ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA

INGENIERÍA TÉCNICA EN INFORMÁTICA DE GESTIÓN

Curso Académico 2009/2010

Proyecto de Fin de Carrera:

APLICACIÓN PARA LA EVALUACIÓN DEL DISEÑO DE ENTORNOS ADAPTATIVOS DENTRO DE COMOLE

Autor: Humberto Herrero Rodríguez

Tutora: Estefanía Martín Barroso

Resumen

Las Tecnologías de la Información, junto con Internet, han irrumpido en todos los aspectos de nuestra vida de una manera incontrolable. Es por ello que las nuevas sociedades originadas en este ambiente, necesitan nuevos patrones y requerimientos en cuanto a la educación de sus miembros se refiere. En este ámbito surgen las herramientas e-learning, gracias a las cuales los estudiantes realizan su aprendizaje utilizando medios electrónicos.

CoMoLE (*Context-based adaptive Mobile Learning Environments)* es una de estas herramientas. En concreto, es un sistema adaptativo que realiza la recomendación de actividades en base a las características de los usuarios, a las acciones que estos realizan mientras se encuentran interactuando con la plataforma y a los distintos contextos en los que los usuarios se puedan encontrar.

Sin embargo el diseño y creación de estos entornos adaptativos no es una tarea fácil. Si a la complejidad inherente del problema, sumamos que los entornos son a menudo creados con herramientas en las cuales no se realizan apenas control de errores, y que además, en muchas ocasiones la creación de los entornos es realizada "a mano" (sin ninguna herramienta que automatice el proceso), puede que el resultado no sea el esperado. Por este motivo resultaría muy útil facilitar a los diseñadores, algún tipo de aplicación que se encargase de evaluar si un curso adaptativo está correctamente diseñado. Esta validación se realizaría en función de los diferentes aspectos configurados en el curso como son: las capacidades adaptativas que se tienen en cuenta, la configuración de rasgos de adaptación, la definición de actividades, las características de todo el curso y el empleo de las herramientas colaborativas.

Con este objetivo, este Proyecto Final de Carrera se centra en la evaluación de entornos adaptativos creados para ser utilizados con el sistema CoMoLE. Se pretende ayudar al diseñador a decidir si un curso ha sido creado de manera correcta. Además no sólo se evaluarán aspectos relacionados con el diseño del curso, sino que también se buscarán posibles errores a partir de los datos sobre la interacción de los estudiantes que ya hayan realizado el curso.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS Y TABLAS

Capítulo 1 :Introducción

En este capítulo se realizará una introducción del Proyecto Fin de Carrera. En primer lugar, se presentará las principales características de CoMoLE, ya que será el sistema base sobre el que se va a realizar el proyecto. Posteriormente se planteará la motivación que ha dado lugar a la elaboración del proyecto, presentándose seguidamente una serie de escenarios de aplicación. A continuación, se enumerarán de forma concisa los objetivos del proyecto. Finalmente, se describe la organización del resto de capítulos de este documento.

1.1. Presentación de CoMoLE

CoMoLE es un sistema que permite la creación de entornos adaptativos móviles. Está centrado en la recomendación de actividades y en la generación dinámica de espacios de trabajo basados en la información disponible de los usuarios y su contexto. Las actividades que el sistema recomendará se realizarán en función de las características personales, preferencias, y acciones de los usuarios, así como del contexto en el que estos se encuentren.

Para conseguir realizar la adaptación, CoMoLE integra un mecanismo de recomendación. Dicho mecanismo se encargará de seleccionar las actividades que se recomendarán a cada usuario, en función de la situación en la que estos se encuentren. Esta adaptación se producirá en base a tres reglas que definirán cómo adaptar las actividades a los distintos usuarios y grupos teniendo en cuenta los rasgos definidos tanto en el modelo de usuario como en el del grupo, así como las características de las actividades. De esta manera se proporcionará a los usuarios los contenidos y herramientas apropiados a sus circunstancias. La recomendación será realizada en función de los criterios que los diseñadores hayan especificado, pero también en función de la información sobre la interacción de otros usuarios.

La generación de espacios de trabajo que permita a los usuarios la realización de las actividades, será realizada de forma dinámica en función de la información almacenada

y gestionada por el sistema. Esta información involucrará a usuarios, grupos, actividades, recursos, capacidades de adaptación, y acciones previas realizadas por otros usuarios.

Los diferentes entornos adaptativos que se generen tendrán una característica común: el fácil acceso a ellos por parte de los usuarios. El sistema, permite que cualquier usuario pueda acceder a ellos desde cualquier dispositivo que disponga de un navegador Web y que esté conectado a Internet. Este aspecto resulta fundamental ya que la red se ha convertido en la forma más fácil de acceder a la información globalmente.

Es importante destacar que para que CoMoLE pueda realizar todo el trabajo anteriormente mencionado, es necesario la creación, configuración y almacenamiento de la información de los diferentes elementos de estos entornos. Por ello se desarrolló una herramienta que facilitase a los diseñadores estas tareas. Dicha herramienta permite la creación de cursos que contendrán todo lo necesario para el funcionamiento de CoMoLE. La labor del presente proyecto es la evaluación de dichos cursos.

1.2. Motivación

Las herramientas de autor proporcionan una interfaz para la creación de entornos adaptativos, permitiendo la configuración de los entornos y la creación de las actividades de los mismos. Gracias a ellas muchos profesores pueden crear sus propios entornos para poder usarlos con sus alumnos, potenciando de esta manera el aprendizaje de los mismos, y aprovechándose de las múltiples ventajas que nos proporciona la enseñanza a través de medios multimedia.

Sin embargo no resulta fácil la creación de estos entornos, ya que es una tarea muy laboriosa, en la que se encuentran involucrados numerosos aspectos. Pese a que las herramientas de autor nos puedan simplificar el problema, sigue siendo difícil controlar la multitud de aspectos que involucran la realización de un entorno adaptativo de cierta envergadura.

A la hora de crear y configurar un nuevo entorno de aprendizaje dentro del sistema CoMoLE habrá que:

- Crear un nuevo entorno y definir sus características generales.
- Definir los criterios generales de recomendación basados en el contexto.
- Describir los conjuntos de herramientas que se emplearán en la realización de actividades de colaboración.
- Crear las propias actividades incluyendo todos los aspectos necesarios para la correcta definición de las mismas.

Estas tareas descritas hacen una idea de la dificultad a la que se enfrentan los diseñadores de los cursos. Imaginemos algunas situaciones:

- Un profesor sin altos conocimientos informáticos quiere realizar un diseño de un curso para sus alumnos. A la hora de realizarlo intenta comprender la herramienta de autor, y tras un gran esfuerzo consigue realizar el curso, que a su parecer le ha quedado bastante bien. Sin embargo ha cometido numerosos errores de los cuales no se dará cuenta hasta que ponga en funcionamiento el curso que ha diseñado con la herramienta de autor (como por ejemplo definición de actividades innecesarias, omisión de contenidos para todos los dispositivos, etc.). En esta situación el sistema no funcionará como el diseñador espera. Sin embargo, el sistema no avisará del fallo concreto que se está produciendo, simplemente tendrá un comportamiento inesperado.

- Un profesor familiarizado con el diseño de cursos y que domina el uso de la herramienta de autor, se dispone a realizar un curso. Tras haberlo realizado no está seguro de si ha utilizado correctamente un determinado aspecto de la configuración de manera correcta, ya que son demasiados aspectos a configurar y por tanto es fácil que haya cometido un error en un pequeño despiste. El diseñador no tendría manera de comprobar esto a no ser que entendiese los ficheros de datos y la configuración del sistema, tarea que no es trivial.

Estas situaciones son sólo dos ejemplos de posibles escenarios en los que los diseñadores se podrían encontrar cuando realizasen el diseño de entornos adaptativos

dentro de CoMoLE, y que permiten hacerse una idea de los numerosos problemas que puede acarrear el diseño de entornos adaptativos.

1.3. Planteamiento del problema

Como ya se ha ilustrado, la creación de entornos adaptativos mediante herramientas de autor es una tarea bastante complicada. Esto se debe principalmente a que son múltiples los factores y variables a tener en cuenta, y resulta prácticamente imposible controlarlos todos.

Además, muchas de las herramientas de autor que se usan para la creación de entornos adaptativos no controlan algunos aspectos básicos de la configuración de los cursos. Esto deriva en que el control de errores realizado en la fase de diseño sea muy básico, ya que dichas herramientas no minimizan el número de errores que se darán en cuando el curso adaptativo sea puesto en marcha en el sistema CoMoLE. Sería conveniente que el profesor o diseñador pudiera tener una herramienta que le validase si lo que ha creado es correcto.

También cabe la posibilidad de que la creación de los cursos se hiciese directamente sin la herramienta de autor, por ejemplo porque no se tuviese a mano. Trabajar sobre los ficheros de datos y configuración es aun más difícil, aunque no imposible, ya que estos están almacenados en unos ficheros especiales utilizando XML (*Extensible Markup Language*). Trabajar directamente sobre los ficheros incrementa mucho la posibilidad de cometer errores, ya que se puede cortar y pegar fácilmente, pero igual de sencillo es también hacer mal el proceso. Por ello también sería útil poder validar un curso creado o modificado de esta manera. De esta forma se sacarían los fallos de diseño en esta fase, y se podrían corregir sin problemas.

Por lo tanto podemos concluir que la parte de diseño de entornos adaptativos es un gran problema, debido principalmente a los siguientes motivos:

1. Existen muchas posibilidades de configuración de estos entornos que pueden desorientar a los diseñadores o hacerles sentir desbordados.

2. La herramienta de autor actual ofrece algún tipo de control, pero en general no es completa a la hora de verificar el correcto diseño de un entorno.

3. Por último, en algunas ocasiones no se utiliza la herramienta de autor para la creación de los cursos, y se trabaja directamente sobre los ficheros de datos y la configuración global de estos cursos.

Estas tres razones hacen que el presente Proyecto Fin de Carrera pueda resultar de gran ayuda en la evaluación de entornos adaptativos creados para utilizarlos con el sistema CoMoLE.

1.4. Objetivos generales

El presente proyecto se enmarca, pues, en dar soporte a la evaluación de entornos adaptativos para CoMoLE. Para ello, la herramienta desarrollada detectará las posibles inconsistencias o problemas en las recomendaciones fijadas por el diseñador del entorno. El análisis realizado se centrará principalmente en cuatro aspectos fundamentales:

- Evaluación de los filtros usados en el curso: Se comprobará el uso de determinadas reglas impuestas por el diseñador en la creación de un curso.

- Evaluación de la configuración del curso: Evaluaremos numerosos aspectos de la configuración básica sobre los aspectos generales de un curso.

- Evaluación de las actividades que componen el curso: Los cursos están compuestos por un conjunto de actividades. Se evaluará la correcta construcción de cada una de las actividades que forman el curso: aspectos generales de definición de reglas estructurales o individuales, contenidos, herramientas asociadas, etc.

- Evaluación de los datos dinámicos: Se detectarán los posibles fallos que puedan existir en el curso, gracias al análisis de las interacciones de las acciones previas de otros usuarios.

Como principal objetivo destacamos que se pretende es avisar de todas las inconsistencias encontradas en los distintos aspectos que se evaluarán del curso. Para ello se informará al diseñador con datos concretos acerca de los problemas encontrados en el diseño del mismo. Será importante no saturar al diseñador con demasiada información, pero a la vez se le deberá informar de forma precisa para que sepa exactamente donde se está produciendo el problema y por qué se está produciendo.

Por otro lado señalar, que también se pretende dotar a la herramienta desarrollada en el presente proyecto, de una gran simplicidad, de tal forma que su uso sea fácil e intuitivo.

1.5. Estructura del documento

El presente documento se compone de los siguientes capítulos:

- Capítulo 1: Este capítulo corresponde al actual y pretende relatar el propósito del proyecto. En él se describe el contexto, la problemática y los objetivos generales del mismo.
- Capítulo 2: En este capítulo se detallarán los objetivos específicos del proyecto incluyendo la descripción de cada uno de los problemas tratados en él, así como un estudio de las alternativas y la metodología empleada.
- Capítulo 3: Este capítulo contendrá todos los aspectos relacionados con el desarrollo de la aplicación que se propone en el proyecto. En él se incluirá la especificación, el diseño y la implementación de la aplicación desarrollada.
- Capítulo 4: Este capítulo expondrá las conclusiones obtenidas en la realización de proyecto y posibles ideas para el futuro.

Capítulo 2 : Motivación

En el presente capítulo vamos a abordar y estudiar a fondo cuales han sido los principales objetivos que han sido tratados en el desarrollo de la aplicación. Para ello analizaremos cada uno de los problemas con los que nos podemos encontrar a la hora de analizar un entorno adaptativo dentro del sistema CoMoLE, y como sería posible su solución.

Partamos de la idea de que la aplicación desarrollada en el presente proyecto analiza los ficheros de datos creados por la herramienta de autor, y que son los ficheros con los que trabajaría el sistema CoMoLE. En estos ficheros es donde se encuentran almacenados todos los datos relacionados con un curso, desde la configuración del mismo, hasta la información de cada una de las actividades que lo componen. Estos ficheros nos servirán como base para poder analizar todos los aspectos del curso, de tal forma, que mediante un análisis exhaustivo de ellos, podremos sacar las conclusiones necesarias para informar de las posibles inconsistencias que puedan existir en el curso, y así, avisar de ellas al diseñador para su posterior corrección.

Por este motivo el principal objetivo sobre el cual se enfoca el proyecto, es el análisis de las inconsistencias que puedan existir en un curso. Utilizaremos el término inconsistencia para referirnos a la falta de coherencia en cualquiera de los aspectos que intervienen en el diseño de un curso, y que provocan el acontecimiento de eventos variables o no controlados en la ejecución del entorno adaptativo. La aplicación dará información precisa de la inconsistencia que se está produciendo, avisando claramente al diseñador, del motivo que puede estar provocando dicha inconsistencia. Además hay que tener en cuenta que las inconsistencias pueden ser de dos tipos:

- Errores: Los errores serán los fallos cometidos en el diseño del entorno adaptativo y que provocarán una ejecución incorrecta del sistema de recomendación de actividades. Este hecho hará que exista diferencias entre el resultado real obtenido y la previsión que se había hecho. Un ejemplo de un posible error sería definir una actividad como compuesta, y no asociarla subactividades en las cuales se descompone.

- Advertencias: Las advertencias sobre el diseño realizado del entorno adaptativo, no provocan una mala ejecución del sistema de recomendación, incluso, el diseñador puede haber decidido realizar así el diseño. Sin embargo es conveniente avisar de ellas por si estas se han producido por un despiste del diseñador. Por ejemplo una posible advertencia sería que el diseñador no hubiera provisto al sistema de todas las versiones de contenidos para los dispositivos utilizados.

Por lo tanto, la aplicación desarrollada va a resultar de vital importancia a la hora de verificar los entornos adaptativos creados por los diseñadores. Nos va a permitir determinar si un entorno está bien construido, y en el caso de que esto no sea así, nos va a facilitar un informe de los problemas (errores o advertencias) que ha encontrado, para que puedan ser corregidos y así impedir que se produzcan situaciones desagradables durante la ejecución de un curso adaptativo.

2.1. Análisis de inconsistencias

A continuación se va a proceder a explicar cada uno de los tipos de inconsistencias que podemos encontrar al realizar la evaluación de un entorno adaptativo para CoMoLE. Se detallará el por qué surgen estos problemas y cuáles han sido las alternativas tomadas para la resolución de cada uno. De esta manera lo que se pretende es dar una idea clara de cuáles son los aspectos fundamentales sobre los que se realiza la evaluación de los entornos así como el procedimiento llevado a la hora de tratar cada uno de ellos.

2.1.1. Inconsistencias en los filtros de adaptación

Los filtros son módulos que se encargan de procesar reglas de adaptación que definen cómo adaptar las actividades a los distintos usuarios y grupos teniendo en cuenta los rasgos definidos tanto en el modelo de usuario como en el del grupo, así como las características de las actividades. Esta adaptación se lleva a cabo fundamentalmente basándose en 3 tipos de reglas:

- Reglas estructurales: Este tipo de reglas nos permiten definir qué actividades van a formar parte de una actividad compuesta, así como las relaciones que existen entre estas subactividades. De este modo, una misma actividad puede descomponerse en diferentes subactividades y la guía de navegación ofrecida entre las subactividades puede ser la misma para todos los estudiantes o distinta dependiendo de a qué tipo de estudiantes vayan orientadas las actividades, e incluso del contexto en el que se encuentren.
- Reglas de contexto: Gracias a ellas podremos asociar requisitos del contexto del usuario a la realización de ciertas actividades, por lo que dependiendo de si un usuario se encuentra en un determinado contexto u otro se le recomendarán distinto tipo de actividades.
- Reglas individuales: Con este tipo de reglas se pretende establecer requisitos específicos para el desarrollo de una actividad. Estos requisitos pueden depender de que se hayan realizado determinadas actividades previamente, de una determinada fecha de inicio, de un lugar de realización, etc.

La inconsistencia que se puede producir asociada al uso de determinados filtros, es que se establezca en la configuración del entorno el uso de un determinado tipo de regla, pero que posteriormente, no se incluyan reglas de ese tipo.

Imaginemos por ejemplo que un diseñador realiza un curso, y establece que se van a usar reglas estructurales con el fin de organizar de manera jerárquica las actividades. Si el diseñador, cuando define las actividades no pone que alguna actividad es compuesta (y por tanto tiene reglas estructurales), es cuando se produce una inconsistencia. Esto es debido a que hemos definido que habrá reglas estructurales pero luego a la hora de estructurar las actividades, esto no ha sido así.

Por ello comprobaremos que si se establece el uso de reglas estructurales o de reglas individuales, al menos una actividad del curso diseñado, debe tener definidas reglas de este tipo. Por otro lado, si se usan reglas de contexto, el fichero de datos que contiene dichas reglas al menos debe tener una regla especificada.

Este tipo de inconsistencia corresponderá a una advertencia, ya que el diseñador puede haber querido definir estas reglas, pero por algún motivo decide no emplearlas. Sin embargo no está demás avisarle de este hecho, para que sepa que las reglas que está diciendo que usa en el curso, verdaderamente luego no las está definiendo.

2.1.2. Inconsistencias en la configuración del curso

Como ya se ha explicado, el diseño de un entorno adaptativo es complicado. A la hora de establecer la configuración del curso, existen multitud de opciones que son fáciles de confundir. Un diseñador experto puede cometer fácilmente un despiste, y no digamos si el diseñador no es especialista en el tema, en cuyo caso se puede complicar mucho la vida y poner configuraciones que no sean las apropiadas.

Para cada entorno se definen una serie de características generales como: idiomas a los que se dará soporte en el entorno (contenidos e interfaz), descripciones del propio entorno en los distintos idiomas en los que estará disponible, rasgos de los usuarios que se van a tener en cuenta para realizar la recomendación de actividades y la adaptación de espacios de trabajo, actividades a realizar por los estudiantes, roles que los estudiantes pueden tener en las actividades colaborativas, características generales de los grupos de trabajo como por ejemplo si los estudiantes serán agrupados de forma automática por el sistema, el tamaño de los grupos de trabajo o el conjunto de herramientas que se facilitarán a los estudiantes para poder interactuar entre sí y realizar las actividades colaborativas.

A continuación se detallará las inconsistencias asociadas a las características anteriormente citadas y que serán analizadas por nuestra aplicación.

2.1.2.1. Rasgos de adaptación

Un rasgo de adaptación es cualquier característica cuyos posibles valores puedan representarse a través de un conjunto de valores discretos (numéricos o estereotipos) o mediante intervalos de valores en el caso de datos numéricos. Estos rasgos nos van a permitir establecer ciertos criterios a la hora de hacer la recomendación de actividades.

Algunos ejemplos de rasgos de adaptación utilizados en el área de la enseñanza son: si el estudiante cursa por primera vez una materia, su nivel de conocimientos previo, las distintas dimensiones del estilo de aprendizaje del usuario, el lugar donde se encuentra, el dispositivo utilizado o el tiempo que tiene disponible para realizar las actividades.

Por lo tanto para que el mecanismo de recomendación adaptase las actividades y el espacio de trabajo, por lo menos debería haber un rasgo de adaptación definido en el curso, para que el sistema sea capaz de hacer la recomendación de actividades más apropiada dependiendo del perfil del usuario.

Si al analizar los ficheros de datos, nuestra aplicación comprueba que no se ha definido ningún rasgo de adaptación para el curso, esta lanzará un error. De esta forma se avisaría al diseñador de que no está especificando los rasgos de adaptación, los cuales resultarían básicos para el funcionamiento de este sistema de recomendación de actividades.

Los rasgos estereotipados son rasgos de adaptación cuyos posibles valores se representan a través de estereotipos. Por ejemplo, se podría definir el rasgo de adaptación "Nivel de conocimiento" y definir tres posibles valores: básico, medio o avanzado (ver fila 2 en la Tabla 1).

	Rasgo	**Posibles valores**
1	Punto de inicio	nuevo, repetidor
2	Nivel de conocimiento	básico, medio, avanzado
3	Lugar	clase, casa, otros

Tabla 1: Posibles rasgos de adaptación cuyos valores son estereotipos

Para que un rasgo estereotipado este definido correctamente, éste debe tener al menos dos posibles valores asociados. Si esto no fuese así, nos encontraríamos ante una nueva inconsistencia. En este caso, dicha inconsistencia sería un error, ya que no tendría sentido definir un rasgo con un único valor.

2.1.2.2. Tamaño de los grupos

Los entornos adaptativos pueden incorporar la opción de realizar actividades en grupos de trabajo. La inclusión de actividades colaborativas enriquece el aprendizaje y favorece la interacción de los usuarios, la discusión sobre determinados conceptos, la resolución cooperativa de problemas y la construcción de conocimiento. Además, realizar actividades colaborativas dentro de este tipo de entornos contribuye a reducir la sensación de aislamiento que puedan tener los usuarios cuando se encuentran interactuando solos con este tipo de entornos.

El tamaño de los grupos que se formarán para la realización de actividades colaborativas es una decisión del diseñador. Por este motivo no es objetivo de esta aplicación controlar que este tamaño este dentro de unos límites determinados. Sin embargo, sí debemos tener en cuenta que no se hayan definido valores que carezcan de sentido. Veamos algunos ejemplos:

- Que se ponga que el tamaño máximo de los grupos es 1 no tiene sentido, ya que no podemos imaginar un grupo formado por una única persona porque se perdería la base de realizar actividades colaborativas entre varios usuarios.
- Tampoco podemos imaginar grupos en los cuales el tamaño mínimo del grupo sea mayor que el tamaño máximo del mismo.

Por este motivo la aplicación avisará de un error en la configuración de los tamaños de los grupos de trabajo, si el tamaño mínimo y máximo de los grupos carecen del sentido obvio para realizar las tareas enfocadas a realizarse de manera conjunta por varias personas.

2.1.2.3. Idiomas y descripciones

Los entornos adaptativos pueden ser diseñados de tal manera que usuarios con distintos idiomas puedan realizar un mismo curso, siempre que el diseñador provea de las descripciones necesarias en los distintos idiomas y de las distintas versiones de contenido. Por este motivo cuando un diseñador esté realizando un curso, podrá introducir los diferentes idiomas para los que se va a diseñar. Por cada idioma es

necesario que el diseñador introduzca una breve descripción sobre el curso. Sin embargo este aspecto no siempre se lleva a cabo.

El no tener un control adecuado de los idiomas y las descripciones de un curso, implica que existan idiomas que carezcan de su descripción correspondiente. Este hecho no sería correcto, ya que como se ha mencionado anteriormente, todo idioma debería llevar asociado una descripción. Por tanto es importante avisar de un error si este hecho se produce. Gracias a la detección de este tipo de inconsistencia, ahorraríamos a los usuarios la desagradable situación de intentar realizar un curso del cual no disponen de información en su idioma.

2.1.2.4. Tipos de actividades y reglas de contexto

Las reglas generales de contexto nos permiten separar del conjunto de todas las actividades, aquellas que se consideran apropiadas recomendar a los usuarios debido al contexto actual en el que se encuentran y aquellas que no lo son. También se puede considerar sus características personales de modo que sea posible ofrecer distintas recomendaciones a usuarios distintos, incluso cuando se encuentren en el mismo contexto.

Las actividades que forman un curso adaptativo pueden clasificarse en diversos tipos: leer una explicación teórica, observar un ejemplo, realizar un ejercicio de tipo test, etc. Toda regla de contexto deberá llevar asociada uno o varios tipos de actividades recomendadas (o no recomendadas), de acuerdo a la situación especificada en las condiciones de la regla. Por ejemplo si el estudiante tiene menos de 10 minutos sería conveniente proponerle únicamente las actividades de repaso y marcar como no adecuadas el resto de los diferentes tipos de actividades debido a la situación actual.

Por lo tanto, nos encontraremos ante una inconsistencia si en una regla de contexto, no se ha incluido los tipos de actividades recomendados o si los tipos de actividades que lleva asociados una determinada regla no concuerdan con los definidos para el sistema CoMoLE. Esta inconsistencia será un error, y así se lo indicaremos al diseñador.

2.1.2.5. Actividades definidas en el curso

Cuando el diseñador está realizando un curso, elabora un listado con los identificadores de las actividades que forman parte del entorno. Este listado no implica ninguna relación estructural entre actividades, sin embargo las actividades deben ser definidas posteriormente, es decir, que se almacenará la información de cada una de las actividades del curso de forma independiente.

Ante la situación de tener por un lado el mencionado listado, y por otro lado la información de las actividades, nos podemos encontrar ante algunas inconsistencias. Por ejemplo podría darse el caso que el listado de actividades que tenemos, contiene actividades de las cuales no se dispone información. La situación inversa sería que hubiéramos almacenado información de actividades, pero no las hubiéramos incluido en nuestro listado de actividades del curso.

Ante situaciones de este tipo, nos encontramos claramente con una inconsistencia, ya que no existe una correspondencia entre el listado elaborado y la información almacenada. Para evitar que esto quede obviado, y que el diseñador no se dé cuenta de esta circunstancia, se ha decido dar soporte al aviso de este error. Para ello, comprobaremos el listado de actividades que disponemos en la configuración general del curso, contrastándolo con las actividades de las cuales se dispone información específica almacenada. En caso de encontrar alguna divergencia, se avisará al diseñador para que sea consciente del posible problema que se puede ocasionar.

2.1.3. Inconsistencias en las actividades

Las actividades son uno de los principales pilares de los entornos adaptativos que estamos evaluando, ya que sobre ellas el sistema realizará las recomendaciones oportunas. Vamos a pasar a explicar las inconsistencias relacionadas con las actividades, a las que se ha decido dar soporte.

2.1.3.1. Características obligatorias de las actividades

De todas las actividades que componen un entorno, se almacena información de cada una de ellas de forma independiente. Esta información contiene su identificador, el tipo de la actividad, las descripciones en los distintos idiomas definidos en el entorno y su lugar de realización. Además, se puede guardar una fecha de finalización máxima, si existe un tiempo mínimo estimado de realización de la actividad, el modo en que se estructura en distintas subactividades cuando es una actividad compuesta, sus requisitos individuales de realización (si los tiene), cuáles son los contenidos asociados a la misma y, en el caso de actividades colaborativas, información sobre contenidos y herramientas para la generación de los espacios de trabajo colaborativos correspondientes.

De todas ellas, las características obligatorias de una actividad son: identificador de la actividad, descripciones de la actividad en los distintos idiomas para los que el curso esté definido, tipo de actividad y lugar de realización. Algunos ejemplos de los problemas que pueden surgir ante la ausencia de alguna de estas características son:

- Una actividad no tiene identificador, al sistema le va a ser imposible reconocer esta actividad, y por lo tanto va a fallar en tiempo de ejecución.
- Una actividad no tiene asignado el tipo. La actividad resulta inútil sin tipo ya que este aspecto es básico a la hora de realizar las recomendaciones, sobre todo en las reglas de contexto.
- Una actividad carece de lugar de realización, a la hora de recomendar la actividad el sistema no va a saber donde se debería realizar dicha actividad.
- Una actividad no tiene descripciones en alguno de los idiomas. El sistema no va a poder facilitar la descripción de dicha actividad ya que dispone de ella.

Estos son solo algunas de las múltiples situaciones embarazosas que se pueden producir ante la falta de ciertas características de las actividades. Por lo tanto resulta fundamental evaluar si todas las actividades tienen sus características obligatorias, para que si esto no fuese así, avisar al diseñador de este hecho para que se corrijan inmediatamente.

Por ello se lanzará un error avisando de las actividades que no tienen sus características obligatorias, informando también de cuáles son las que faltan. Para ello se analizará una por una todas las actividades que componen el curso, comprobado: identificador de la actividad, descripciones de la actividad en los distintos idiomas para los que el curso esté definido, tipo de actividad y lugar de realización.

2.1.3.2. Actividades compuestas

Una actividad es compuesta si se descompone en varias subactividades más simples, en cuyo caso se almacena información sobre cómo se estructura dicha actividad en términos de reglas estructurales de adaptación dentro de la información específica de la actividad. Cada regla estructural expresa la manera en la que una actividad compuesta se descompone en otras más simples de acuerdo a una determinada condición. Además se almacenará cuáles son las subactividades en las que se divide la actividad compuesta, dependiendo de alguno de sus rasgos de adaptación, de los alumnos a los que vaya dirigida cada una de las descomposiciones, y de la guía de navegación ofrecida entre dichas subactividades en cada uno de los casos.

Es importante asegurar que si el diseñador decide que una actividad es compuesta, que verdaderamente lo sea. Pongamos unos ejemplos para aclarar este aspecto:

- Si el diseñador designa a una actividad como compuesta dotándola de reglas estructurales y luego no define subactividades en cada una de estas reglas, se está produciendo una inconsistencia. Esto es debido a que se está diciendo que una actividad es compuesta y luego no se está definiendo las subactividades que la componen.
- Si el diseñador designa a una actividad como compuesta dotándola de reglas estructurales y luego define una única subactividad en alguna de estas reglas, se está produciendo una inconsistencia. Esto es debido a que la actividad definida como compuesta podría haber sido definida como simple, ya que sólo tiene una subactividad en alguna de las reglas estructurales.

Por lo tanto para considerar que una actividad compuesta está bien definida se debe cumplir lo siguiente: que cada una de las reglas estructurales que forman parte de la

actividad tengan definidas por lo menos dos subactividades en la que se descompone la misma. La aplicación se encargará de evaluar que esta restricción se cumpla. Por cada actividad: se comprobará si la actividad es compuesta, y si lo es, se comprobarán cada una de sus reglas estructurales examinando el listado de sus subactividades.

Tras la evaluación de este tipo de inconsistencias, nos podemos encontrar con: un error si no se define ninguna subactividad dentro de una regla estructural, o una advertencia si sólo se define una subactividad en la regla estructural. Ante la detección de cualquiera de ellas, se debe mostrar al diseñador el aviso correspondiente de la manera más detallada posible.

En relación con la guía de navegación entre actividades, se pueden definir dos posibles guías: directa, donde el estudiante será guiado en cada paso, o flexible, dejando mayor libertad al estudiante para decidir que subactividad quiere realizar. Si no se incluye un valor para la guía de navegación en una regla estructural, nos encontramos ante un nuevo error, ya que el sistema no sabrá qué guía de navegación ofrecer a los estudiantes entre las subactividades que forman parte de una regla. Por ello se seguirá un procedimiento similar al anterior, comprobando la guía de navegación en cada una de las reglas estructurales de las actividades compuestas.

Por último, es necesario comprobar que dentro de la misma regla estructural no se encuentre definida una misma actividad dos veces, ya que si esa regla estructural tuviese varias subactividades con el mismo identificador, no estaría bien definida y sería un error de diseño.

2.1.3.3. Contenidos

Las actividades tienen asociadas una o varias versiones de contenidos multimedia. Si se quieren adaptar los contenidos presentados en una actividad al dispositivo utilizado por los alumnos, es necesario almacenar información sobre qué versión de contenido será la que se utilice a la hora de generar la página de contenidos presentada al usuario en cada caso. También se puede tener en cuenta las características personales de los

alumnos, desarrollando distintas versiones de contenidos para cada uno de los posibles valores de los rasgos que se quieren tener en cuenta en la adaptación de contenidos.

Por lo tanto resultaría útil evaluar que el diseñador haya incluido los contenidos en todas las actividades. Las actividades que especifiquen contenidos serán las que no sean compuestas (actividades atómicas). Además cuando se incluyan los contenidos en un curso, se deberá definir el identificador del fichero del enunciado de la actividad, por lo que es importante comprobar que todo contenido tenga asociado por lo menos una localización para su enunciado.

Para avisar al diseñador de una posible deficiencia en el diseño respecto a los contenidos de las actividades, se lanzarán advertencias en dos posibles casos: i) si la actividad no es compuesta y no tiene contenidos asociados a ella, ii) si la actividad no es compuesta y tiene contenidos asociados a ella pero alguno de ellos no dispone de localización para los mismos.

2.1.3.4. Herramientas colaborativas

Las actividades colaborativas son aquellas que han sido diseñadas para realizarse de manera grupal. Este tipo de actividades tendrán asociados recursos para facilitar tanto la interacción entre los miembros de los grupos de trabajo como la realización de la propia actividad. Este tipo de actividades nuevamente llevan asociadas un nuevo tipo de reglas, son las reglas de espacios colaborativos. Las reglas de espacios de trabajo colaborativos tienen asociado un identificador, el enunciado a presentar a los estudiantes sobre la actividad a resolver, y el conjunto de herramientas que se ofrecerá a los estudiantes para soportar el desarrollo de la actividad colaborativa. El principal problema puede surgir si en estas reglas no definen adecuadamente de qué herramientas se va a proveer a los estudiantes para realizar la actividad colaborativa.

Para analizar este tipo de inconsistencia, es necesario verificar en las actividades definidas como colaborativas, sus reglas de espacios de trabajo, comprobando que todas ellas dispongan de un identificador en su apartado de herramientas. Si esto no fuese así se lanzaría un error, avisando de esta manera al diseñador.

2.1.3.5. Actividades y dispositivos

Uno de los principales fines de los entornos adaptativos desarrollados para el sistema CoMoLE, es que estos se puedan utilizar desde múltiples dispositivos móviles. De esta forma se facilitará el trabajo de los usuarios y su aprendizaje se podrá realizar en cualquier momento y lugar. La adaptación a las características de los dispositivos se refiere al tipo de contenidos que soporta cada uno de los dispositivos y que se encuentra determinado por su propio *software* y *hardware* junto con las características de los usuarios.

Se puede dar la situación de que no se incluyan contenidos para todos los dispositivos especificados por el diseñador. Esta inconsistencia es considerada como una advertencia, ya que el diseñador no ha incluido los contenidos para todos los dispositivos.

Para solventar esta situación, se ha decido avisar al diseñador de qué actividades simples no tienen contenidos asociados para todos los tipos de dispositivos definidos en el curso. Esto implica solamente un aviso ya que el motor de recomendación del sistema CoMoLE se encarga de avisar al estudiante que una determinada actividad no se encuentra recomendada debido a que una característica de su contexto no es apropiada. En este ejemplo, la razón de la no recomendación de una actividad se debería a que no existen los contenidos apropiados.

2.1.4. Datos dinámicos

Los datos dinámicos son la información almacenada en el sistema sobre la interacción de otros usuarios que previamente han realizado actividades de un curso. Esto permite el que cuando no exista información suficiente sobre la adecuación de una determinada actividad para un usuario concreto que se encuentre en un contexto específico, se pueda efectuar una recomendación basada en información de usuarios similares. Además, se pueden utilizar estos datos dentro de las condiciones de las reglas de adaptación para recomendar o no determinadas actividades dependiendo de acciones previas de los usuarios. Por ejemplo, se podría recomendar actividades de refuerzo a aquellos usuarios

que no hubiesen superado con éxito unos determinados conceptos. Por ello es importante realizar una evaluación de diseño o funcionamiento de ciertos aspectos de los datos dinámicos, para detectar posibles errores en la evaluación del curso y así en el futuro poder realizar recomendaciones basadas en la información de otros usuarios con características similares de una forma segura.

2.1.4.1. Alta tasa de fallos en actividades realizadas

Los datos dinámicos nos pueden ayudar a detectar la mala construcción de las actividades o de los contenidos asociados a las mismas. Una manera de comprobar esta posible mala construcción, es detectar las actividades en las que se produzca una alta tasa de fallos. Si muchos usuarios fallan en un determinado ejercicio de tipo *test* o de rellenar un campo de texto (ejercicios de tipo "*fill in the blank*"), se debería informar de ello, ya que se puede estar dando una inconsistencia en dicha actividad. Este alto número de fallos puede ser provocado principalmente por dos circunstancias:

1. Que la solución dada por el profesor de la actividad sea incorrecta, y por eso estén fallando la mayoría de los alumnos.
2. Que el enunciado de la actividad sea muy avanzado o esté formulado de una forma incorrecta para un determinado perfil de usuario.

Por ello es importante avisar al diseñador de este tipo de error, ya que así podemos ayudarle a detectar el motivo que está originando que tantos estudiantes fallen en una determinada actividad. Este análisis se debe realizar analizando todos los datos dinámicos generados por los usuarios y detectando las actividades falladas por más de un 70 % de los estudiantes.

2.1.4.2. Adecuación de las actividades

Un aspecto importante de los entornos adaptativos es que exista un diálogo activo entre el sistema y los estudiantes. De esta forma el profesor podrá saber que opinan los estudiantes de las actividades que están realizando, y así en un futuro poder evolucionar y mejorar el curso diseñado en un principio. Por ello los estudiantes cuando finalicen una actividad podrán decidir si la recomendación ofrecida por el sistema sobre la

actividad les aparecido adecuada o no, quedando registrada dicha opinión en los datos dinámicos generados por el usuario.

Estas opiniones pueden analizarse para mejorar el diseño del curso. De esta manera, se avisará al diseñador de un posible error en las actividades que el sistema recomendó al estudiante, y que este último ha considerado por algún motivo que no fueron adecuadas.

2.1.4.3. Actividades no realizadas

Por último sería también importante sacar alguna conclusión sobre las actividades que el usuario no ha realizado. Este aspecto es importante para detectar dos causas principales:

- Se ha producido un bloqueo y el sistema no le ha permitido realizar dichas actividades. Esto puede venir provocado por una determinada regla estructural o por una regla individual.
- Las actividades han sido consideradas como no recomendadas debido a que una condición de contexto ha impedido la realización de las mismas, ya que ese contexto está estrechamente relacionado con las reglas estructurales o individuales de las actividades no realizadas.

La evaluación de las actividades no realizadas por los estudiantes nos permitiría avisar de este tipo de error al profesor, para así solucionar de alguna manera estos bloqueos, y que de esta forma el usuario pudiese realizar todo el conjunto de actividades que componen el curso.

2.2. Metodología

La metodología empleada para el desarrollo del proyecto ha sido el **Proceso Unificado de Desarrollo**. Dicha metodología es un marco de desarrollo de *software* que se caracteriza por estar dirigido por casos de uso, centrado en la arquitectura y por ser iterativo e incremental. El Proceso Unificado no es simplemente un proceso, sino un

marco de trabajo extensible que puede ser utilizado para una gran cantidad de tipos de sistemas de *software*, para diferentes áreas de aplicación, diferentes tipos de organizaciones, diferentes niveles de competencia y diferentes tamaños de proyectos. Gracias a él se provee a los proyectos *software* de un enfoque disciplinado en la asignación de tareas, y su meta es asegurar la producción de *software* de alta calidad que satisfaga las necesidades de los usuarios finales.

Las principales características de esta metodología son:

- Dirigido por casos de uso: Un sistema de *software* se crea para servir a sus usuarios. Por lo tanto, para construir un sistema exitoso se debe conocer qué es lo que quieren y necesitan los usuarios prospectos. Un caso de uso es una pieza en la funcionalidad del sistema que le da al usuario un resultado de valor. Los casos de uso capturan los requerimientos funcionales, y juntos constituyen el modelo de casos de uso el cual describe la funcionalidad completa del sistema. Sin embargo, los casos de uso no son solamente una herramienta para especificar los requerimientos del sistema, sino que también están dirigidos su diseño, implementación y pruebas, esto es, dirigen el proceso de desarrollo. Los casos de uso son desarrollados a la par que la arquitectura del sistema, es decir, que estos dirigen la arquitectura del sistema y la arquitectura del sistema influencia la elección de los casos de uso.

- Centrado en la arquitectura: El papel del arquitecto de sistemas es similar en naturaleza al papel que el arquitecto desempeña en la construcción de edificios. El edificio se mira desde diferentes puntos de vista: estructura, servicios, plomería, electricidad, etc. Esto le permite al constructor ver una radiografía completa antes de empezar a construir. Similarmente, la arquitectura en un sistema de *software* es descrita como diferentes vistas del sistema que está siendo construido. El concepto de arquitectura de *software* involucra los aspectos estáticos y dinámicos más significativos del sistema. La arquitectura es la vista del diseño completo con las características más importantes hechas más visibles y dejando los detalles de lado. La arquitectura debe proveer espacio para la realización de todos los casos de uso, hoy y en el futuro, por lo que arquitectura y casos de uso deben evolucionar en paralelo.

- Iterativo e incremental: Desarrollar un producto de *software* es una tarea enorme que puede continuar por varios meses o años. Es práctico dividir el trabajo en pequeñas partes o mini-proyectos. Cada mini-proyecto es una iteración que finaliza en un incremento. Las iteraciones se refieren a pasos en el flujo de trabajo, los incrementos se refieren a crecimiento en el producto. Para ser más efectivo, las iteraciones deben estar controladas, esto es, deben ser seleccionadas y llevadas a cabo de una manera planeada. Los desarrolladores basan su selección de qué van a implementar en una iteración en dos factores: primero, la iteración trata con un grupo de casos de uso que en conjunto extienden la usabilidad del producto, y segundo, la iteración trata con los riesgos más importantes. Las iteraciones sucesivas construyen los artefactos del desarrollo a partir del estado en el que fueron dejados en la iteración anterior. En cada iteración, los desarrolladores identifican y especifican los casos de uso relevantes, crean el diseño usando la arquitectura como guía, implementan el diseño en componentes y verifican que los componentes satisfacen los casos de uso. Si una iteración cumple sus metas el desarrollo continúa con la siguiente iteración. Cuando la iteración no cumple con sus metas, los desarrolladores deben revisar sus decisiones previas y probar un nuevo enfoque.

El uso de ésta metodología ha resultado muy importante a lo largo del desarrollo del proyecto, ya que ha servido de guía para ordenar las actividades que había que realizar. Además dicha metodología ha permitido establecer un plan para el proyecto y ha ayudado a decidir la arquitectura correcta para el desarrollo de la aplicación.

Capítulo 3 : Descripción informática

3.1. Especificación

El *software* desarrollado en el presente proyecto será una aplicación para la evaluación de entornos adaptativos dentro de CoMoLE. Con él se pretende validar si un entorno adaptativo creado por un profesor o diseñador, está correctamente construido. En caso de que esto no fuese así, el sistema se encargará de dar la información apropiada al diseñador, para que así este pueda corregir los aspectos que considere oportunos.

Los datos de información del entorno adaptativo se encuentran almacenados en una serie de ficheros que previamente han sido generados a través de la herramienta de autor, o de forma manual por el diseñador del curso. Nuestra aplicación se encargará de analizar dichos ficheros, y extraer de ellos las conclusiones oportunas.

El funcionamiento general se presenta a continuación. El primer paso es leer dichos ficheros de datos. Para facilitar el trabajo se ha pensando que la aplicación simplemente reciba un único fichero comprimido con todos los ficheros de datos que genere la herramienta de autor o que cree de forma manual el diseñador. Este fichero comprimido deberá contener una carpeta que contenga toda la estructura de datos con la información necesaria que necesita el sistema CoMoLE. En este fichero comprimido puede especificarse información de varios cursos a la vez (siempre que se respete la estructura de datos del sistema) para que de esta forma se pueda evaluar un curso u otro según decida el diseñador.

Tras proporcionar los datos a la aplicación, el sistema comprobará que el fichero facilitado sea correcto. Si éste no lo fuese se avisaría de lo ocurrido. En caso de que todo el proceso haya transcurrido satisfactoriamente, el diseñador o profesor deberá decidir qué curso desea evaluar, y qué aspectos se quieren verificar. Para ello, se deberá

elegir el nombre del curso que se desea evaluar, y seleccionar el tipo de inconsistencias que se quieren analizar.

Una vez seleccionados estos aspectos, la aplicación se encargará en primer de lugar de leer la información almacenada en distintos ficheros sobre el curso que se ha decido evaluar. Si se produjese algún fallo en la lectura de estos ficheros (por ejemplo porque no se dispongan de ciertos ficheros de datos del curso indicado, o por una mala construcción de estos ficheros), se avisará de ello. Si la lectura de los datos transcurre sin ninguna anomalía, se procederá a evaluar el tipo de inconsistencias que se desea evaluar en dichos ficheros. Está evaluación será llevada a cabo en función de los criterios explicados en el apartado de objetivos del presente documento.

Tras ello llegará el momento de mostrar al diseñador la información recabada de la evaluación del curso. Se indicará al diseñador las inconsistencias (errores y advertencias) que se hayan detectado durante el análisis del curso especificado. También se permitirá visualizar y descargar el/los fichero/s de datos donde se han detectado inconsistencias. De esta manera, el diseñador podrá modificar el fichero y almacenarlo una vez corregido.

La aplicación desarrollada deberá ser facilitar la realización de las siguientes tareas fundamentales:

- Recoger los datos del fichero comprimido que contiene todos los datos necesarios para poder llevar a cabo la evaluación del entorno adaptativo.
- Permitir indicar que curso se desea evaluar, y en base a qué criterios se desea llevar a cabo dicha evaluación.
- Detectar las inconsistencias encontradas durante la evaluación del entorno adaptativo.
- Ofrecer al diseñador la información necesaria para poder entender el motivo de las inconsistencias halladas.
- Facilitar la visualización y descarga de los ficheros que presenten problemas, para que el diseñador pueda modificarlos a su conveniencia.

El presente proyecto deberá proporcionar una propuesta de desarrollo para cada una de las partes implicadas en la evaluación de entornos adaptativos para CoMoLE. Se podría realizar la siguiente clasificación sobre las partes o módulos que debería contener esta aplicación:

- Módulo que permita la recogida del fichero comprimido con los datos del curso.
- Módulo que permita decidir el curso a evaluar y en qué términos hacerlo.
- Módulo encargado de leer la información asociada al curso oportuno.
- Módulo confiado a realizar los diferentes análisis de los datos.
- Módulo de presentación de inconsistencias.
- Módulo que permita la visualización y descarga de los ficheros de datos que hayan presentado problemas a lo largo del proceso de evaluación.

A continuación presentamos el principal escenario de uso de la aplicación desarrollada (ver Figura 1). En él, un diseñador decide evaluar un entorno adaptativo para saber si este está correctamente construido.

Figura 1: Caso de uso principal de la aplicación

El siguiente escenario nos muestra como el diseñador facilita a la aplicación los ficheros de datos, y como éstos son gestionados (ver Figura 2).

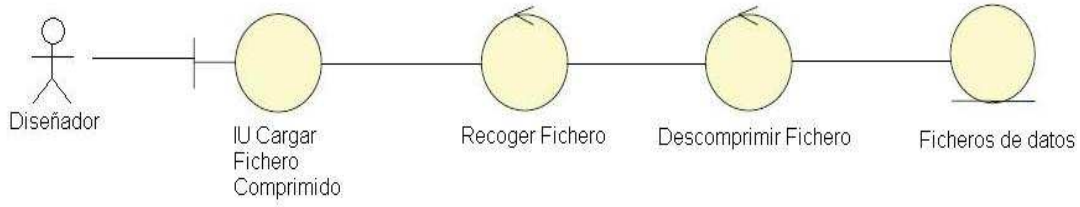

Figura 2: Caso de uso donde el diseñador facilita a la aplicación el ficheros de datos

La Figura 3 representa el caso de uso que se corresponde con la evaluación de los ficheros de datos, gracias a la cual el sistema obtendrá las conclusiones oportunas sobre las inconsistencias encontradas en los ficheros.

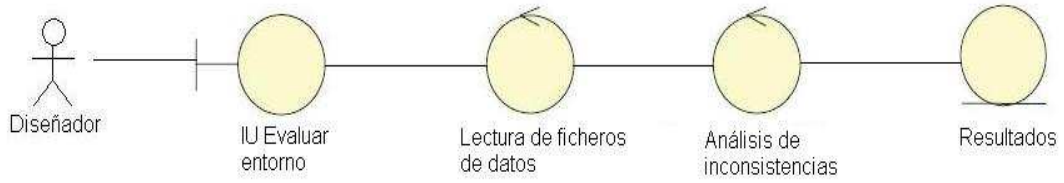

Figura 3: Caso de uso para la evaluación de un curso

El escenario donde un diseñador visualizaría las inconsistencias detectadas durante la evaluación del entorno adaptativo, estaría reflejado por el caso de uso de la Figura 4.

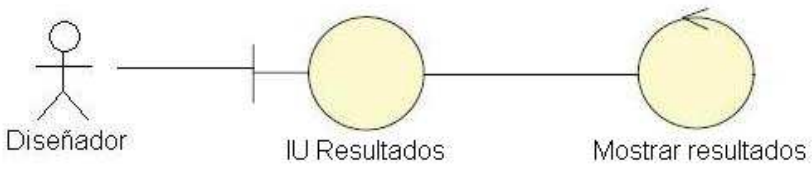

Figura 4: Caso de uso para visualizar los resultados de la evaluación

Por último, el diagrama de la Figura 5 nos permitirá representar cómo el diseñador podrá ver y descargar los ficheros de datos que contengan las inconsistencias detectadas por nuestro sistema de evaluación.

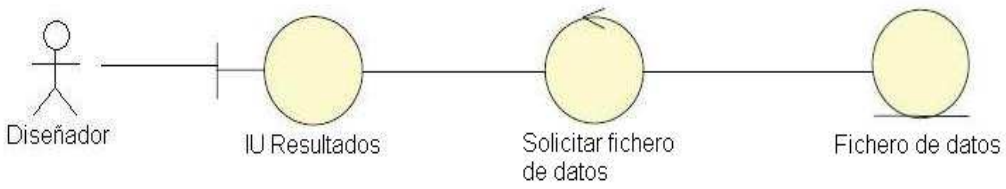

Figura 5: Caso de uso para la descarga de un fichero

3.2. Arquitectura de la aplicación

La aplicación propuesta en el proyecto es una **aplicación Web**. Esta decisión ha sido tomada debido a que de esta forma los usuarios podrán utilizarla accediendo simplemente a un servidor Web (ya sea a través de Internet o de una intranet) mediante un navegador Web. Este hecho nos permite aprovechar el potencial del navegador Web como cliente ligero, facilitando de esta manera la actualización y mantenimiento de la aplicación sin necesidad de tener que distribuir e instalar el *software* a los usuarios. También es importante señalar que la aplicación al mostrarse como una página Web dinámica, permite mantener una comunicación activa entre el usuario y la información. De esta forma el usuario puede acceder a los datos de un modo interactivo, ya que la página responderá de forma diferente a cada una de sus acciones.

La arquitectura de la aplicación desarrollada se basa en el patrón **Modelo-Vista-Controlador**.. Este patrón nos permite separar el modelo de datos de la aplicación, de su representación de cara al usuario y de la interacción de éste con la aplicación, mediante la división de la aplicación en tres partes fundamentales:

- El modelo, que contiene la lógica de negocio de la aplicación.
- La vista, que muestra al usuario la información que éste necesita.
- El controlador, que recibe e interpreta la interacción del usuario actuando sobre modelo y vista de manera adecuada para provocar cambios de estado en la representación interna de los datos, así como en su visualización.

Esta arquitectura ha demostrado ser muy apropiada para las aplicaciones Web y especialmente adaptarse bien a las tecnologías proporcionadas por Java de manera que:

- El modelo, conteniendo la lógica de negocio, sería modelado por un conjunto de clases Java.
- La vista proporcionará una serie de páginas Web al cliente, que serán construidas de manera dinámica. Existen múltiples maneras de generar estas páginas Web, la utilizada en el presente proyecto ha sido el uso de páginas JSP (*Java Server Pages*),
- El controlador se ha implementado mediante *servlets*, que hacen de intermediarios entre la vista y el modelo.

Con todo lo anterior, el funcionamiento de una aplicación Web implementada con el patrón MVC se puede descomponer en una serie de pasos:

1. El usuario realiza una acción en su navegador, que llega al servidor mediante una petición HTTP y es recibida por un *servlet* (controlador). Esa petición es interpretada y se transforma en la ejecución de código Java que delegará al modelo la ejecución de una acción de éste.
2. El modelo recibe las peticiones del controlador, a través de una interfaz que encapsulará y ocultará la complejidad del modelo al controlador. El resultado de esa petición será devuelto al controlador.
3. El controlador recibe del modelo el resultado, y en función de éste, selecciona la vista que será mostrada al usuario, y le proporcionará los datos recibidos del modelo y otros datos necesarios para su transformación a HTML. Una vez hecho esto, el control pasa a la vista para la realización de esa transformación.
4. En la vista se realiza la transformación tras recibir los datos del controlador, elaborando la respuesta HTML adecuada para que el usuario la visualice.

Esta arquitectura de aplicaciones otorga varias ventajas claves para el desarrollo de aplicaciones Web. Cabe destacar que al separar de manera clara la lógica de negocio (modelo) de la vista, se permite la reusabilidad del modelo de modo que la misma

implementación de la lógica de negocio que maneja una aplicación pueda ser usado en otras aplicaciones, sean éstas Web o no.

3.2.1. Tecnologías utilizadas

Para realizar el modelo de datos se ha utilizado UML (*Unified Modeling Language*), un lenguaje de modelado que permite visualizar, especificar, construir y documentar un sistema *software*. Ofrece un estándar para describir una especie de "plano" del sistema, incluyendo aspectos conceptuales tales como: procesos de negocio y funciones del sistema, aspectos concretos como expresiones de lenguajes de programación, esquemas de bases de datos y componentes reutilizables. Es importante resaltar que UML permite especificar y describir métodos o procesos, y que se utiliza para definir un sistema, para detallar los artefactos en el sistema y para documentar y construir el mismo.

La información necesaria para realizar los diferentes análisis que lleva a cabo la aplicación, se ha almacenado en ficheros de datos con formato XML (*Extensible Markup Language*).

Como la aplicación a desarrollar va a ser una aplicación Web se ha decidido utilizar el lenguaje Java. Java es lenguaje de programación orientado a objetos cuya principal característica es que las aplicaciones escritas en este lenguaje están típicamente compiladas en un *bytecode* que en tiempo de ejecución es normalmente interpretado o compilado por su máquina virtual, proporcionando código nativo listo para su ejecución.

Las aplicaciones desarrolladas con *servlets* necesitan un servidor Web que responda ante nuestras peticiones, por ello se ha utilizado un servidor Apache que es un servidor HTTP de código abierto y multiplataforma. Una de las principales ventajas del servidor Apache es que su arquitectura es muy modular. De esta manera el servidor consta de una sección *core* y diversos módulos que aportan mucha de la funcionalidad que podría considerarse básica para un servidor Web. Además en el proyecto se ha empleado Tomcat para dar soporte a los servlets y JSPs.

Por último, se ha utilizado HTML y JavaScript para presentar la interfaz de nuestra aplicación. HTML (*HyperText Markup Language*), es el lenguaje de marcado generalmente usado en la elaboración de páginas Web. Es usado para describir la estructura y el contenido en forma de texto, así como para complementar el texto con objetos tales como imágenes. HTML utiliza "etiquetas", para describir, hasta un cierto punto, la apariencia de un documento. Por su parte JavaScript es un lenguaje de *scripting* orientado a objetos utilizado para acceder a objetos en aplicaciones. Principalmente, se utiliza integrado en un navegador Web permitiendo el desarrollo de interfaces de usuario mejoradas y páginas Web dinámicas.

3.3. Diseño de la aplicación

El diseño de la aplicación ha sido llevado a cabo en función de la arquitectura descrita en el apartado anterior. Se ha utilizado un diseño orientado a objetos debido a la calidad del *software* escrito bajo este paradigma, ya que contribuye a que los programas sean más flexibles, modulares, reutilizables y comprensibles. Esto nos permite que los programas diseñados bajo el paradigma de orientación a objetos, eleven su calidad interna y externa. Las clases de objetos nos facilitan nuevos medios para encapsular abstracciones, dividir la realidad, y disminuir la complejidad.

A continuación se detallará cada uno de los componentes que forman parte de la aplicación, su estructura y funcionamiento.

3.3.1. Capa de lógica de negocio

La lógica de negocio es la capa intermedia que maneja el intercambio de información entre la vista y el modelo; es decir, el controlador. Sin embargo en este proyecto se define la lógica de negocio como un término que no engloba la transferencia de información entre capas, sino a la lógica aplicativa que permite el correcto funcionamiento del sistema. Por ello la lógica de negocio del proyecto se corresponde con la parte de nuestro sistema encargada de realizar el análisis de inconsistencias de los ficheros de datos. Con este objetivo nuestra aplicación dispondrá de una serie de clases que le permitirán llevar a cabo dicha tarea. Es importante señalar que aparte de las

clases que vamos a describir (las cuales han sido desarrolladas en el presente proyecto), también se han utilizado ciertas clases de objetos implementadas en el sistema CoMoLE que nos permiten dar soporte a algunos aspectos de nuestra aplicación.

Las clases involucradas en la evaluación de entornos adaptativos son principalmente cuatro: inconsistencias, lecturas, ficheros y codigosError. Gracias a estas clases obtendremos todas las posibles inconsistencias existentes en un determinado entorno.

La clase **Inconsistencias** es la clase fundamental de nuestro sistema. Se encargará del análisis de todas las inconsistencias estudiadas en el capítulo anterior sobre la motivación del trabajo. En esta clase se encontrarán implementados todos los métodos que hacen posible la evaluación de un entorno adaptativo, y que sin los cuales nuestro sistema carecería de funcionalidad.

La Figura 6 muestra como está implementada la clase. Al diseñarla se ha considerado oportuno que tenga como atributos toda la configuración de un entorno adaptativo, con todo lo que ello conlleva: tipos, actividades, reglas, descripciones, idiomas, etc. De esta forma se facilitará el desarrollo de los métodos encargados del estudio de inconsistencias, y se dejará la puerta abierta a la posibilidad de incluir métodos encargados del análisis de nuevos problemas. Para ello la clase incluye tres atributos donde se recoge toda la configuración de un curso:

- tipos_i: En este atributo se almacenarán todos los tipos de actividades permitidas en curso, todos los lugares permitidos para la realización de las actividades, todos los dispositivos que se pueden emplear para realizar el curso, las posibles guías que se pueden seguir al realizar una actividad colaborativa, y las herramientas que se podrán utilizar para realizar actividades colaborativas.

- meta_a_i: Este atributo recogerá toda la configuración interna de un entorno adaptativo. Gracias a este atributo podremos disponer de las reglas de adaptación que se seguirán en el curso, los idiomas y descripciones que tiene, los diferentes rasgos en base a los que realizar las recomendaciones, un listado de las actividades que compone el curso y toda la información asociada a cada una de ellas, y el tamaño de los grupos para las actividades colaborativas..

- contexto_i: Con este atributo se consigue disponer de las reglas de contexto disponibles para un curso, y que resultan de vital importancia a la hora de realizar la recomendación de actividades basada en el tipo de estas.

Gracias a estos atributos esta clase implementará la estructura interna de un entorno adaptativo, y nos servirá de base a la hora de realizar el análisis de inconsistencias.

De los métodos de esta clase, tan solo señalar que mediante todos los "Comprueba" se realizará el análisis exhaustivo que este proyecto demanda, y que el método "VerificarInconsistencia" resulta de importancia a la hora de determinar si un curso adaptativo ha sido leído de manera correcta.

La clase **Lecturas** permite hacer las lecturas de todos los datos de un entorno adaptativo, para que de esta forma dispongamos de toda la información necesaria para poder realizar la evaluación del curso. Por ello en esta clase se implementan todos los métodos necesarios para poder realizar dichas lecturas, proporcionando de esta manera una forma sencilla de obtener toda la información almacenada en los ficheros de datos del sistema CoMoLE.

La Figura 7 muestra como ha sido implementada la clase Lecturas. Esta implementación sólo necesita de unos métodos que permitan realizar las lecturas del curso, y que devuelvan los atributos utilizados en la clase Inconsistencias.

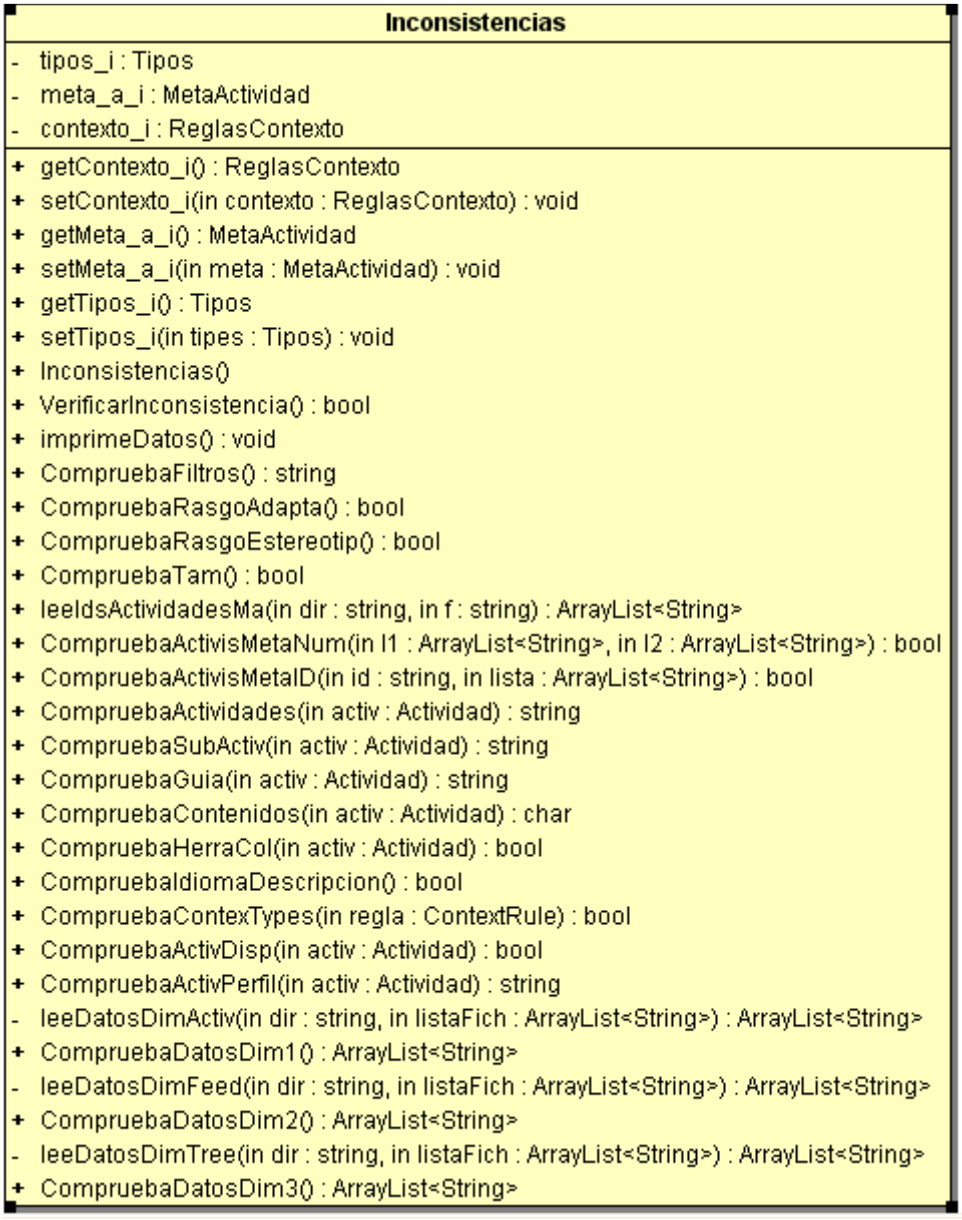

Figura 6: Clase Inconsistencias

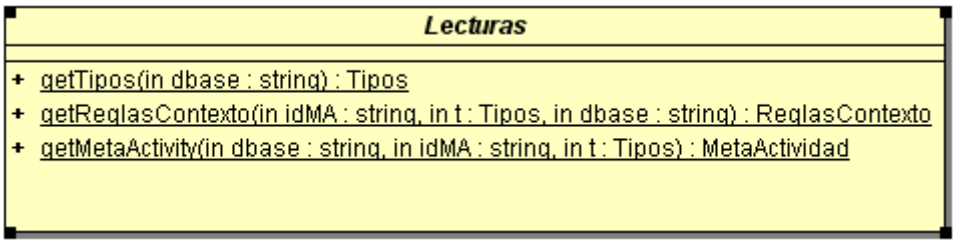

Figura 7: Clase Lecturas

La clase **Ficheros** es necesaria debido a que los ficheros de datos de un entorno adaptativo para CoMoLE están organizados de una determinada forma. La herramienta de evaluación necesitará conocer dicha organización para poder realizar las lecturas de dichos ficheros. Por ello se ha diseñado esta clase, en la cual implementaremos todos los aspectos necesarios para mantener un control de dicha organización. Además esta clase será la que contenga la ruta donde se almacenan todos los ficheros de datos, y sobre la que trabajará todo el proceso de lectura de datos. La Figura 8 tanto los atributos de esta clase como los métodos gracias a los cuales se tendrá acceso a la estructura mencionada.

Figura 8: Clase Ficheros

Por último el proyecto implementará la clase **CodigosError** ya que cada tipo de inconsistencia tendrá asociada (si lo requiere) unos determinados códigos que permitirán identificar la inconsistencia que se está produciendo. De esta manera se consigue que si por algún motivo se decide realizar alguna modificación en las inconsistencias, tan solo sea necesario añadir cuales son los nuevos códigos.

En la Figura 9 reflejamos los códigos de error que hemos implementado para las inconsistencias que analizará la aplicación dentro de la clase CodigosError..

Figura 9: Clase CodigosError

El diagrama de clases de la Figura 10 refleja las relaciones existentes entre las diferentes clases explicadas anteriormente. Se puede observar como la clase Inconsistencias tiene una relación de uso con cada una de las clases involucradas en el diagrama, ya que esta clase utiliza al resto para llevar a cabo sus tareas. Además es importante señalar la relación de asociación existente entre la clase Inconsistencias y la clase Lecturas, ya que para construir de forma correcta un objeto Inconsistencia resulta fundamental la intervención de la clase Lecturas.

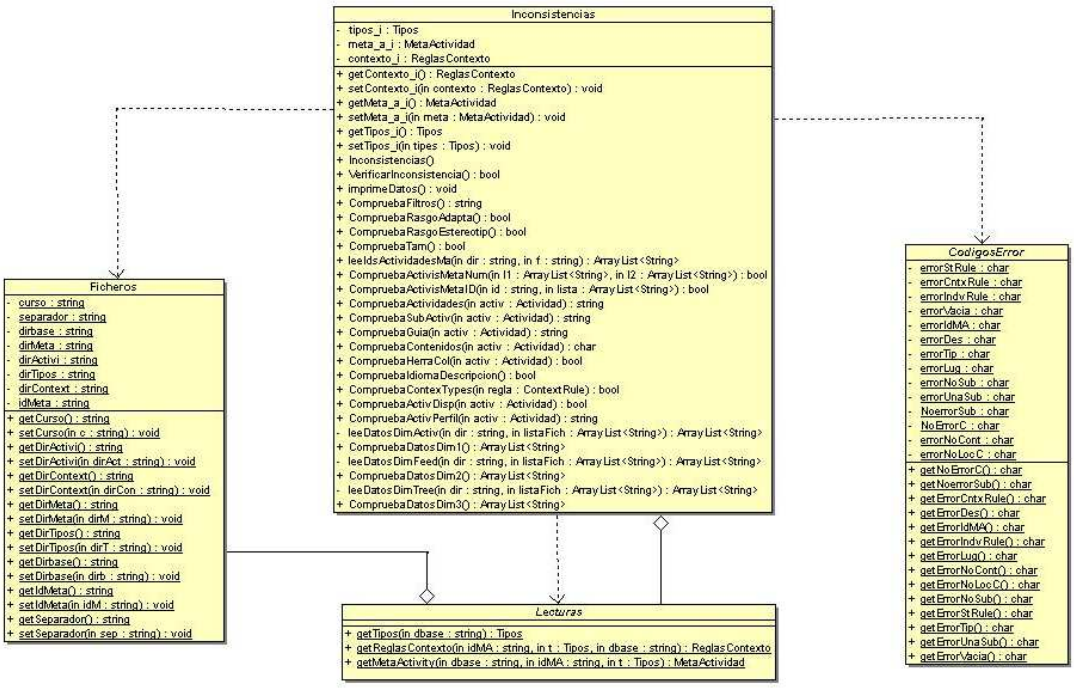

Figura 10: Diagrama de clases de la aplicación

3.3.2. Controlador

El controlador será el objeto encargado de dirigir el flujo de control de la aplicación en función de los mensajes externos recibidos, como pueden ser datos introducidos por el usuario u opciones del menú seleccionadas por él. A partir de estos mensajes, el controlador se encargará de modificar el modelo y/o de abrir y cerrar vistas. El controlador tiene acceso al modelo y a las vistas, pero las vistas y el modelo no se conocen entre sí. La implementación del controlador se ha realizado utilizando diferentes *servlets* que se encargarán de realizar las siguientes operaciones:

- Almacenar en el servidor el fichero comprimido con los datos del entorno adaptativo el cual es suministrado por el diseñador o profesor del curso.
- Descomprimir el fichero para obtener los ficheros de datos necesarios con el objetivo de realizar la evaluación.
- Realizar la evaluación del entorno adaptativo utilizando las clases del modelo.
- Presentar la vista correspondiente en función de los criterios seleccionados por el usuario.
- Permitir al usuario la descarga de los ficheros de datos.

A continuación se presentan una serie de diagramas de secuencia que ayudarán a entender las funciones correspondientes al controlador del sistema. Con estos diagramas se pretende mostrar de forma clara, la secuencia cronológica de las interacciones entre las clases que intervienen en los procesos involucrados en esta parte.

La Figura 11 muestra el funcionamiento del sistema Modelo-Vista-Controlador. En ella se puede observar como el usuario interactúa con la vista desencadenando la entrada en escena del controlador, quien será el encargado de hacer la petición correspondiente al modelo. Una vez que se haya producido las operaciones convenientes, se mostrará al usuario la vista originada.

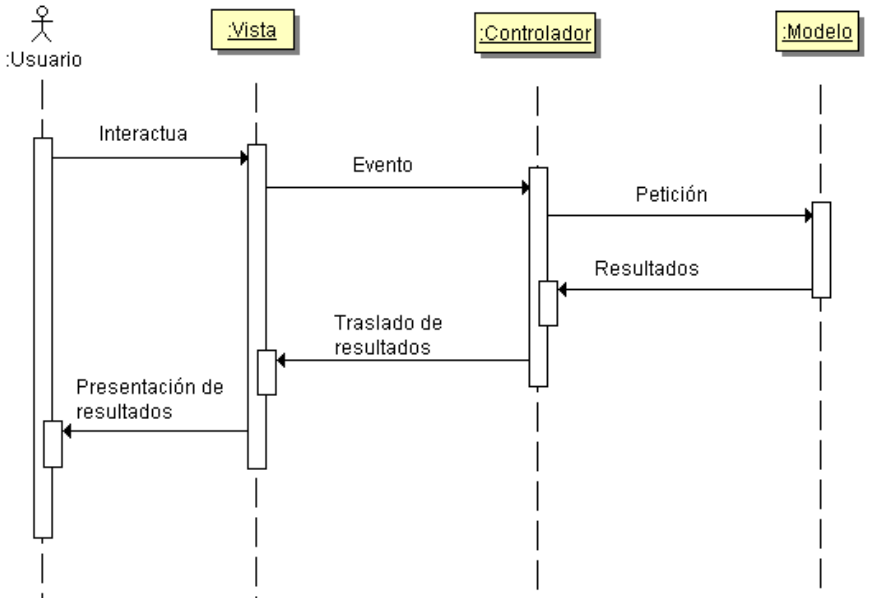

Figura 11: Diagrama de secuencia del patrón Modelo-Vista-Controlador

En la Figura 12 se puede observar el diagrama de secuencia correspondiente a la subida de un fichero comprimido al servidor. En este caso el diseñador facilita a la aplicación el fichero comprimido que contiene los ficheros de datos de los entornos adaptativos que se desea evaluar. Este fichero se envía a un *servlet* (Inicio), que será el encargado de almacenar el fichero en el servidor del sistema. Posteriormente descomprimirá el fichero, obteniendo los ficheros de datos del curso a analizar, los cuales estarán preparados para ser leídos por nuestra aplicación.

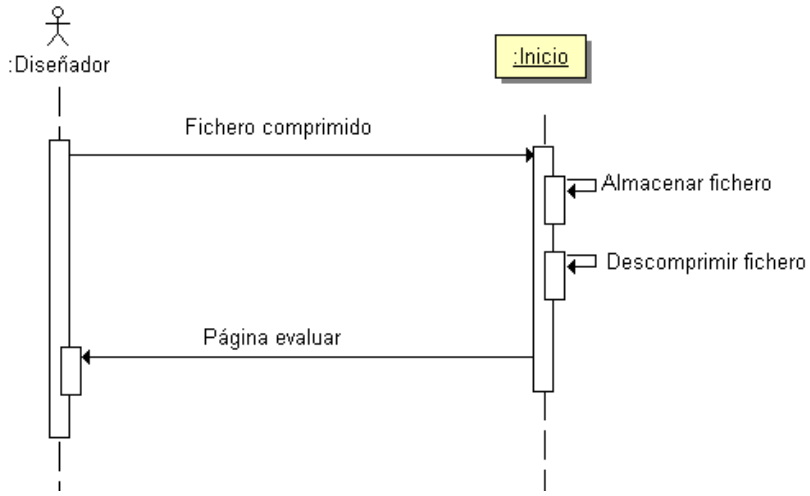

Figura 12: Diagrama de secuencia a la hora de subir fichero al servidor

Una vez facilitados los datos se procede a la evaluación del entorno. El diagrama de secuencia de la Figura 13 refleja el proceso correspondiente que realiza el *servlet* "Evaluador", ante la solicitud por parte de un diseñador de la evaluación de un determinado entorno adaptativo. Éste se encargará de realizar las operaciones necesarias para poder leer el curso correspondiente, y detectar sus posibles inconsistencias.

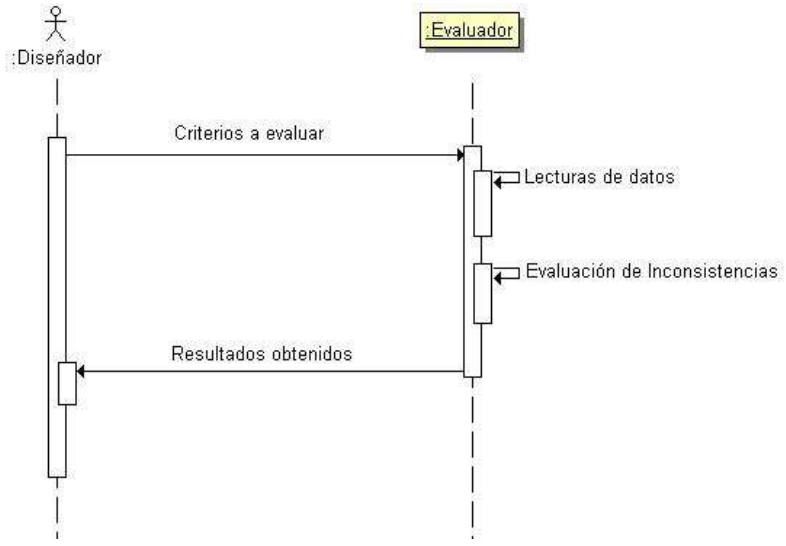

Figura 13: Diagrama de secuencia del caso de evaluación de un curso

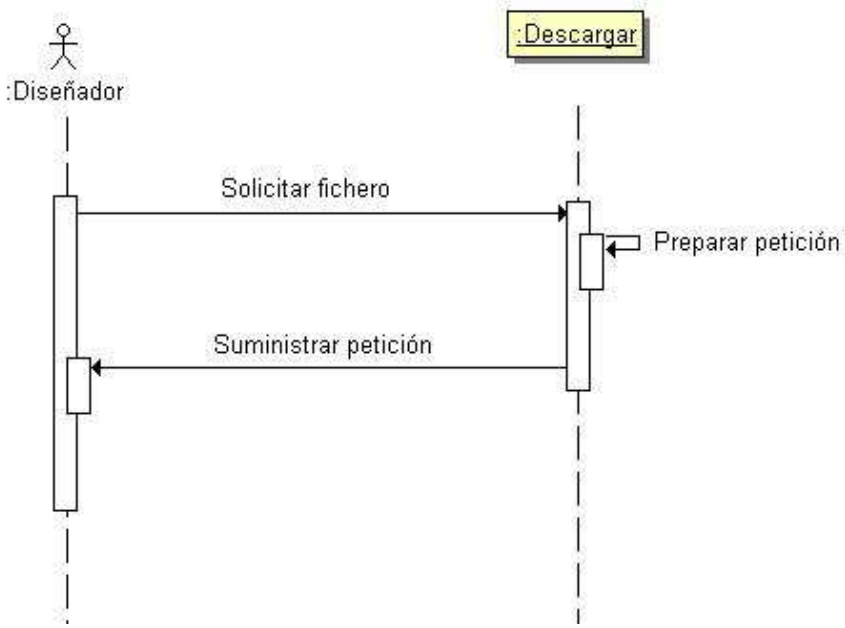

Figura 14: Diagrama de secuencia de descargar fichero

La Figura 14 refleja el diagrama de secuencia correspondiente a la descarga de ficheros de datos. Este *servlet* entrará en escena cuando el error o advertencia sobre el diseño del curso requiera al sistema un fichero de datos, para poder ver de esta forma el fallo de una manera más clara, o para corregir el problema. Ante ello el servidor permitirá la descarga del fichero solicitado, delegando en el diseñador la responsabilidad del mismo.

Por último, cuando el usuario decida salir de la aplicación o volver al inicio, el sistema deberá borrar todos los ficheros de datos que hayan podido quedar almacenados en el servidor. Si esto no se realizase sería un gran problema, ya que quedaría almacenada información innecesaria en el servidor. El *servlet* Cierre se encarga de esta tarea. La Figura 15 refleja el diagrama de secuencia correspondiente al borrado de estos ficheros de datos.

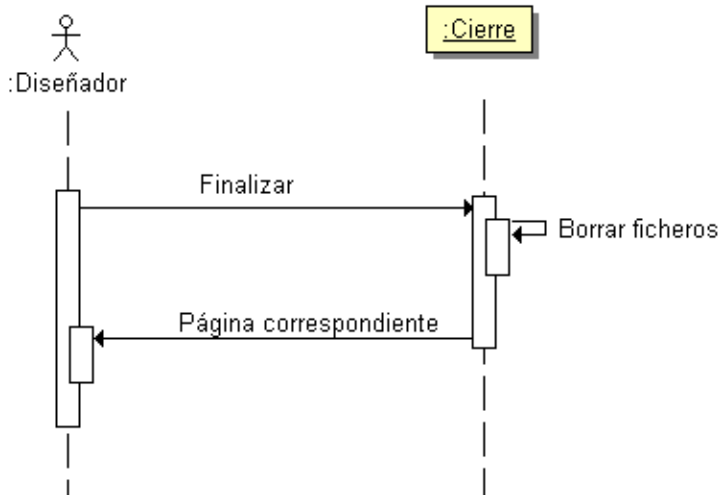

Figura 15: Diagrama de secuencia de finalización de la evaluación

3.3.3. Capa de presentación

La capa de presentación dota a la aplicación desarrollada de la navegabilidad necesaria para poder realizar las peticiones oportunas y avisar al sistema de las tareas que se deben llevar a cabo. Será la encargada de dotar de una interfaz gráfica al usuario, en la cual habrá una comunicación bidireccional entre la aplicación y el usuario. Así mismo también se realizará en esta capa, la presentación adecuada de los datos de salida, proporcionando al usuario todos los resultados obtenidos de la evaluación de inconsistencias (ver Figura 16).

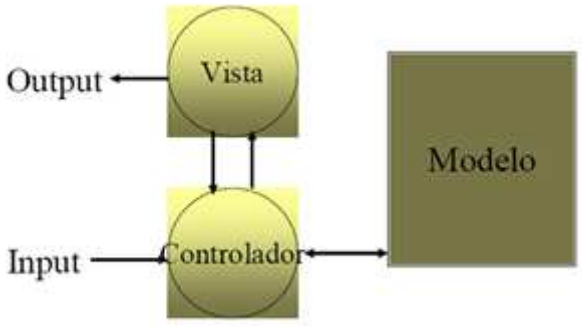

Figura 16: Comunicación mediante el patrón MVC

El diseño de la interfaz de usuario se ha realizado con el objetivo de realizar una aplicación lo más amistosa posible, que no entrañe dificultades de uso a los usuarios, y que sea lo más eficiente posible.

A continuación se hará una breve explicación de las páginas más importantes que forman parte de la aplicación. Las explicaciones están acompañadas de unas capturas de pantalla para que estas sean lo más claras y detalladas posibles.

La página de bienvenida (ver Figura 17) nos permitirá indicar el fichero en el cuál se encontrarán almacenados los datos del entorno adaptativo. Es importante señalar que el fichero debe ser un fichero comprimido en formato ZIP. De esta forma la simplicidad de la página es máxima, ya que el diseñador sólo debe indicar el fichero y continuar con el proceso.

Figura 17: Página de inicio de la aplicación desarrollada

Si el fichero es correcto, se mostrará al usuario la página de evaluación de inconsistencias (ver Figura 18). En ella tan sólo se deberá seleccionar el nombre del entorno adaptativo que se desea evaluar, y seleccionar también los criterios en base a los cuales se quiere realizar la evaluación de inconsistencias. Tras estas sencillas tareas, el usuario podrá continuar con la evaluación del curso.

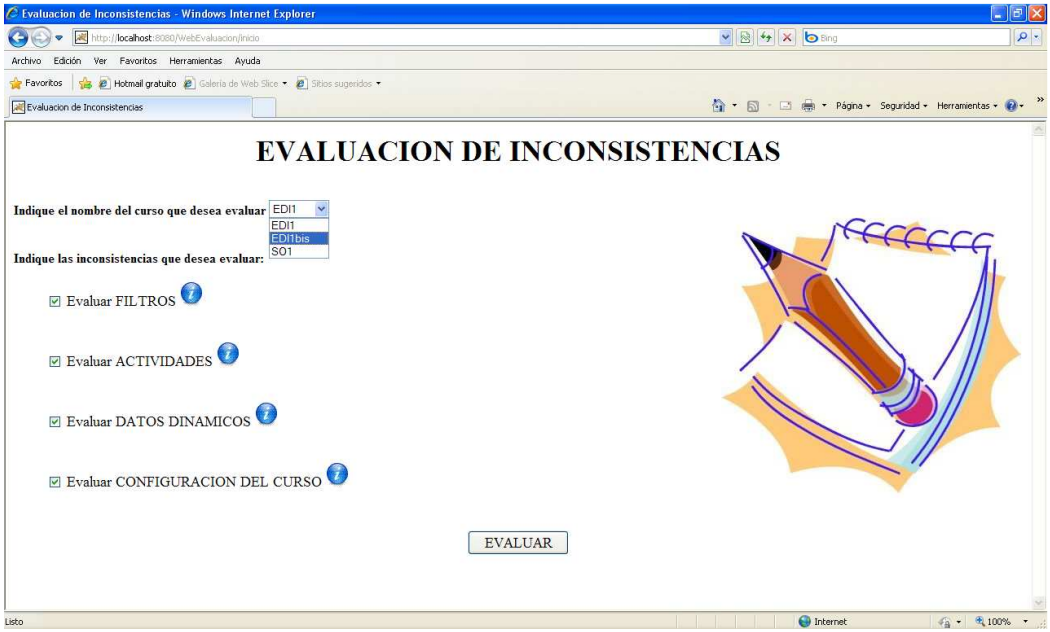

Figura 18: Página de evaluar

Por último, se mostrarán los resultados obtenidos de la evaluación de inconsistencias (ver Figura 19) relacionada con los criterios seleccionados por el diseñador en la pantalla anterior. En ellos se mostrarán los errores y advertencias acompañados la información necesaria para connocer el por qué del problema. Además se tendrá la opción descargar los ficheros de datos que están provocando los inconvenientes, para poder ver más claramente lo que está pasando, y modificarlos si se quiere.

Figura 19: Página de resultados

3.4. Pruebas

Las pruebas permiten garantizar la calidad del *software* y nos sirven como revisión final de la especificación, diseño y codificación de nuestra aplicación. Para comprobar el correcto funcionamiento de la misma, se han realizado un gran número de pruebas, para así asegurar que se cumplen todos los objetivos planteados. A continuación se algunas de las pruebas llevadas a cabo:

1. A la hora de realizar el análisis de inconsistencias se ha comprobado que los resultados obtenidos para cada una de ellas eran los esperados.
2. Se ha verificado que el fichero comprimido facilitado por eldiseñador tenga el formato adecuado y respecte la estructura de datos apropiada.
3. También se ha comprobado que si los ficheros de datos que se evalúan tienen algún tipo de anomalía, se avise de que el análisis no se podrá realizar debido a esta circunstancia.
4. Se ha asegurado que la descarga de ficheros del servidor, funciona de manera correcta. El servidor devuelve al navegador Web el fichero solicitado, para que el diseñador decida qué hacer con él.
5. Por último se ha comprobado (de manera manual) que cuando se sale de la aplicación, los ficheros de datos no quedan almacenados en el servidor.

Todas estas pruebas se han realizado utilizando como base datos de cursos previamente diseñados para el sistema CoMoLE donde se encontraban definidos tres entornos de actividades educativas distintos.

Capítulo 4 : Conclusiones

4.1. Logros alcanzados

La valoración final del proyecto es muy positiva, ya que se ha conseguido desarrollar satisfactoriamente un *software* que cumple con todos los objetivos planteados en un principio.

Por un lado, se ha conseguido satisfacer el principal objetivo del proyecto que es el de evaluar entornos adaptativos para el sistema CoMoLE. Se ha ofrecido una propuesta que evalúa el correcto diseño de estos entornos informando al diseñador de los posibles problemas que se hayan podido encontrar en dicho análisis.

Además de avisar de los problemas que existan debido al diseño del curso, también se han evaluado ciertos aspectos relacionados con los datos dinámicos existentes en los ficheros de datos. De esta manera, ayudaremos al diseño de futuros entornos adaptativos.

Es importante mencionar que también se ha facilitado la corrección de las inconsistencias que existan en los cursos evaluados. Esto se ha conseguido permitiendo la descarga del fichero de datos que plantee el problema, ayudando de esta manera a que el diseñador o profesor pueda realizar la corrección del fichero en el momento, y almacenar si lo considera oportuno, el fichero corregido.

Otro aspecto importante que se ha conseguido, ha sido que la aplicación sea fácil e intuitiva, beneficiando de esta manera, el uso de la misma por parte de todo tipo de profesores y diseñadores.

Por último considero importante que la aplicación haya sido desarrollada como una aplicación Web, ya que con ello se facilita el acceso de la misma.

4.2. Trabajos futuros

Pese a que el proyecto ha cumplido con todos los objetivos marcados en un principio, se podrían introducir varias mejoras que aumentasen la utilidad y calidad de la aplicación. Alguna de ellas son las siguientes:

- Implementar posibles nuevas inconsistencias que evaluar en un entorno adaptativo. A pesar de que se ha implementado un gran número de aspectos que evaluar, siempre cabe la posibilidad de incluir nuevos problemas que puedan ser analizados. Un ejemplo sería el realizar un análisis más profundo de los datos relacionados con la interacción del sistema CoMoLE con otros usuarios. Además, la aplicación ha sido diseñada de tal manera que el añadir nuevas inconsistencias no sería una tarea muy costosa.

- Diseñar una interfaz más potente para usuarios expertos. La simplicidad de la interfaz es un aspecto importante, pero cabe la posibilidad de que cierto tipo de usuarios requiera de una interfaz más potente que le permita realizar la evaluación de sólo ciertos criterios que él considere oportuno. Un ejemplo podría ser el que un profesor decidiese evaluar una única actividad, o el que en vez de analizarse todas las inconsistencias de un tipo se pudiese evaluar cada una de ellas de manera independiente.

- Incluir un analizador de los ficheros de datos. Como en muchas ocasiones los ficheros de datos son modificados de una forma manual, cabe la posibilidad de que a veces no se respete la estructura que el fichero debería tener. Ante esta circunstancia la herramienta desarrollada, avisaría de un error en la construcción de dichos ficheros, pero no identificaría el motivo del mismo. Por ello creo que sería interesante extender la herramienta con el mencionado analizador de ficheros.

Bibliografía

Martín, Estefanía: **Creación de entornos adaptativos móviles: recomendación de actividades y generación dinámica de espacios de trabajo basadas en información sobre usuarios, grupos y contextos.** Tesis doctoral.

Ceballos, Francisco Javier: **Java 2. Curso de programación**.

Google: http://www.google.es/

FileUpload: http://commons.apache.org/fileupload/

JDom: http://www.jdom.org/

JavaHispano: http://www.javahispano.org/

Tomcat: http://tomcat.apache.org/

Sun: http://java.sun.com/

Anexo A: Manual de instalación

Para poder utilizar la herramienta desarrollada en el presente proyecto, es necesario hacer una instalación previa de algunos componentes. Por ello se va exponer de manera breve como se debe realizar la instalación de todos los programas necesarios para la ejecución de la aplicación.

A.1. Máquina virtual de Java

La herramienta ha sido desarrollada en el lenguaje de programación Java, por ello será necesario tener instalada la maquina virtual de Java. Para la realización de este proyecto se ha utilizado la versión de la máquina virtual 1.6, por lo que sería conveniente la instalación de dicha versión. La maquina virtual de Java se puede descargar desde la página http://www.java.com/es/download/ donde la descarga es totalmente gratuita.

A.2. Servidor Web

Como la aplicación desarrollada es una aplicación Web, será necesario tener instalado un servidor Web. En la actualidad existen gran cantidad de servidores Web, sin embargo el utilizado durante el desarrollo del proyecto ha sido el servidor Apache, ya que es un servidor de código abierto y es gratuito. Debido a que en el desarrollo de la aplicación se han utilizado servlets, se requerirá de un servidor Web que lo soporte. El servidor empleado ha sido Tomcat, el cual se puede obtener de manera gratuita a través de http://tomcat.apache.org/

Instalación de Tomcat:

1. Se deberá definir las variables de entorno JAVA_HOME y JRE_HOME, cuya ubicación normalmente se encuentra en C:\Program Files\Java\. Para ello seleccionaremos con el botón secundario en “Mi PC” y pulsaremos en “Propiedades”. Tras ello nos iremos a la pestaña de Opciones avanzadas, en

donde pulsaremos en “Variables de entorno”. Una vez abierta esta ventana procederemos a añadir los valores a las variables mencionadas.

2. Una vez realizado el paso anterior, será necesario ejecutar el archivo *startup.bat* que se encuentra en la carpeta *bin* del lugar donde hayamos instalado Tomcat. De esta manera, el servidor Web quedará lanzado.

Tras realizar estos sencillos pasos, será necesario detener el servicio. Para ello nos iremos al lugar donde tengamos instalado Tomcat y dentro de la carpeta *bin* ejecutaremos el fichero *shutdown.bat*. Tras ello copiaremos el fichero del proyecto “WebEvaluacion.war” en la carpeta de *webapps* dentro de donde tengamos instalado Tomcat. Por último volveremos a lanzar el servidor, para ello será necesario ejecutar de nuevo el archivo *startup.bat* que se encuentra en la carpeta *bin* del lugar donde hayamos instalado Tomcat. .

Para ejecutar el proyecto desarrollado sólo tendremos que ir a un navegador Web y escribir http://localhost:8080/WebEvaluacion. De esta forma lanzaremos la aplicación de manera local.

Anexo B: Contenido del CD

- Ejecutable de la aplicación “WebEvaluacion.war”.
- Varios ficheros comprimidos que contendrán los ficheros de datos de entornos adaptativos. Estos ficheros serán necesarios para poder realizar la evaluación de los entornos.
- Carpeta con los ficheros de datos, por si se quiere generar los ficheros comprimidos.
- Código fuente de toda la aplicación.
- Librerías que utiliza la aplicación.
- Memoria del presente Proyecto Final de carrera.