOICE TYPE" name="COMENTARIO"/>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="Who is buried in Grant´s tomb?" name="MC">
          <possibleAnswers possibleAnswer="Grantt - -Identifier: 1"/>
          <possibleAnswers possibleAnswer="no one - -Identifier: 9"/>
          <possibleAnswers possibleAnswer="Jefferson  - -Identifier: 2"/>
          <parameters defaultValue="9" description="CorrectResponse"/>
        </questionList>
      </nodeList>
    </sectionList>
    <sectionList description="COMENTARIO">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="//MISSING WORD TYPE" name="COMENTARIO"/>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="Grant is _____ in Grant´s tomb" name="MW">
          <possibleAnswers possibleAnswer="buried - -Identifier: 19"/>
          <possibleAnswers possibleAnswer="entombed - -Identifier: 21"/>
          <possibleAnswers possibleAnswer="living - -Identifier: 20"/>
          <parameters defaultValue="21" description="CorrectResponse"/>
        </questionList>
      </nodeList>
    </sectionList>
    <sectionList description="COMENTARIO">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="//SHORT ANSWER TYPE" name="COMENTARIO"/>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="Two plus two equals" name="SA">
          <possibleAnswers xsi:type="gpc:CPGParameter" defaultValue="four"
description="CorrectResponse"/>
          <possibleAnswers xsi:type="gpc:CPGParameter" defaultValue="4"
description="CorrectResponse"/>
        </questionList>
      </nodeList>
    </sectionList>
```

```
    <sectionList description="COMENTARIO">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="//TRUE FALSE TYPE" name="COMENTARIO"/>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="Grant is buried in Grant´s tomb"
defaultAnswer="/0/@sectionList.8/@nodeList.0/@questionList.0/@possibleAnswers
.0" name="TF">
          <possibleAnswers possibleAnswer="F"/>
          <parameters name="F1: OclUndefined F2: OclUndefined"
description="Interpretation"/>
        </questionList>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="La capital de España es Madrid"
defaultAnswer="/0/@sectionList.9/@nodeList.0/@questionList.0/@possibleAnswers
.0" name="TF">
          <possibleAnswers possibleAnswer="TRUE"/>
          <parameters name="F1: OclUndefined F2: OclUndefined"
description="Interpretation"/>
        </questionList>
      </nodeList>
    </sectionList>
    <sectionList description="COMENTARIO">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="//MATCHING TYPE" name="COMENTARIO"/>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="Match each country with its capital"
name="MA">
          <possibleAnswers possibleAnswer="Canada - -Identifier: 11"/>
          <possibleAnswers possibleAnswer="India - -Identifier: 13"/>
          <possibleAnswers possibleAnswer="Italy - -Identifier: 15"/>
          <possibleAnswers possibleAnswer="Japan - -Identifier: 17"/>
          <possibleAnswers possibleAnswer="Ottawa - -Identifier: 12"/>
          <possibleAnswers possibleAnswer="New Delhi - -Identifier: 14"/>
          <possibleAnswers possibleAnswer="Rome - -Identifier: 16"/>
          <possibleAnswers possibleAnswer="Tokyo - -Identifier: 18"/>
          <parameters defaultValue="11  12"
description="CorrectResponsePairs"/>
          <parameters defaultValue="13  14"
description="CorrectResponsePairs"/>
          <parameters defaultValue="15  16"
description="CorrectResponsePairs"/>
          <parameters defaultValue="17  18"
description="CorrectResponsePairs"/>
        </questionList>
      </nodeList>
    </sectionList>
    <sectionList description="COMENTARIO">
```

```
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="//NUMERICAL TYPE" name="COMENTARIO"/>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="When was Ulysses S. Grant born?"
defaultAnswer="/0/@sectionList.13/@nodeList.0/@questionList.0/@possibleAnswer
s.0" name="NU">
          <possibleAnswers possibleAnswer="1822"/>
          <parameters name="" description="Interpretation"/>
        </questionList>
      </nodeList>
    </sectionList>
    <sectionList description="COMENTARIO">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="//NUMERICAL WITH ERROR MARGIN"
name="COMENTARIO"/>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="What is the value of pi (to 3 decimals
places)?"
defaultAnswer="/0/@sectionList.15/@nodeList.0/@questionList.0/@possibleAnswer
s.0" name="NU">
          <possibleAnswers possibleAnswer="3.1415"/>
          <parameters name="0.0005" description="Interpretation"/>
        </questionList>
      </nodeList>
    </sectionList>
    <sectionList description="COMENTARIO">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="//NUMERICAL UNTIL MAX VALUE PERMITTED"
name="COMENTARIO"/>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="What is the value of pi (to 3 decimals
places)?"
defaultAnswer="/0/@sectionList.17/@nodeList.0/@questionList.0/@possibleAnswer
s.0" name="NU">
          <possibleAnswers possibleAnswer="3.1415"/>
          <parameters name="3.142" description="Interpretation"/>
        </questionList>
      </nodeList>
    </sectionList>
    <sectionList description="COMENTARIO">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="//ESSAY TYPE" name="COMENTARIO"/>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
```

```
        <questionList question="Write a short biography of Ulysses S. Grant"
name="ES"/>
      </nodeList>
    </sectionList>
    <sectionList description="COMENTARIO">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="//MULTIPLE CHOICE WITH PARTIAL ANSWERS
WEIGHT" name="COMENTARIO"/>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="What two people are entombed in Grant´s
tomb?" name="MC">
          <possibleAnswers possibleAnswer="no one - -Identifier: 3"/>
          <possibleAnswers possibleAnswer="Grand´s mother - -Identifier: 4"/>
          <possibleAnswers possibleAnswer="Grant - -Identifier: 7"/>
          <possibleAnswers possibleAnswer="Grant´s father - -Identifier: 8"/>
          <parameters defaultValue="7" description="CorrectResponse"/>
          <parameters defaultValue="8" description="CorrectResponse"/>
        </questionList>
      </nodeList>
    </sectionList>
    <sectionList description="COMENTARIO">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="//WITH FEEDBACKS" name="COMENTARIO"/>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="What´s the answer to this multiple choice
question?" name="MC">
          <possibleAnswers possibleAnswer="wrong answer - -Identifier: 5"/>
          <possibleAnswers possibleAnswer="another wrong answer - -
Identifier: 6"/>
          <possibleAnswers possibleAnswer="right answer - -Identifier: 10"/>
          <parameters defaultValue="10" description="CorrectResponse"/>
        </questionList>
      </nodeList>
    </sectionList>
    <sectionList description="COMENTARIO">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="//SPECIAL FEEDBACK FOR TRUE FALSE TYPE"
name="COMENTARIO"/>
      </nodeList>
    </sectionList>
    <sectionList description="PREGUNTA">
      <nodeList xsi:type="gpc:QuestionForm">
        <questionList question="Grant is buried in Grant´s tomb."
defaultAnswer="/0/@sectionList.25/@nodeList.0/@questionList.0/@possibleAnswer
s.0" name="TF">
          <possibleAnswers possibleAnswer="FALSE"/>
          <parameters name="F1: Wrong, No one is buried in Grant´s tomb. F2:
Right, well done" description="Interpretation"/>
        </questionList>
      </nodeList>
```

```
    </sectionList>
  </gpc:CPG>
</xmi:XMI>
```

Figura 86. Aspecto de modelo generado conforme a MGPC en vista XML

2.6. Pruebas

A lo largo de toda la implementación del proyecto se han ido realizando pruebas para poder comprobar el correcto funcionamiento de la aplicación y obtener los resultados satisfactorios deseados. Se han recopilado todas las pruebas, tanto unitarias como de aceptación, en la tabla que a continuación se muestra. Se han agrupado en los diferentes casos de uso, con el fin de tratar de mantener un orden adecuado y exponer el mayor número de ellas posibles. Por cada prueba realizada se dirá si ha sido satisfactoria o no, explicando las razones en caso de que no lo fuera.

Tabla 9. Pruebas C.U. Validar Test Aiken/Gift

C.U. Validar Test Aiken/Gift			
Descripción de la prueba	Resultado obtenido	Resultado esperado	Solución
Validar un test acorde al estándar Aiken o Gift	No se pasa la validación	NO	Revisión de la gramática o DSL y corrección de la misma
Validar un test acorde al estándar Aiken o Gift	Se pasa la validación	OK	--------
Validar un test no acorde al estándar Aiken o Gift	No pasa la validación y muestra los errores en el editor	OK	--------

Tabla 10. Pruebas C.U. Crear Test Aiken/Gift

C.U. Crear Test Aiken/Gift			
Descripción de la prueba	Resultado obtenido	Resultado esperado	Solución
Crear un test acorde al estándar Aiken o Gift	El editor no muestra bien las ayudas y no reconoce valores	NO	Revisión de la gramática o DSL y corrección de la misma
Crear un test acorde al estándar Aiken o Gift	El editor muestra bien las ayudas y reconoce todas las combinaciones posibles. Genera en la salida el fichero per no lo muestra bien.	NO	Revisión de la plantila de transformación XTend Model2Text
Crear un test no acorde al estándar Aiken o Gift	El editor muestra los errores y no deja validar ni crear el tes	OK	--------

Tabla 11. Pruebas C.U. Modelar Test Aiken/Gift

C.U. Modelar Test Aiken/Gift			
Descripción de la prueba	Resultado obtenido	Resultado esperado	Solución
Dado un fichero de texto con un test en formato Aiken o Gift generar el modelo	La aplicación no reconoce el fichero de entrada.	NO	La extensión del fichero de entrada no es de tipo txt sino que será .maiken o .mgift
Dado un fichero de texto con un test en formato Aiken o Gift generar el modelo	La aplicación genera el modelo pero no se visualiza en modo xmi	NO	Es debido a que el proyecto en el que está alojado no permite registrar un metamodelo asociado al xmi.
Dado un fichero de texto con un test en formato Aiken o Gift generar el modelo	Se genera el modelo y muestra el mensaje el pantalla. Se copia el modelo y se visualiza desde el proyecto M2M	OK	--------

Tabla 12. Pruebas C.U. Transformar a modelo MTest

C.U. Transformar a modelo MTest			
Descripción de la prueba	Resultado obtenido	Resultado esperado	Solución
Dado un modelo acorde al metamodelo MAiken generar un modelo conforme a MTest	Errores y conflictos en las reglas	NO	Revisión de la regla de transformación y mejor filtrado de elementos para evitar colisiones
Dado un modelo acorde al metamodelo MAiken generar un modelo conforme a MTest.	No errores pero el modelo generado no queda bien anidado	NO	Revision de la regla de transformación, orden, lazy rules, funciones resolveTemp( ) y anidaciones en las mismas
Dado un modelo acorde al metamodelo MGift generar un modelo conforme a MTest	Errores y conflictos en las reglas	NO	Revisión de la regla de transformación y mejor filtrado de elementos para evitar colisiones
Dado un modelo acorde al metamodelo MGift generar un modelo conforme a MTest.	No errores pero el modelo generado no queda bien anidado	NO	Revision de la regla de transformación, orden, lazy rules, funciones resolveTemp( ) y anidaciones en las mismas
Dado un modelo acorde al metamodelo MGift generar un modelo conforme a MTest	Problemas para diferenciar tipos de preguntas y respuestas	NO	Tras la generación del modelo, asignar de modo manual los tipos a cada una de las preguntas/respuestas del modelo (atributo QuestionType)

Tabla 13. Pruebas C.U. Transformar a modelo MGPC

C.U. Transformar a modelo MGPC			
Descripción de la prueba	Resultado obtenido	Resultado esperado	Solución
Dado un modelo acorde al metamodelo MTest generar un modelo conforme a MGPC	Errores y conflictos en las reglas	NO	Revisión de la regla de transformación y mejor filtrado de elementos para evitar colisiones
Dado un modelo acorde al metamodelo MTest generar un modelo conforme a MGPC.	No errores pero el modelo generado no queda bien anidado	NO	Revision de la regla de transformación, orden, lazy rules, funciones resolveTemp( ) y anidaciones en las mismas
Dado un modelo acorde al metamodelo MTest generar un modelo conforme a MGPC.	No errores pero quedan elementos duplicados fuera de la raíz	NO	Tras revisión de código la mejor opción es una vez generado eliminar de forma manual los elementos apilados fuera de la raiz
Dado un modelo acorde al metamodelo MTest generar un modelo conforme a MGPC.	Según el tipo de preguntas algunas tienes más de una respuesta correcta, por lo que no vale el apuntador defaultAnswer	OK	Se crea un elemento CPGParameter para cada Question con el fin de almacenar los valores correctos
Dado un modelo acorde al metamodelo MTest generar un modelo conforme a MGPC.	En algunos casos, no es posible almacenar automáticamente la relación defaultAnswer para una Pregunta	OK	Tras generar el modelo de forma manual se rellena dicha información desde el editor

2.6.1. Pruebas de aceptación

Para comprobar si la aplicación satisfacía los requerimientos esperados y funcionaba conforme a lo establecido, cumpliendo los objetivos marcados, el 3 de noviembre de 2014 se realizó una prueba de aceptación bajo la supervisión del director. Tal prueba se realizó con total normalidad y fue todo un éxito, ya que el sistema funcionaba acorde con lo establecido. Consistió en simular parte de las pruebas documentadas en el apartado anterior

Para la totalidad de la prueba de aceptación, no surgieron dificultades, por lo que se aprobó el correcto funcionamiento de la aplicación desarrollada.

3. Revisión

Una vez finalizado el proyecto, se han detectado desviaciones de tiempo respecto a la planificación temporal inicial realizada en el apartado Esquema de planificación de tareas (1.6.3.1.) del documento de objetivos de proyecto. Tales incidentes se reflejan en la siguiente tabla:

Tabla 14. Desviación de tiempo duración estimada-real

Tarea	Duración estimada	Duración real
1. Planificación inicial	**15 horas**	**15 horas**
2. Recopilación de información	14 horas	14 horas
3. Reunión con el director	1 hora	1 hora
4. Redacción del documento de viabilidad	**28 horas**	**31 horas**
5. Introducción	1 hora	1 hora
6. Objetivos del proyecto	2 horas	2 horas
7. Descripción general	4 horas	5 horas
8. Arquitectura del proyecto	1 horas	1 horas
9. Herramientas y tecnologías de la aplicación	4 horas	4 horas
10. Estructuración y calendario del sistema	8 horas	9 horas
11. Análisis de riesgos	4 horas	4 horas
12. Evaluación económica	4 horas	5 horas
13. Aprendizaje y recopilación de información	**30 horas**	**40 horas**
14. Implementación aplicación	**180 horas**	**260 horas**
15. Instalación y compatibilidades	3 horas	3 horas
16. Meta modelado de los modelos	22 horas	52 horas
17. Gramáticas asociadas a los formatos	75 horas	110 horas
18. Transformaciones entre modelos	80 horas	95 horas
19. Integración y pruebas	**20 horas**	**25 horas**
20. Documentación	**50 horas**	**70 horas**
21. Redacción de la memoria	50 horas	70 horas
22. Preparación de la presentación	**10 horas**	**10 horas**
Total horas	**333 horas**	**451 horas**

A modo más intuitivo, se ha realizado el siguiente gráfico:

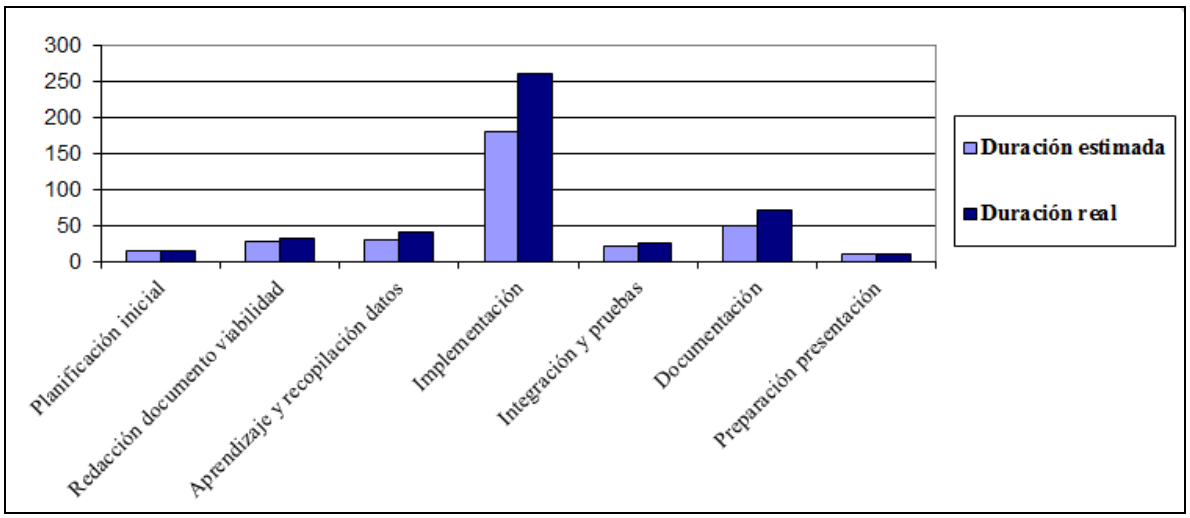


Figura 87. Representación gráfica de la desviación temporal

En resumen, la desviación ha sido de 118 horas y las diferencias más significativas se pueden apreciar en las tareas de implementación y documentación.

La fase de implementación fue más duradera de lo esperado inicialmente, debido a la falta de conocimiento sobre la plataforma de desarrollo y el framework utilizados, la forma de trabajar mediante MDA, lenguajes específicos de modelado, etc. En varias ocasiones, y de forma reiterada, se conocía la solución al problema establecido, sin embargo, se desconocía la forma de abordarlo desde el punto de vista de la tecnología y los lenguajes de programación utilizados en el presente proyecto. Este hecho marcó un considerable aumento del tiempo para finalizar dicha fase.

La tarea de documentación se alargó ya que hubo que buscar más información de la que inicialmente se estimó. La mayoría de ésta información a buscar, (tutoriales, libros artículos y foros), no resultaba provechosa, por lo que la búsqueda se prolongó en el tiempo hasta haber obtenido la información necesaria.

Por último en la tarea de aprendizaje y recopilación de información, también se puede apreciar un aumento notable de tiempo. El hecho de aprender sobre temas de estándares de test, tratamiento de datos y procesamiento de los mismos, para comprender el objetivo del proyecto principal, no fue tan sencillo como parecía. Es por ello que la estimación de esta tarea no fue correcta.

Debido al consiguiente retraso, hubo que posponer la fecha de defensa del proyecto hasta la convocatoria de noviembre 2014. Siendo esta la más próxima a la convocatoria anterior.

Tabla 15. Inicio y finalización del proyecto (real)

Fecha de inicio	Fecha de finalización
5 de mayo de 2014	3 de noviembre de 2014

El proyecto pudo haber quedado finalizado antes de la fecha arriba indicada, sin embargo, debido a causas laborales, y por falta de tiempo, la carga diaria de horas que se estimó para algunas semanas no se cumplió, sino que disminuyó. Alargando así su fecha de finalización y por consiguiente dedicando una menor carga lectiva diaria.

Habiendo aumentado la duración del proyecto en 118 horas, el coste total del proyecto también se vio incrementado. El salario neto, así como los gastos de inmovilizado de material y amortización, se vieron afectados.

Salario Neto → 451 horas * 15 euros/hora = 6765 euros brutos – 6% Cotización SS – 15% Cotización IRPF = 6765 euros – 405,9 euros – 1014,75 euros = **5344,35 euros**

Inmovilizado de material y amortización → (35 euros/mes * 6 meses) + 47,25 euros + 2,40 euros + 1,20 euros = **260,85 euros**

Coste total → = Salario Neto analista/programador + Seguridad Social + IRPF + Inmovilizado de material y amortización + Otros gastos = 5344,35€ + 405,9€ + 1014,75€ + 260,85€ + 80€ = 7105,85 **euros**

Además de estos cambios y revisiones, también cabe destacar las alteraciones ocasionadas en el diseño del proyecto. Las distintas implementaciones de los DSL, así como el diseño de los metamodelos, han ido sufriendo modificaciones a lo largo de su desarrollo, hasta concluir, finalmente, en los presentes mostrados en éste documento.

Para cerrar el apartado de revisión, y tras haber revisado todos los puntos acerca de la planificación temporal de las tareas, se presenta el nuevo diagrama de Gantt, acorde ahora sí, a la planificación temporal real de las tareas del proyecto.

	Nombre de la tarea	Comienzo	Fin	Duración
1	**Planificación inicial**	**lun 05/05/14**	**jue 22/05/14**	**15 hrs**
2	Recopilación de información	lun 05/05/14	jue 22/05/14	14 hrs
3	Reunión con el director	mar 20/05/14	mar 20/05/14	1 hr
4	**Redacción del documento de viabilidad**	**mar 20/05/14**	**lun 02/06/14**	**31 hrs**
5	Introducción	mar 20/05/14	mié 21/05/14	1 hr
6	Objetivos del proyecto	mié 21/05/14	mié 21/05/14	2 hrs
7	Descripción general	jue 22/05/14	dom 25/05/14	5 hrs
8	Arquitectura del proyecto	lun 26/05/14	lun 26/05/14	1 hr
9	Herramientas y tecnologías de la aplicación	lun 26/05/14	mar 27/05/14	4 hrs
10	Estructuración y calendario del sistema	mar 27/05/14	jue 29/05/14	9 hrs
11	Análisis de riesgos	vie 30/05/14	vie 30/05/14	4 hrs
12	Evaluación económica	sáb 31/05/14	lun 02/06/14	5 hrs
13	**Aprendizaje y recopilación de información**	**lun 19/05/14**	**lun 30/06/14**	**40 hrs**
14	**Implementación aplicación**	**mar 27/05/14**	**mar 28/10/14**	**260 hrs**
15	Instalación y compatibilidades	mar 27/05/14	mié 28/05/14	3 hrs
16	Metamodelado de los modelos	lun 02/06/14	lun 06/10/14	52 hrs
17	Gramáticas asociadas a los formatos	mié 04/06/14	lun 15/09/14	110 hrs
18	Transformaciones entre modelos	mié 03/09/14	mar 28/10/14	95 hrs
19	**Integración y pruebas**	**jue 10/07/14**	**mar 28/10/14**	**25 hrs**
20	**Documentación**	**mar 20/05/14**	**jue 30/10/14**	**70 hrs**
21	Redacción de la memoria	mar 20/05/14	jue 30/10/14	70 hrs
22	**Preparación presentación**	**lun 27/10/14**	**lun 03/11/14**	**10 hrs**

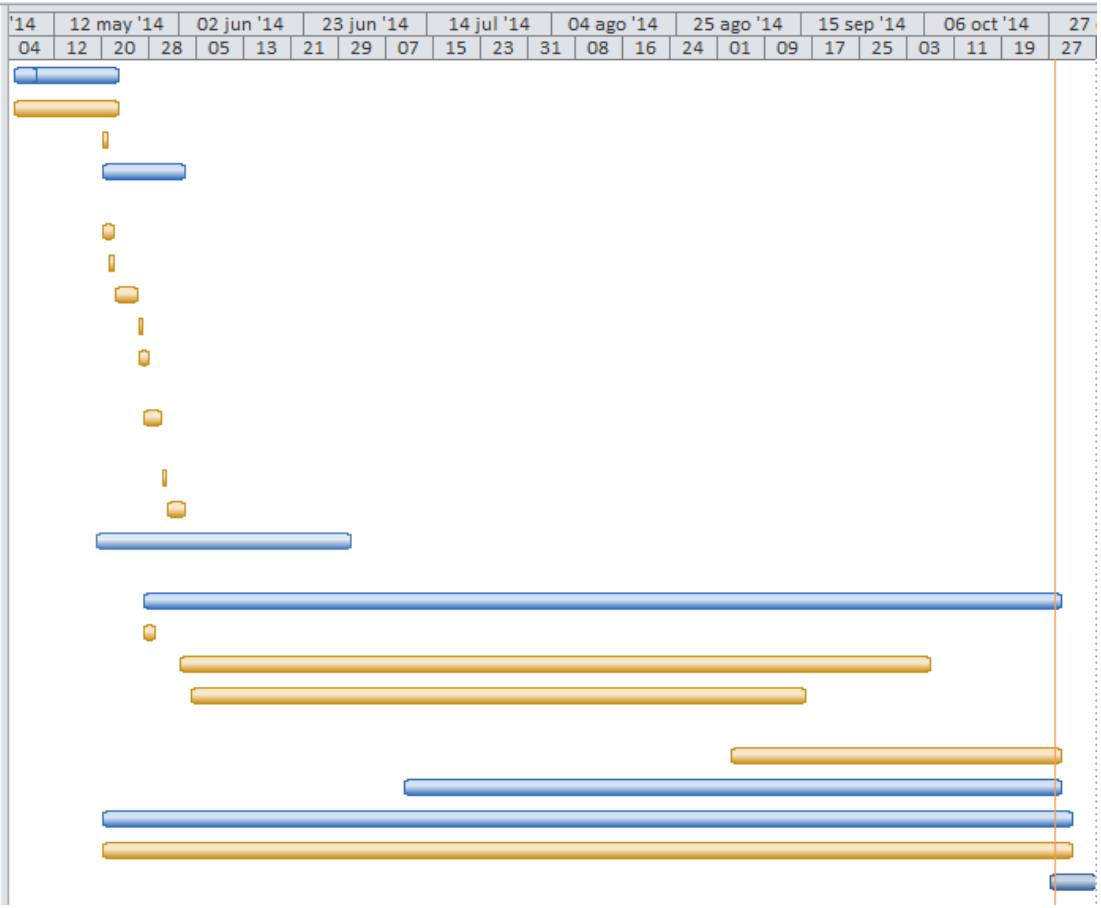


Figura 88. Diagrama de Gantt (real)

4. Conclusiones

En este apartado se van a comentar las aportaciones, las modificaciones, las dificultades y los aspectos positivos que han tenido lugar a lo largo del desarrollo de la aplicación para la generación de Guías de Test.

En el comienzo del desarrollo de este proyecto, se desconocía en gran medida en qué consistía toda aquella terminología como metamodelos, gramáticas, formatos estándares de Test Aiken, Gift, IMS QTI, Guías de Práctica Clínica, etc. Fue una tarea laboriosa empaparse de toda esa nueva información, la cual sería imprescindible para un correcto desarrollo del proyecto. Esa dificultad, la dificultad de comprender en que consistía el objetivo principal, fue un gran hándicap a tener en cuenta. Hubo que leer artículos sobre estándares de test y formalismos de Guías de Práctica Clínica entre otros, para focalizar bien las necesidades y objetivos del proyecto. Todas aquellas horas, dedicadas al estudio y previo acomodamiento, no se recogen en la bolsa de horas de aprendizaje. Sin embargo son consideradas de vital importancia, ya que gracias a ese estudio previo se consiguió afrontar y entender lo que realmente había que hacer.

A la hora de realizar la aplicación se concocía muy poco sobre el funcionamiento de la plataforma y leguajes y tecnologías utilizadas. Por tanto, este hecho fue una de las mayores complicaciones. Fue preciso leer manuales, visitar foros y apuntes para familiarizarse con el uso del IDE de desarrollo, su Framework y complementos. A su vez, paulatinamente se fueron adquiriendo conocimientos en los lenguajes de programación utilizados en el presente proyecto, gracias a la consulta de libros y páginas Web. Antes de enfrentarse cara a cara con la aplicación en sí, se realizaron pequeños proyectos con el fin de ir aprendiendo las técnicas básicas y formarse en las áreas necesarias. Pequeños proyectos como sencillos DSL, creación de proyectos EMF, diseño de metamodelos simples, etc.

Además, en el ámbito personal se han adquirido firmes conocimientos acerca del desarrollo de software dirigido por modelos. Sin olvidar la pequeña incursión en el campo de las Guías de Práctica Clínica y los Test Informatizados. Y no sólo eso, sino que también se ha conseguido estimar, planificar y documentar un proyecto a nivel profesional. Todo ello, partiendo de la base del desconocimiento de las tecnologías y herramientas utilizadas en el proyecto, supone una grata sensación de satisfacción y superación personal. Ya que al comienzo afloraba un cúmulo de dudas y desconocimiento y finalmente se han superado los obstáculos, hasta llegar al desarrollo final de la aplicación presentada en este documento.

En conclusión, en este proyecto se han desarrollado los módulos para una aplicación de mayor medida, los cuales permiten facilitar la tarea de generación de Guías de Práctica Clínica partiendo de formatos de test estándar.

5. Trabajo futuro

Este punto consistirá en tratar las posibilidades de ampliación y mejora de la aplicación desarrollada. Se van a exponer una serie de funcionalidades, las cuales se podrían incluir en el proyecto, con el fin de conseguir una aplicación más completa.

Una posibilidad de mejora, para el presente proyecto, sería la automatización de alguno de sus pasos mediante la mejora de alguna de sus reglas de transformación y quizá algún cambio en alguno de sus metamodelos. La inclusión automática del tipo de preguntas para los elementos Pregunta y Respuesta en la generación de modelos conformes a MTest desde modelos conformes a MGift. También sería interesante que la regla que transforma modelos conformes a MTest en modelos conformes a MGPC filtrase de un modo más correcto algunos elementos de los cuales genera elementos "basura", para evitar así tener que borrarlos a mano posteriormente.

Sería muy atractivo poder conseguir una integración de los cuatro modulos o proyectos en uno solo, y que mediante el uso de interfaces se le fuera guiando al usuario por todos los paso del proceso. Pudiendo elegir él que tipo de test crear o validar y que el sistema lo validase contra uno de los dos estándares automáticamente, para posteriormente generar su modelo y proceder a las transformaciones hasta llegar a un modelo conforme a MGPC.

Sería aconsejable realizar una tarea de automatización de pruebas. De este modo se conseguiría ahorrar mucho tiempo. Ya que, si se consiguen automatizar las pruebas descritas en el proyecto, para futuras ampliaciones de la aplicación, se podrían ejecutar de un modo automático, comprobando así las alteraciones ocasionadas en la funcionalidad de la aplicación, de un modo más eficaz.

En lo que a pruebas de seguridad se refiere, a pesar de que el sistema podría tratar con datos sensibles de carácter personal y privados, es el usuario quién los introduce, genera el modelo y las transformaciones y posteriormente es responsable de los elementos generados de un modo local. El sistema no almacena datos ni los retiene, por lo que no se ha considerado el realizar pruebas de seguridad.

Para concluir, cabe añadir la finalización del desarrollo y prueba del caso de uso Generar GPC en Eguider. El cual cerraría el proceso completo del propósito del presente proyecto de principio a fin.

6. Actas de reunión

En éste apartado se especifican las reuniones efectuadas durante el desarrollo del proyecto mediante sus respectivas actas, una por reunión. Se describen a continuación.

Tabla 16. Acta de reunión 1

ACTA DE REUNIÓN 1	
Proyecto	Aplicación para la generación de Guías de Test.
Asunto	Presentación del proyecto de viabilidad.
Lugar	Despacho de Tomás Antonio Pérez.
Fecha	5 de mayo de 2014.
Hora comienzo y finalización	10:00 – 11:00
PARTICIPANTES	
Tomás Antonio Pérez Fernández.	
Eduardo García García.	
ORDEN DEL DÍA	
- Presentación inicial del proyecto. - Primeras impresiones y dudas surgidas.	
RESUMEN / DECISIONES TOMADAS	
- Presentación del proyecto, en que consiste, así como el camino y las pautas a seguir gracias a las orientaciones dadas por el director del mismo.	

Tabla 17. Acta de reunión 2

ACTA DE REUNIÓN 2	
Proyecto	Aplicación para la generación de Guías de Test
Asunto	Memoria e implementación.
Lugar	Despacho de Tomás Antonio Pérez.
Fecha	20 de mayo de 2014.
Hora comienzo y finalización	10:25 – 12:15
PARTICIPANTES	
Tomás Antonio Pérez Fernández.	
Eduardo García García.	
ORDEN DEL DÍA	
- Revisión del proyecto de viabilidad. - Toma de contacto con estándares y organización de módulos.	
RESUMEN / DECISIONES TOMADAS	
- Se ha presentado el material desarrollado hasta el momento. El proyecto de viabilidad ha sido pulido, limado y mejorado, aun así aún quedan detalles que se revisarán al finalizar el proyecto. Se ha marcado el camino para ir avanzando en el diseño de las gramáticas (implementación) y metamodelos.	

Tabla 18. Acta de reunión 3

ACTA DE REUNIÓN 3	
Proyecto	Aplicación para la generación de Guías de Test
Asunto	Análisis, diseño e implementación.
Lugar	Despacho de Tomás Antonio Pérez.
Fecha	23 de julio de 2014.
Hora comienzo y finalización	11:15 – 13:00
PARTICIPANTES	
Tomás Antonio Pérez Fernández.	
Eduardo García García.	
ORDEN DEL DÍA	
- Revisión de gramáticas - Revisión del análisis y el diseño. - Posible diseño de los metamodelos	
RESUMEN / DECISIONES TOMADAS	
- Se han puesto sobre la mesa los problemas presentados hasta el momento y los diseños desarrollados. Tanto los casos de uso como los metamodelos y reglas de transformación han sido comentados y analizados. Para los primeros se ha establecido de forma robusta su diseño final. Los segundos en cambio aún sufrirán alteraciones, no obstante se han establecido unas pautas a tener en cuenta a la hora de diseñarlos. Cabe comentar las primeras impresiones acerca de las gramáticas, metamodelos y tablas de transformación. El conjunto durante la reunión, ha sufrido modificaciones.	

Tabla 19. Acta de reunión 4

ACTA DE REUNIÓN 4	
Proyecto	Aplicación para la generación de Guías de Test
Asunto	Finalización del proyecto.
Lugar	Despacho de Tomás Antonio Pérez.
Fecha	3 de noviembre de 2014.
Hora comienzo y finalización	12:20 – 13:30
PARTICIPANTES	
Tomás Antonio Pérez Fernández.	
Eduardo García García.	
ORDEN DEL DÍA	
- Revisión de la memoria del proyecto. - Realización de prueba de aceptación.	
RESUMEN / DECISIONES TOMADAS	
- Presentación de la ejecución de la aplicación, para realizar una prueba con la misma, la cual certifique su correcto funcionamiento y cumpla con los objetivos del proyecto. Dicha prueba ha sido satisfactoria. Una vez finalizada la parte que corresponde a la implementación del proyecto, se procedió a revisar la documentación presentada en la memoria del proyecto, realizando las correcciones pertinentes y dejando la memoria preparada para ser presentada.	

7. Bibliografía

Se ha decidido dividir este apartado en tres bloques, las referencias bibliográficas, los libros utilizados en el proyecto y las páginas Web consultadas.

7.1. Referencias

Las referencias plasmadas en el presente documento se describen a continuación.

- [1] Van Der Linden, W. J., & Glas, C. A. W. (2000). Computerized adaptive testing: Theory and practice. Dordrecht (The Netherlands): Kluwer Academic. Press

- [2] Wainer, H. (2000). Computerized adaptive testing: A primer (2nd ed.). Mahwah, New Jersey (USA): Lawrence Erlbaum Associates.

7.2. Libros

- Javier López Cuadrado, Anaje Armendariz, Tomás A. Pérez, Rosa Arruabarrena y José Á. Vadillo. (2009). Computerized adaptative testing, the item bank calibration and a tool for easing the process. Pag: 457- 478. Technology, Education and Development. Alexandar Lazinika y Carlos Calafate (eds). In-Teh editorial. ISBN: 978-953-307-007-0.

- Conchi Presedo, Anaje Armendariz, Javier López Cuadrado. Calibración de Ítems para Test Informatizados. Editorial Académica Española. ISBN: 978-3-8465-7649-6.

- Anaje Armendariz Leunda. Calibración psicométrica guiada por un sistema experto. Editorial Académica Española. ISBN: 978-3-659-02221-0.

7.3. Sitios Web

Para desarrollar la implementación de la aplicación, además de los libros anteriormente enumerados, se ha necesitado de la información alojada en diversas direcciones Web. Estas url también se han consultado para ampliar la información descrita en diversos apartados de la memoria del proyecto, como pudieran ser Herramientas y Tecnologías utilizadas, o Estado del arte, entre otros. A continuación se citan los hiperenlaces a las páginas consultadas:

- http://en.wikipedia.org
- http://en.wikipedia.org/wiki/Model-driven_engineering

- https://docs.moodle.org/24/en/Aiken_format
- https://docs.moodle.org/23/en/GIFT_format

- http://ares.cnice.mec.es/informes/16/contenido/31.htm
- http://www.imsglobal.org/question/
- http://www.imsglobal.org/question/qtiv2p1/imsqti_implv2p1.html
- http://www.imsglobal.org/question/qtiv2p1/imsqti_infov2p1.html
- http://www.imsglobal.org/question/qtiv2p1/examples/items/choice.xml
- http://www.imsglobal.org/question/qtiv2p1/examples/items/inline_choice.xml
- http://www.imsglobal.org/question/qtiv2p1/examples/items/associate.xml
- http://www.imsglobal.org/question/qtiv2p1/examples/items/text_entry.xml

- https://www.eclipse.org/
- http://www.eclipse.org/modeling/
- http://www.eclipse.org/modeling/emf/
- http://www.eclipse.org/Xtext/
- http://www.eclipse.org/xtend/
- http://www.eclipse.org/atl/
- http://wiki.eclipse.org/OCL
- http://wiki.eclipse.org/Xbase
- http://wiki.eclipse.org/Xpand

- http://www.omg.org/spec/XMI/
- http://www.omg.org/spec/OCL/
- http://www.jetbrains.com/mps/

- http://www.fisterra.com/formacion/cursos/casoclinico.asp?idCaso=262
- http://u008769.si.ehu.es/Guidmex/login.seam
- http://www.e-guidesmed.ehu.es
- http://www.guider.ehu.es
- https://docs.moodle.org/all/es/Formato_Aiken
- https://docs.moodle.org/23/en/GIFT_format
- https://docs.moodle.org/23/en/GIFT_format#Question_format_examples

- http://e5.onthehub.com/WebStore/ProductsByMajorVersionList.aspx?ws=0060a29a-689b-e011-969d-0030487d8897&vsro=8

Todas las direcciones Web han sido consultadas a fecha: noviembre de 2014.