AdapTeach:
Framework para la construcción de sistemas de enseñanza adaptativos para la Web Semántica.

Tesista

Lic. Vanesa Maiorana
maioranavanesa@yahoo.com.ar

Director

Dr. Ricardo Fabián Orosco
rorosco@uade.edu.ar

Co-Director

Dr. Gustavo Rossi
gustavo@sol.info.unlp.edu.ar

Tesis presentada para obtener el grado de
Magister en Ingeniería de Software.
Facultad de Informática – Universidad Nacional de La Plata

Abril de 2007

Contenidos

Agradecimientos

Más importante que lograr la meta es recorrer el camino. Recorrer el camino significa concentrarse en las pequeñas cosas, poniendo toda la energía y el esfuerzo en cada uno de los pasos que se dan, aunque a veces parezcan insignificantes….porque sólo dando pequeños pasos puede alcanzarse el objetivo.

Tan importante como poner esfuerzo, concentración y entusiasmo en lo que se hace, es necesario vivir y trabajar en un entorno favorable.

No me hubiera sido posible realizar este trabajo si no hubiera contado con el apoyo de las grandes personas que me han acompañado en cada paso de este camino.

En primer lugar quiero agradecer a mi director de tesis, Ricardo Orosco, por haberme guiado desde el comienzo hasta el final, dedicando su valioso tiempo y sobre todo, su valiosísima guía, conocimientos y apoyo incondicional.

Quiero también agradecer a Gustavo Rossi, co-director de este trabajo, quien siempre ha estado dispuesto para orientarme y darme los consejos necesarios para seguir adelante.

Mi maravillosa familia ha sido el apoyo fundamental para que yo pueda desarrollar este trabajo. Mi hijo Nicolás y mi marido Nani me permitieron dedicar tiempo y energía a este proyecto, acompañándome en todo momento. Además de ellos, mi gran familia incluye a mis padres Norberto y Susy, mis hermanas Giselle y Julieta, mis cuñados Leo y Jorge, y mis hermosos sobrinos Franco, Juan Manuel y el/la que está en camino. A esta gran familia, que funciona como un verdadero equipo, debo agradecer la compañía, la ayuda y el apoyo constante. Formar parte de este equipo es uno de los regalos más lindos de la vida que me ayudan a crecer como persona y como profesional.

Por último, no quiero dejar de agradecer a cada uno de mis alumnos, ya que la enseñanza y la experiencia que ellos me dejan día a día, es lo que ha motivado la realización de este proyecto y son ellos quienes me permiten seguir aprendiendo en la emocionante tarea de enseñar.

1. Introducción

1.1. Objetivos y alcances del trabajo

El objetivo de este trabajo de tesis es la construcción de un framework o marco de trabajo para la construcción de sistemas adaptativos de enseñanza.

El objetivo del framework es su utilización para la construcción de sistemas de enseñanza en cualquier dominio, y su implementación en la Web semántica.

1.2. Motivación

En el proceso de enseñanza-aprendizaje que ocurre dentro de un aula, existe una interacción entre docentes y alumnos. En este proceso los participantes interactúan de manera constante. El alumno realiza ejercicios, lee la explicación de conceptos, observa al docente cómo se resuelven los problemas, etc. El docente explica conceptos, demuestra la resolución de ejercicios, responde preguntas de los alumnos y observa el avance de los mismos para explicar conceptos no comprendidos, para definir el siguiente ejemplo o concepto a explicar.

Un sistema de enseñanza computarizado (e-learning) debería simular estas interacciones que se dan naturalmente entre docentes y alumnos. Si el sistema representa al docente, tendrá que simular su comportamiento y tener la capacidad de observar al usuario para tomar decisiones.

Si bien ningún sistema podría reemplazar completamente a un docente en el aula interactuando con sus alumnos, los sistemas de e-learning pueden ayudar y servir de soporte al proceso de enseñanza.

En este trabajo se considera al e-learning como un proceso en el cual el éxito no solo depende del alumno, que utiliza la tecnología para aprender. Sino también se considera que la habilidad del docente para transmitir los conocimientos, para observar al alumno y tomar decisiones, es tan importante como la actitud del alumno frente al aprendizaje. Por ello podríamos decir que el término e-teaching es más adecuado para enmarcar este trabajo.

Un sistema adaptativo permite que el sistema se adapte al usuario. Un sistema adaptativo de e-teaching, estaría orientado a brindar una enseñanza personalizada. La idea de personalización está ampliamente difundida en trabajos realizados por otras personas y también en los productos de e-learning desarrollados en el mercado. Sin embargo, el concepto de adaptabilidad en los sistemas de enseñanza tiene varios aspectos y objetivos, y actualmente se está trabajando en ellos con diferentes enfoques. Uno de los objetivos en los sistemas de enseñanza adaptativos, es la secuenciación de contenidos, es decir, el orden en el que se presentan los contenidos a cada alumno en particular.

Este trabajo se centra en tres aspectos importantes: la representación del conocimiento del dominio, la representación de la experiencia del docente, y la secuenciación de contenidos.

Para proveer los contenidos, se ha adoptado representar el conocimiento a enseñar en una ontología. Para la representación de la experiencia del docente y la determinación de la secuencia de contenidos, se ha utilizado la técnica de razonamiento basado en casos, que proviene del área de la inteligencia artificial.

En este trabajo se presenta un framework, que incluye la representación del conocimiento del dominio a enseñar, la estructura de la experiencia que almacenará el sistema, la estructura básica del perfil del usuario y la forma en que se utiliza el conocimiento, la experiencia y el perfil, para ofrecer la secuencia de contenidos a un alumno.

El framework presentado describe en forma detallada la estructura de la ontología, que podría utilizarse para cualquier tipo de contenido que se desee enseñar. La ontología permite estructurar el conocimiento y además, permite implementar el sistema adaptativo en lo que se cree que será el futuro de la Web: la Web semántica.

La experiencia del docente está representada por el caso, concepto que proviene de la técnica de razonamiento basado en casos. Estos casos permiten definir un algoritmo que, según el perfil del usuario, identifica el siguiente paso en el proceso de enseñanza, que puede ser un nuevo conocimiento, un ejemplo, un ejercicio, una pregunta, etc. En este trabajo se presenta la estructura básica que tienen los casos y provee algunos ejemplos de los mismos. Además se describe la forma en que el algoritmo toma el perfil del usuario y obtiene el mejor o los mejores pasos a seguir.

El conocimiento a enseñar está estructurado utilizando el concepto de learning object, que es una unidad de aprendizaje.

1.3. Contenido del trabajo

El capítulo siguiente (capítulo 2) contiene definiciones de los conceptos básicos utilizados para este trabajo.

En el capítulo 3 se describen los sistemas de enseñanza adaptativos. Esto incluye las formas de organizar el conocimiento y las formas de adaptabilidad que se han identificado para estos sistemas.

El capítulo 4 describe el framework propuesto en este trabajo, denominado Adapteach. Este capítulo incluye una descripción de la adaptabilidad navegacional en el framework AdapTeach, que es la que tiene que ver con la secuenciación de contenidos; la arquitectura del framework con una descripción de cada uno de los componentes; el detalle de la ontología que servirá de base de conocimiento; el perfil del usuario básico para realizar la adaptabilidad navegacional; y el mecanismo para la secuenciación de contenidos basada en el estado actual de un usuario y la experiencia.

El capítulo 5 muestra cómo se puede instanciar AdapTeach para enseñar cualquier contenido, a través de un ejemplo para la enseñanza de orientación a objetos. Esto incluye principalmente la instanciación de la Ontología y la definición de los casos de entrenamiento del sistema.

En el capítulo 6 se describe la prueba de concepto desarrollada para probar que el framework presentado se puede implementar. Esta prueba de concepto es la base para el desarrollo futuro de un sistema que implemente el framework.

El capítulo 7 presenta una selección de trabajos relacionados a éste por tener objetivos similares.

En el capítulo 8 se presentan las conclusiones obtenidas a lo largo de este trabajo y se enumera una serie de trabajos futuros posibles, algunos de los cuales se encuentran en desarrollo[1].

Finalmente se enumeran las referencias bibliográficas utilizadas y los anexos.

[1] Consultar el sitio Web www.adapteach.com.ar para ver los trabajos en desarrollo.

2. Definición de conceptos relacionados

En este capítulo se definen algunos de los conceptos más importantes relacionados al trabajo realizado. En las siguientes secciones se describirán brevemente los siguientes temas:

- ◊ Sistemas adaptativos.
- ◊ Web semántica.
- ◊ Ontologías.
- ◊ Razonamiento basado en casos.
- ◊ Objetos de aprendizaje o Learning Objects.

2.1. Sistemas de enseñanza adaptativos

El ámbito del E-Learning moderno está dominado por los llamados Learning Management Systems (LMS), tales como Blackboard o WebCT. Los LMSs son sistemas integrados que proveen soporte para una variedad de actividades realizadas por docentes y estudiantes en el proceso de e-learning. Los docentes utilizan los LMSs para desarrollar apuntes y ejercicios, para comunicarse con los estudiantes y para monitorear el progreso de los mismos. Los alumnos utilizan estos sistemas para el aprendizaje, la comunicación y la colaboración. Tal como lo hacen muchos sistemas actuales basados en la Web, los LMSs ofrecen un servicio "one size fits all" , que significa que todos los alumnos que tomen un curso basado en un LMS recibirán el mismo material y las mismas herramientas, sin importar su conocimiento, objetivos e intereses. No existe el soporte personalizado en este tipo de sistemas. Los sistemas educacionales adaptativos, pretenden luchar contra este enfoque "one size fits all" en el e-learning. [Brusilovsky2]

En un sistema de enseñanza, la palabra adaptativo puede tomar diferentes significados [Nodenot at al.]:

Adaptabilidad de navegación (adaptive guidance): significa proveer al alumno de un camino óptimo de navegación a través de los materiales, de acuerdo a sus preferencias, objetivos, historia de navegación y/o conocimiento testeado.

Adaptabilidad de presentación: es la adaptación del contenido en sí mismo de acuerdo a preferencias, objetivos, historia de navegación y/o conocimiento testeado.

Adaptabilidad de colaboración: permite la generación de grupos de colaboración que tengan objetivos de aprendizaje similares o la búsqueda del usuario más competente para responder a una pregunta sobre un tema específico.

2.2. Web Semántica

La semántica es el estudio del significado. La palabra semántica proviene del griego *semantikos*, o "significado con sentido" (significant meaning). Las tecnologías de la Web semántica pretenden separar el significado de los datos, contenidos o código de la aplicación, utilizando tecnologías basadas en estándares abiertos.

Si una computadora entiende la *semántica* de un documento, no solo interpreta la serie de caracteres que componen el documento, sino que comprende el *significado* del mismo.

La Web semántica provee un framework común que permite que los datos sean compartidos y reusados a través de la aplicación, la empresa y los límites de la comunidad. Se puede pensar la Web semántica como una forma eficiente de representar los datos en la Web, o como una base de datos vinculada globalmente, de manera que puede ser comprendida por máquinas. Las tecnologías semánticas representan el significado utilizando ontologías y proveen razonamiento a través de las relaciones, reglas, lógica, y condiciones representadas en las ontologías [Balani]

2.3. Ontologías: conocimiento en la Web Semántica

Según Knublauch [Knublauch et al.], las ontologías cumplen un rol fundamental en la Web semántica. Estas proveen modelos formales del conocimiento del dominio que pueden ser procesados por agentes inteligentes. Por lo tanto, las herramientas de desarrollo para la Web semántica deben proveer servicios para acceder, visualizar, editar y utilizar ontologías. Además, dado que las ontologías son muy difíciles de construir, se requieren herramientas para asistir de forma inteligente en la construcción y evolución de las mismas.

Las ontologías proveen agentes con conocimiento sobre los objetos de un dominio y sus relaciones. Estas pueden ser instanciadas para crear individuos que describan los recursos de la Web semántica o entidades del mundo real. Por ejemplo, los individuos de una ontología para agentes de viajes, podría representar destinos específicos de vacaciones y actividades . En este escenario, un repositorio proveería los datos sobre estos individuos, y agentes podrían usar este conocimiento ontológico para buscar ofertas que coincidan con los intereses de los usuarios.

Protégé : Una herramienta de edición de ontologías

Protégé es una herramienta open-source desarrollada en la Universidad de Stanford (Stanford Medical Informatics). Si bien el desarrollo ha estado dirigido principalmente por aplicaciones biomédicas, el sistema es independiente del dominio y ha sido usado con éxito en otras áreas de aplicación.

La arquitectura de Protégé se separa en una parte visual y un modelo. El modelo es la representación interna de las ontologías y las bases de conocimiento. Los componentes de la parte visual proveen una interfaz para mostrar y manipular el modelo. El modelo está basado en un meta-modelo flexible, que se compara con los sistemas orientados a objetos y orientados a marcos (frames). El modelo permite representar ontologías que consisten en clases, propiedades (slots), características de propiedades (facets y constraints), e instancias. El meta-modelo de Protégé es una ontología en sí misma, con clases que representan las clases, propiedades, etc.

Protégé provee un API de Java para consultar y manipular modelos. Utilizando las vistas de Protégé, se pueden asignar propiedades a las clases, y luego proveer restricciones sobre esas propiedades. Protégé puede usarse para cargar, editar y guardar ontologías en varios

formatos, tales como Clips, RDF, XML, UML y bases de datos relacionales. Recientemente se ha agregado soporte para OWL.

Lenguaje OWL (Ontology Web Language)

Las ontologías se utilizan para capturar el conocimiento sobre un dominio de interés, ya que describen los conceptos del dominio y las relaciones entre ellos. Existen diferentes lenguajes que proveen diferentes facilidades. El desarrollo más reciente en los lenguajes para ontologías es OWL, desarrollado por el World Wide Web Consortium (W3C). OWL permite describir conceptos, pero además provee otras facilidades. Está basado en un modelo lógico que hace posible la definición y descripción de conceptos. Los conceptos complejos pueden ser construidos a través de las definiciones de conceptos más simples. Además, el modelo lógico permite el uso de un razonador para chequear si las sentencias y las definiciones en la ontología son consistentes y también para reconocer qué conceptos entran en qué definiciones. El razonador puede, entonces, ayudar a mantener la jerarquía de conceptos correcta. Esto es particularmente útil cuando se trabaja con casos donde las clases tienen más de un padre (herencia múltiple). [OWLTutorial]

El *plugin OWL* es una extensión de Protégé, que como se ha dicho anteriormente, es una plataforma abierta para el modelado de ontologías y adquisición de conocimiento. El plugin puede usarse para editar ontologías en lenguaje OWL. [Knublauch et al.]

2.4. Razonamiento basado en casos (Case-Based Reasoning)

Razonamiento basado en casos (CBR) es una técnica del campo de la Inteligencia artificial, que permite resolver problemas adaptando soluciones anteriores a problemas nuevos similares.

El proceso de CBR puede representarse como un ciclo, según Aamodt & Plaza, que tiene 4 pasos, llamados los 4 REs [Watson&Marir]:

- ◊ REcuperar(Retrieve) el/los caso/s más similar/es
- ◊ REusar (Reuse) el caso para intentar resolver el problema
- ◊ REvisar (Revise) la solución propuesta si es necesario
- ◊ REtener (Retain) la nueva solución como parte de un nuevo caso

Un problema nuevo se compara con los casos existentes en la base de casos y se recuperan uno o más casos similares. Se obtiene una solución de los casos recuperados, se aplica y se testea el resultado. A menos que el caso sea muy similar, la solución es revisada o adaptada y se produce un nuevo caso para almacenar en la base de casos.

El ciclo ocurre con intervención humana en la mayoría de los casos. Por ejemplo, muchas aplicaciones de CBR actúan principalmente como sistemas de recuperación y reuso. La adaptación o revisión es realizada por un administrador de la base de casos. Esto no se considera una debilidad de CBR sino que impulsa la colaboración de humanos en la toma de decisiones de los agentes.

Un caso es una pieza contextualizada de conocimiento, que representa la experiencia. Contiene las lecciones aprendidas. Generalmente, un caso incluye:

- ◊ El problema: describe el estado del mundo cuando ocurrió el caso
- ◊ La solución: establece la solución para el problema y/o
- ◊ La salida (outcome): describe el estado del mundo después de que se aplicó el caso.

Los casos pueden representarse en una variedad de formas, utilizando diferentes formalismos de representación de Inteligencia artificial, tales como marcos (frames), objetos, predicados, redes semánticas, reglas, etc.

El método del vecino más próximo

El enfoque que se describe en esta sección, involucra el cálculo de la similitud entre los casos almacenados en la base de casos y el nuevo caso o problema. Este cálculo se realiza en base a la suma pesada de las características que definen un caso. El problema en este caso es la determinación de los pesos de cada característica. La limitación de este enfoque tiene que ver con los tiempos de recuperación, ya que el tiempo se incrementa linealmente con el número de casos. Por ello este método es más efectivo cuando la base es pequeña. Muchas aplicaciones de CBR utilizan este método para la recuperación. [Watson&Marir]

El algoritmo para el cálculo del caso más similar o más cercano, es el siguiente:

$$\frac{\sum_{i=1}^{n} w_i \times sim\left(f_i^I, f_i^R\right)}{\sum_{i=1}^{n} w_i}$$

donde:

- ◊ W es la importancia de una característica o slot
- ◊ Sim es la función de similitud
- ◊ Fl y fR son los valores para la característica i del caso nuevo y el recuperado, respectivamente

En este trabajo se utiliza este algoritmo, por lo cual su utilización en el caso concreto se muestra en el desarrollo del trabajo.

2.5. Objetos de aprendizaje / Learning Objects

Los objetos de aprendizaje o learning objects son elementos de un nuevo tipo de enseñanza basada en computadoras, que tiene sus raíces en el paradigma de orientación a objetos. La orientación a objetos se basa en la creación de componentes que pueden ser reusados en

diferentes contextos. Esta es la idea fundamental del enfoque de learning objects: se pueden construir pequeños componentes (en relación al tamaño de un curso completo) de enseñanza que puede ser reusados varias veces en diferentes contextos.

Teniendo en cuenta que los learning objects son unidades pequeñas y reusables de contenido educativo, Reigelith y Nelson (1997) sugieren que los docentes o instructores descompongan los materiales educativos en componentes o partes y que luego utilicen esos componentes para adecuarse a diferentes objetivos educativos.

Los learning objects son definidos por *"The Learning Technology Standards Committee"* como una entidad, digital o no digital, que puede ser usada, reusada o referenciada en el aprendizaje por computadora, tales como sistemas de enseñanza o entrenamiento, entornos de aprendizaje interactivos, sistemas educativos inteligentes, sistemas de enseñanza a distancia, y entornos colaborativos de enseñanza. Según este comité, los learning objects abarcan contenido multimedia, contenido educativo, objetivos de aprendizaje, software y herramientas de enseñanza, personas, organizaciones y eventos referenciados durante el aprendizaje.

Según Wiley [Wiley], la definición anterior es demasiado general, porque abarca desde herramientas de software hasta personas. Esta no es la única definición, y ocurre que la proliferación de definiciones del término learning object genera confusiones. Pero además de las varias definiciones diferentes del término, existen muchas otros términos que se utilizan en este ámbito y que pueden generar más confusión. Ejemplos de estos términos son: "knowledge objects", "Components of Instruction", "instructional components", "pedagogical documents", "educational software components" , "online learning materials", entre otros. El autor propone una definición más específica que es "cualquier recurso digital que puede ser usado para e-learning". Esta definición, incluye cualquier cosa que pueda ser entregada en Internet a demanda, sea grande o pequeño. Ejemplos de recursos digitales pequeños serían: imágenes o fotos, videos o archivos de audio, pequeñas porciones de texto, animaciones, pequeñas aplicaciones Web, entre otros. Ejemplos de recursos más grandes incluyen páginas Web que combinan texto, imágenes u otros medios o aplicaciones.

3. Sistemas de enseñanza adaptativos

3.1. El conocimiento en los sistemas adaptativos para la enseñanza.

En esta sección se describe la forma de estructurar el conocimiento de un dominio a enseñar en un sistema adaptativo para la enseñanza (e-teaching). Esto incluye la descripción de diferentes tipos de conocimiento, la división del mismo en objetos de aprendizaje o learning objects, y los diferentes caminos que un usuario podría tomar navegando entre estos objetos.

3.1.1. Diferentes tipos de conocimiento

Cualquier dominio incluye diferentes tipos de conocimiento. En este trabajo se ha definido la siguiente clasificación:

- ◊ Conocimiento declarativo
- ◊ Conocimiento procedural

A continuación se describen las diferencias entre ambos y se dan algunos ejemplos de cada tipo.

Conocimiento declarativo

En cualquier dominio de conocimiento aparece la necesidad de describir conocimiento declarativo. El conocimiento declarativo podría ser alguna de las siguientes cosas:

- ◊ Definiciones
- ◊ Ejemplos
- ◊ Reglas de aplicación
- ◊ Criterios
- ◊ Características
- ◊ Experiencias

Conocimiento procedural

Además del conocimiento declarativo, en cualquier dominio existen los procesos o fenómenos. Si queremos transmitir conocimiento sobre éstos, debemos transmitir cómo funcionan estos procesos o fenómenos, o cómo se desarrollan determinadas tareas. Llamaremos a este tipo de conocimiento, conocimiento procedural, conocimiento que podría describir alguna de las siguientes cosas:

- ◊ Secuencia de pasos para hacer o construir algo
- ◊ Explicación de un fenómeno natural
- ◊ Explicación de un proceso
- ◊ Demostración de un fenómeno o proceso.

3.1.2. Diferentes tipos de Learning objects

Un sistema de enseñanza debería considerar ambos tipos de conocimiento para transmitir un dominio de la mejor manera a quien desea aprenderlo. Pero además, deberíamos tener algún mecanismo de evaluación de estos contenidos aprendidos. La evaluación será también un contenido en el sistema de enseñanza.

En base a los diferentes tipos de conocimiento descriptos anteriormente y a la necesidad de tener una manera de evaluar al alumno en el sistema, se han identificado tres tipos de learning objects:

- ◊ Learning objects declarativos (Declarative Learning Object - DLO) : contienen conocimiento declarativo.
- ◊ Learning Objects procedurales (Procedural Learning Object -PLO): contienen conocimiento procedural.
- ◊ Learning objects de testeo o evaluación (Test Learning Object - TLO): contienen preguntas o ejercicios que el alumno debe responder para ser evaluado por el sistema y así actualizar sus puntajes.

En la Figura 1 se muestra un mapa de los learning objects correspondiente a una lección particular. En este ejemplo la lección se denomina "*Contracts*" y corresponde al dominio de la enseñanza de una metodología de orientación a objetos [Larman]. Más adelante se utilizará este dominio en los ejemplos de las diferentes partes del framework presentado.

Los diferentes tipos de learning objects aparecen en diferentes colores en el mapa. En rojo los que contienen conocimiento declarativo, en azul los que contienen conocimiento procedural y en verde los que contienen contenido de evaluación.

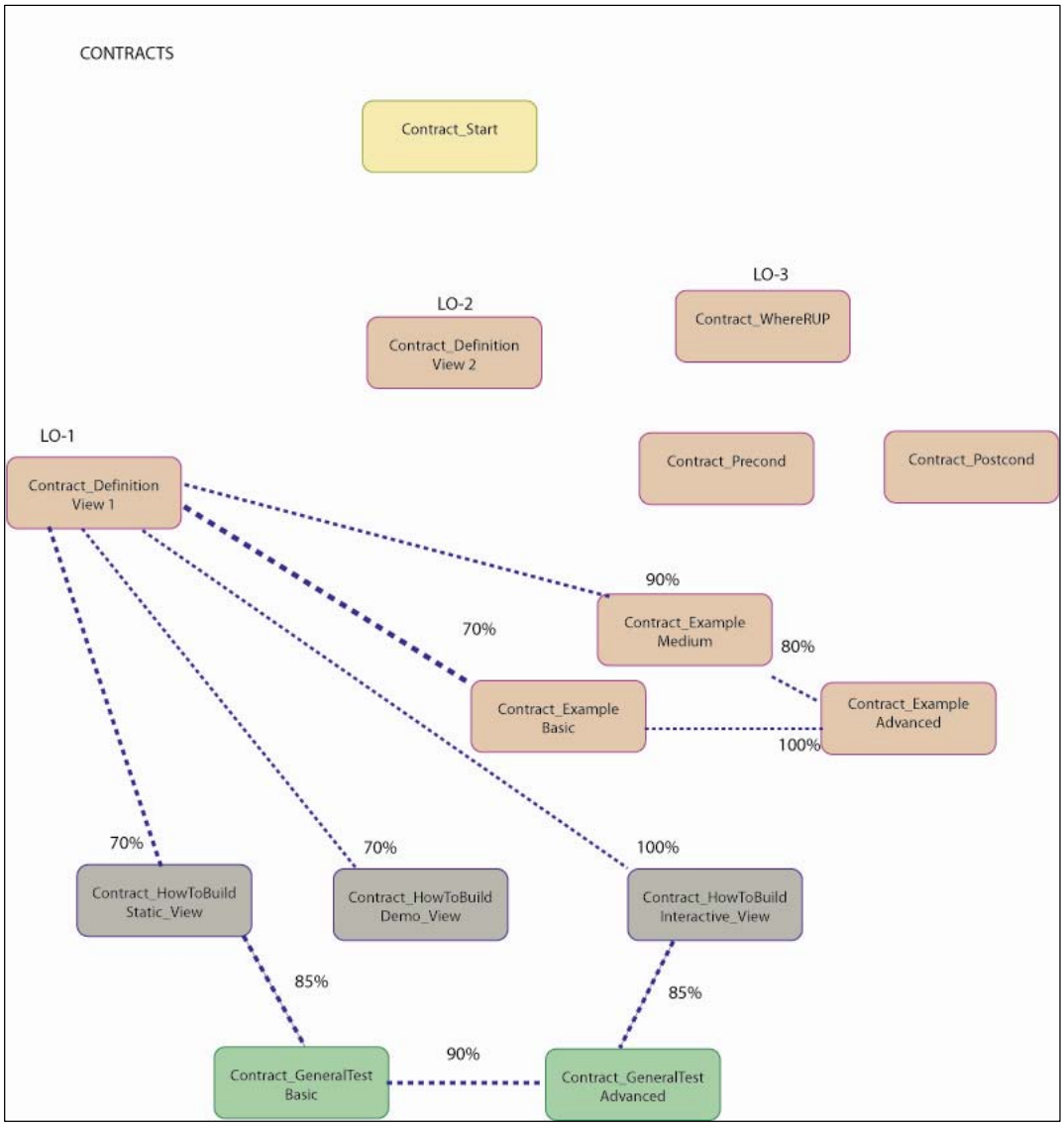


Figura 1: Mapa de Learning Objects de una lección llamada "*Contracts"*.

Relaciones de pre-requisitos entre learning objects

Como se ve en el gráfico de la sección anterior, hay líneas punteadas entre algunos learning objects. Estas líneas representan relaciones de pre-requisitos y tienen un porcentaje asociado. Este porcentaje indica el puntaje que se necesita haber obtenido en un learning object para pasar a otro.

Por ejemplo, tomemos en cuenta el learning object "*Contract Definition–View 1*", que tiene dos conexiones de pre-requisitos:

- "*Contract Example Basic*", con un 70%.
- "*Contract Example Medium*" con un 90%.

Esto significa que el usuario necesita obtener un 70% para poder pasar desde el LO "*Contract Definition – View 1*" hacia el LO "*Contract Example Basic*", pero necesita un 90% para pasar al LO "*Contract Example Medium*".

Con este ejemplo particular podemos ver que no siempre el usuario podrá acceder a los mismos learning objects, sino que depende de sus puntajes en determinados momentos.

Algunos learning objects no tienen relaciones de pre-requisitos, lo cual significa que el usuario no necesita haber obtenido un puntaje determinado en otros learning objects.

Cabe aclarar que en el mapa ejemplo dado en la sección anterior, solo se están mostrando las relaciones de pre-requisitos existentes entre learning objects de la misma lección. Pero si modelamos todo el dominio a enseñar, habrá relaciones de pre-requisitos con learning objects de otras lecciones.

Diferentes formas que pueden tomar los learning objects procedurales.

El conocimiento que estaría dentro de la categoría de los PLO, puede tomar las siguientes formas:

- ◊ Explicación textual sobre los pasos de un proceso.
- ◊ Animación o video que muestra cómo se hace o como funciona algo.
- ◊ Animación o video que muestra algún fenómeno natural que debe ser aprendido.

Los PLOs tendrían tres formas de ser aprendidos: un modo estático, un modo de demostración y un modo interactivo.

- ◊ Modo estático: Este modo muestra en forma textual o gráfica la secuencia de pasos para hacer algo. Se dice que es estático porque el contenido se presenta como un conjunto de ítems, en los cuales puede encontrarse texto, gráficos o fotografías. El modo estático sería la explicación de los pasos del proceso, como si una persona estuviera leyéndola de un libro.
- ◊ Modo demo: El modo demo muestra un determinado proceso en forma de película o animación. En este modo no hay intervención del usuario (quien solo interviene para controlar la velocidad de la explicación).
- ◊ Modo interactivo: muestra un proceso, como en el modo demo, pero en este caso el sistema pide intervención al usuario solicitando respuestas a preguntas. También sería una especie de película o animación, pero que se detiene cada vez que tiene que pedir respuesta del usuario.

3.2. Tipos de adaptabilidad en un sistema de enseñanza.

En esta sección se describen las diferentes maneras de adaptabilidad que se han identificado en un sistema de enseñanza. Algunas de estas formar provienen de la adaptabilidad de los sistemas adaptativos en general, y otras surgen del desarrollo de este trabajo.

Los tipos de adaptabilidad identificados son:

- ◊ Adaptabilidad de contenido declarativo.
- ◊ Adaptabilidad de contenido procedural.
- ◊ Adaptabilidad de contenido de testeo.
- ◊ Adaptabilidad navegacional.
- ◊ Adaptabilidad de preferencias de usuario.
- ◊ Adaptabilidad de presentación.

3.2.1. Adaptabilidad de contenido declarativo

Un DLO se explica de diferentes maneras según el aprendizaje del alumno y sus preferencias. Que un learning object se explique de diferentes maneras puede significar que:

- ◊ Se explica con diferentes palabras el mismo concepto.
- ◊ Se explica con diferentes ejemplos.
- ◊ Se explica en diferentes niveles de detalle.
- ◊ Se explica en diferentes niveles de complejidad.

3.2.2. Adaptabilidad de contenido procedural

Una persona que aprende cómo hacer algo, puede utilizar el conocimiento adquirido de diferentes maneras. Por ejemplo:

- ◊ Tomar el texto sobre cómo hacer algo, leerlo todo y luego aplicarlo en la experiencia real para poder comprobarlo.
- ◊ Tomar el mismo texto y leer un paso, luego aplicarlo, leer el paso siguiente y aplicarlo, etc.
- ◊ Ver el video o animación completo de cómo hacer algo y luego al finalizar aplicarlo para comprobarlo.
- ◊ Ver el video paso a paso, pudiendo pararlo y arrancarlo a su gusto.

El sistema que enseña este contenido, podría adaptarse al usuario para que éste aprenda.

La adaptabilidad en los *modos estático* y *demo* tendría que ver con el orden en el que se muestran los pasos que explican el proceso o fenómeno, que podría variar según el estado actual del alumno. Por ejemplo, un determinado proceso podría considerar algunos pasos cuando el usuario tiene conocimientos avanzados, y evitarlos (abstraer) cuando el usuario es principiante.

En el modo Interactivo, la adaptabilidad se podría dar según la interacción del usuario. El proceso se enseña mediante una serie de pasos con preguntas o ejercicios que el usuario tiene que ir respondiendo. Según la respuesta o resultado del ejercicio, el sistema ofrecerá un siguiente paso determinado. Se considera que esto es adaptabilidad de contenido ya que el conocimiento sobre el proceso o fenómeno completo se considera un único Learning Object. Si fueran varios learning objects, se estaría considerando la adaptabilidad navegacional entre ellos.

3.2.3. Adaptabilidad de contenido de testeo, de tipo declarativo.

Un ejercicio o test puede variar según el aprendizaje hasta el momento del alumno, y según la manera en la que el alumno responde a las preguntas o ejercicios. Las diferentes alternativas son:

- ◊ se puede hacer una pregunta u otra en función de una respuesta.
- ◊ se puede proponer un ejercicio similar en complejidad si el alumno resolvió un ejercicio de manera incorrecta, y proponer uno de mayor complejidad si respondió correctamente.

Se puede proponer a un alumno una nueva pregunta aunque su respuesta haya sido incorrecta, para que en determinado punto se de cuenta solo de su error. Esto implicaría una "estrategia" global, en lugar de evaluar cada paso del alumno e indicarle si respondió correcta o incorrectamente.

3.2.4. Adaptabilidad de contenido de testeo, de tipo procedural.

En un sistema de enseñanza, el objetivo principal es que el usuario aprenda un conocimiento. Pero para lograrlo, es necesario que el sistema verifique que el usuario está aprendiendo, es decir, validar este aprendizaje. Por tal motivo es que el sistema podría evaluar el conocimiento procedural, tanto al final del aprendizaje como durante el aprendizaje, mientras el usuario está aprendiendo el paso a paso. Por eso es que se considera que además de las formas mencionadas antes, un sistema de aprendizaje debería considerar que:

- ◊ El usuario observa el video o animación en su totalidad y al finalizar responde preguntas o realiza un ejercicio.
- ◊ El usuario observa el video o animación paso a paso, y en cada paso el sistema evalúa su aprendizaje, haciendo alguna pregunta o evaluando algún ejercicio para determinar el paso siguiente.

En este aspecto el sistema de enseñanza podría tomar diferentes "actitudes":

- ◊ Evaluar al usuario e informar la respuesta correcta en cada paso, luego continuar con el siguiente paso.
- ◊ Evaluar la respuesta del usuario, mostrar un siguiente paso que depende de esa respuesta aunque no sea la correcta. Esto llevaría al usuario a un punto donde sea capaz de darse cuenta de su error y volver hacia atrás para corregirlo.

3.2.5. Adaptabilidad navegacional.

La adaptabilidad navegacional es la que tiene lugar cuando el usuario quiere navegar de un learning object a otro.

Adaptabilidad navegacional es la que ocurre cuando el sistema a se adapta al usuario cuando éste navega por los learning objects. En un sistema de enseñanza, el sistema debería ofrecer el mejor o los mejores learning objects por donde seguir en función del estado actual del usuario, según su experiencia.

El sistema podría decidir sugerir visitar:

- ◊ un conocimiento no visitado.
- ◊ un conocimiento que el usuario no ha aprendido correctamente.
- ◊ un conocimiento ya visitado, pero explicado de otra manera, con otro ejemplo o, si se trata de un proceso, en algún otro modo no visitado.

La sugerencia dependerá del estado actual del alumno y de la experiencia del docente.

No se diferencia una adaptabilidad navegacional para cada tipo de learning object. En el caso de los PLO, podría pensarse que existe una adaptabilidad navegacional dentro del mismo learning object cuando el usuario se mueve de un paso a otro en el modo interactivo, pero esto se ha considerado como adaptabilidad de contenido porque estaría navegando dentro del mismo learning object.

3.2.6. Adaptabilidad de preferencias de usuario

El usuario puede definir diferentes preferencias que el sistema podría tomar en cuenta para facilitar su aprendizaje. Algunas de ellas son las que se mencionan a continuación:

- ◊ Color: Colores de la interfaz del sistema
- ◊ Idioma: idioma de presentación de los contenidos. Podrían definirse los contenidos en diferentes idiomas, de manera que el usuario pueda decidir en cual ver los contenidos.
- ◊ Modo preferido para los learning objects procedurales: un usuario puede preferir ver un PLO en modo demo la primera vez que lo visita, puede preferir el interactivo antes del demo, porque prefiere el paso a paso y no la visión global. O puede preferir ver el texto explicativo antes de las imágenes. Por lo tanto, una preferencia de usuario podría ser la elección del orden en el cual el sistema ofrecerá los modos de los PLO.
- ◊ Nivel inicial de complejidad de los TLOs: un sistema de aprendizaje debería tener ejercicios en diferentes niveles de complejidad para ofrecer al alumno. Una preferencia de usuario podría ser el nivel de complejidad con el que prefiere ver los ejercicios. Esto puede ser limitado por el sistema, si considera que no es coherente la preferencia del usuario con su conocimiento actual. Por ejemplo, si el usuario prefiere empezar por los ejercicios más complejos pero su nivel de conocimiento es bajo, entonces el sistema no le permitirá esta preferencia y le ofrecerá solo los niveles básico e intermedio para seleccionar como preferencia. En este aspecto el sistema debería tener cierta inteligencia a la hora de permitir seleccionar el nivel de complejidad. Si un usuario es nuevo, se podría preguntar al mismo qué nivel tiene de conocimientos en la materia, de manera que si no tiene ningún conocimiento previo, no pueda elegir el nivel de ejercicios más complejo. Si el usuario no es nuevo para el sistema, podría cambiar esta preferencia. En este caso el sistema podría ver qué nivel de conocimientos tiene, para ofrecer los tres tipos de complejidad como preferencia

Podrían existir otros tipos de adaptabilidad que tienen que ver con preferencias del usuario, las mencionadas son solo algunas.

3.2.7. Adaptabilidad de presentación

El usuario podría elegir diferentes templates para su aprendizaje, en los cuales se muestran en diferentes posiciones los links de navegación y los contenidos a enseñar.

4. AdapTeach: Framework para la construcción de sistemas adaptativos de enseñanza.

Como se ha mencionado en la introducción, este trabajo presenta un framework para la construcción de sistemas de enseñanza que adaptan la secuencia de los contenidos que se ofrecen a un usuario alumno, según el estado actual del mismo.

El framework, que ha sido llamado AdapTeach, incluye las siguientes descripciones:

- ◊ Definición de la arquitectura del framework con todos sus componentes.
- ◊ Construcción de la estructura básica de la ontología que contiene el conocimiento del dominio a enseñar.
- ◊ Definición del perfil del usuario básico, que contiene la información de su estado actual, o sea, el conocimiento adquirido hasta el momento.
- ◊ Definición del entrenamiento del sistema, esto es, la formulación de los casos que representan la experiencia docente que entrena el sistema.
- ◊ Definición del mecanismo o algoritmo que toma el perfil del usuario y la experiencia docente representada en el sistema, y determina el o los learning objects a sugerir como siguiente paso en la navegación.
- ◊ Descripción de la forma de instanciar la ontología.
- ◊ Descripción de la forma de generar los casos de entrenamiento del sistema
- ◊ Sugerencia sobre tecnologías a utilizar en la implementación del Framework para la Web semántica.
- ◊ Sugerencia de trabajos vinculados a éste que, en conjunto, podrán hacer realidad la implementación de sistemas adaptativos para enseñar en la Web semántica.

A continuación se describirá la diferencia entre un sistema de e-learning tradicional y un sistema construido utilizando el framework, en lo que se refiere a la secuenciación de los contenidos, que es el objetivo de este trabajo. Luego, se mostrará la arquitectura del framework, describiendo cada uno de los componentes.

4.1. Adaptabilidad navegacional en AdapTeach

El framework AdapTeach se centra en la adaptabilidad navegacional, es decir, la forma de adaptación del sistema que tiene que ver con la secuencia de contenidos que el usuario visitará.

Como se ha dicho al comienzo de este trabajo, un sistema de enseñanza tradicional, mostraría los contenidos en una secuencia fija, una lección detrás de otra, según un orden especificado al momento de diseñar el curso.

Un sistema adaptativo no mostraría una secuencia fija, sino que se adaptaría al usuario según su nivel de conocimientos y preferencias.

Como ejemplo, se muestra en la figura 2 un conjunto de learning objects que corresponden a una lección en un sistema de enseñanza. Los learning objects representan las diferentes unidades de conocimiento y pueden ser de los tres tipos indicados en el capítulo 3: DLO, PLO, TLO.

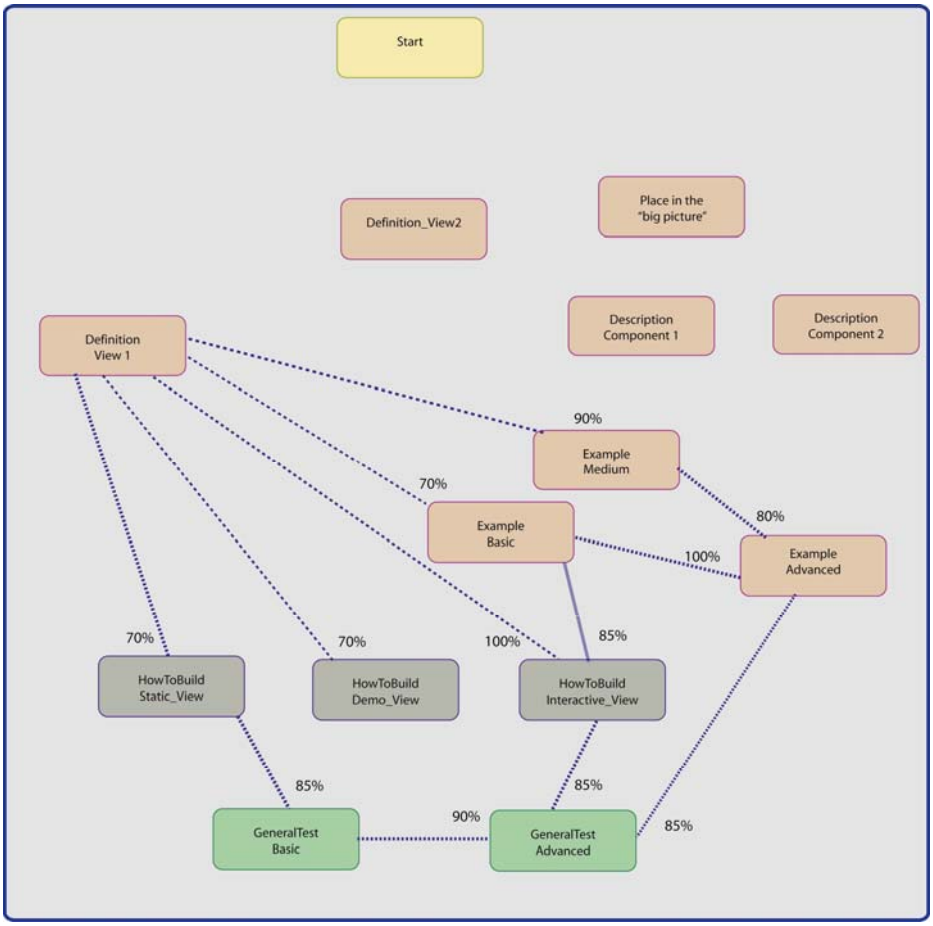


Figura 2: Estructura de learning objects para una lección con relaciones de pre-requisitos.

En esta figura, los learning objects tienen diferentes colores según el tipo: rojo para learning objects con conocimiento declarativo, azul para learning objects de conocimiento procedural y verde para los de testeo. Las líneas punteadas no indican secuencia, sino relaciones de pre-requisitos entre learning objects. Más adelante se explica con mayor detalle este tipo de relaciones.

Navegación en un sistema de e-learning tradicional.

Teniendo la estructura de learning objects de una lección, consideremos cómo sería la navegación en un sistema de e-learning tradicional. Como se ha dicho anteriormente, es una navegación secuencial, igual para todos los usuarios. La figura 3 muestra cómo sería la navegación a modo de ejemplo.

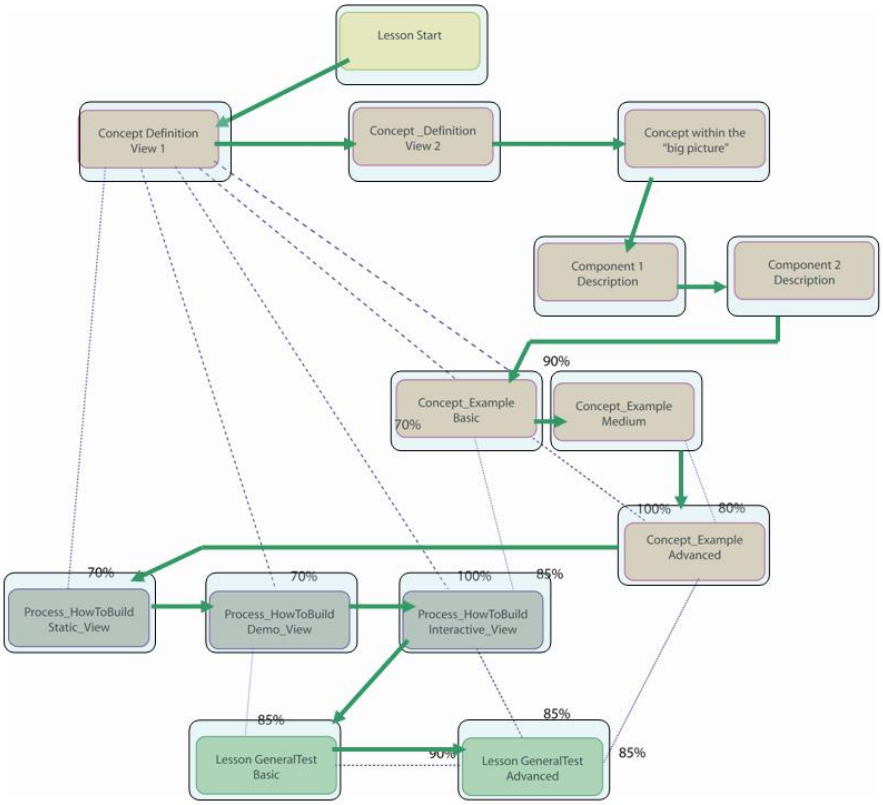


Figura 3: Navegación en un sistema de e-learning tradicional, para cualquier usuario.

Las flechas verdes indican la forma en que el usuario recorrerá los learning objects. Esta secuencia está definida por el diseñador del curso y es igual para todos los usuarios que usen el sistema.

Navegación en un sistema adaptativo

En un sistema adaptativo, la navegación no es secuencial, sino adaptada a cada usuario. Tomemos como ejemplo dos usuarios diferentes y sus navegaciones:

Navegación del usuario1:

El usuario ingresa en la lección y el sistema le sugiere el learning object de definición del concepto. Como comprendió la definición, le sugiere continuar con un ejemplo básico. Como el ejemplo básico fue comprendido en gran medida pero no en su totalidad, el sistema le sugiere ver otro ejemplo más avanzado. Luego le sugiere ver cómo es el proceso de construcción de lo que se está explicando en forma estática, y luego, la demostración dinámica del mismo proceso. Por último le sugiere hacer el test básico de la lección y luego, como respondió el test básico correctamente, el test avanzado.

Navegación del usuario 2:

El usuario 2 ingresa en la lección y el sistema le sugiere la definición del concepto que está siendo explicado en la lección. Como el usuario no comprendió demasiado la definición, se le muestra otra definición del mismo concepto. Luego se le muestra la vista general de lo que se está explicando (the big picture), para que el usuario ubique el concepto dentro de un marco y tenga una comprensión mayor. Luego le sugiere visitar el ejemplo de aplicación básico. Como el usuario comprendió algo del ejemplo pero quedaron puntos importantes sin resolver, pasar a un ejemplo avanzado no es la mejor opción. Por eso el sistema le sugiere un ejemplo intermedio en nivel de complejidad. Luego del intermedio, le sugiere el ejemplo avanzado. Y así continúa la lección hasta que el usuario llega a los learning objects de testeo.

A continuación se muestran las figuras 4 y 5, que permiten ver gráficamente estas diferentes navegaciones. Las flechas verdes indican la secuencia en la navegación. Suponiendo que el usuario aceptó todas las sugerencias del sistema, las diferentes secuencias indican que el sistema sugirió diferentes caminos a diferentes usuarios.

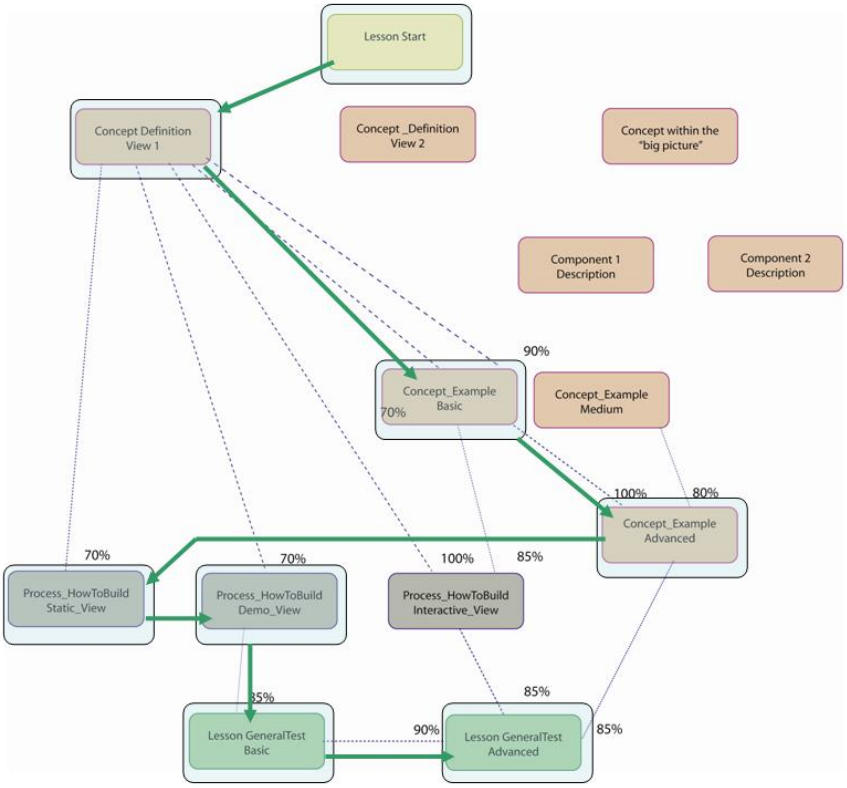


Figura 4: Navegación del usuario 1

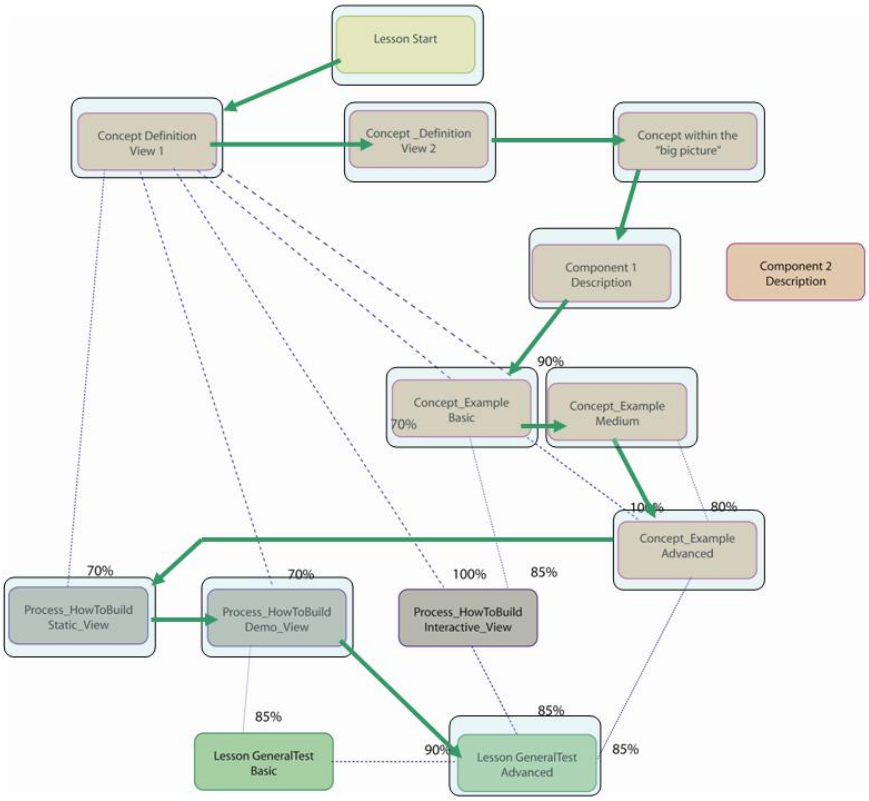


Figura 5: Navegación del usuario 2

Como podemos observar, según cual es el nivel de conocimientos que el usuario va adquiriendo a través de la navegación del sistema, el sistema puede sugerir un learning object u otro.

En las siguientes secciones veremos la estructura de AdapTeach, y analizaremos cómo realiza esta adaptación de navegación en usuarios diferentes.

4.2. Arquitectura de AdapTeach

Como se ha dicho anteriormente, AdapTeach es un framework. Se ha utilizado el concepto de framework porque se presenta el marco de trabajo general, la arquitectura y sus componentes y porque permite ser instanciado para la enseñanza de dominios totalmente diferentes.

En este capítulo se describe la arquitectura, y se detalla cada componente de la misma. En capítulos posteriores se detallará cómo se utiliza el framework para la construcción de un curso en particular a través de un ejemplo. La figura 6 muestra la arquitectura general de AdapTeach, que cuenta con tres partes principales: el conocimiento o memoria, el cerebro y la interfaz. Cada una de estas tres partes tiene componentes encargados de diferentes funciones.

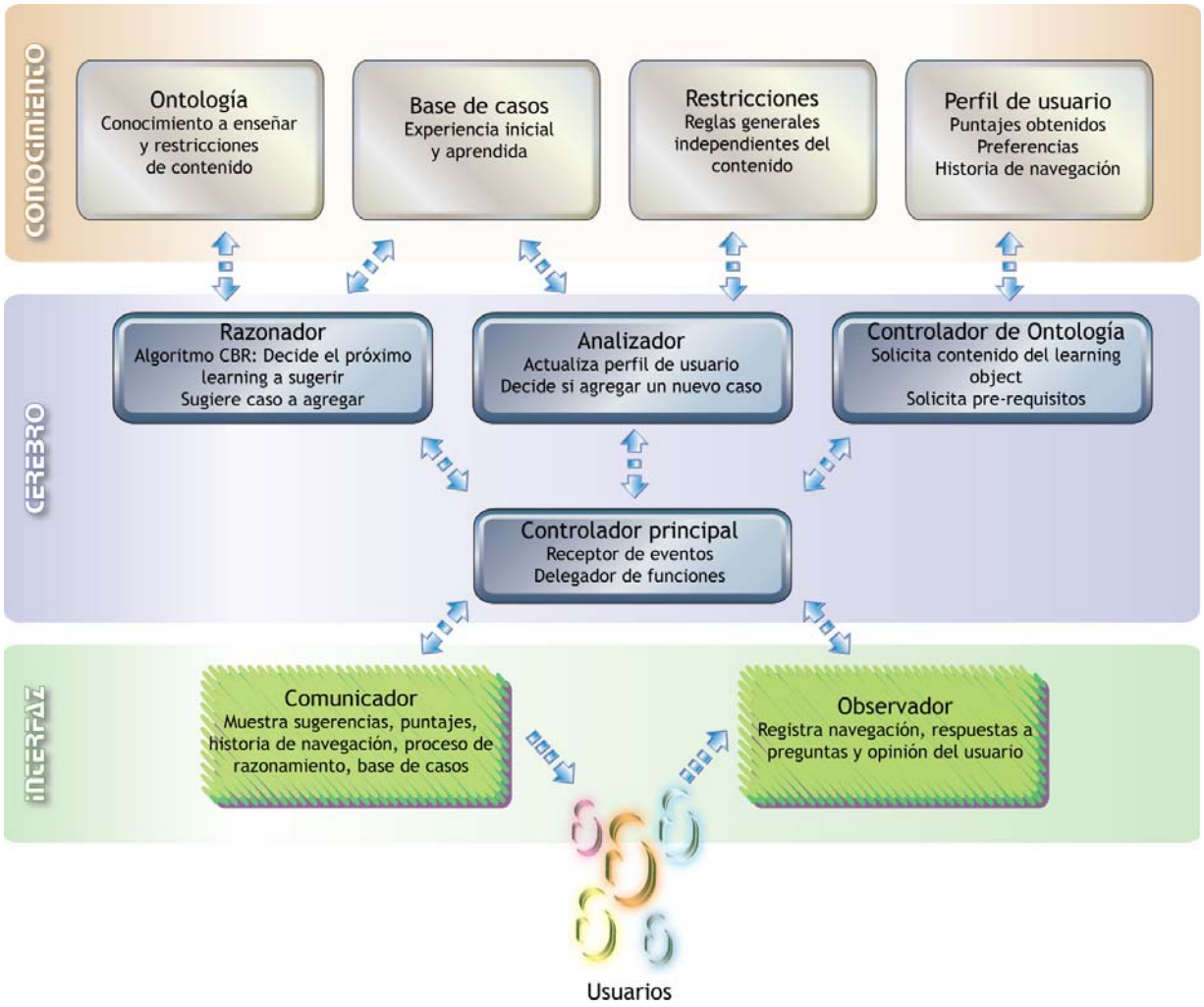


Figura 6: Arquitectura de AdapTeach

4.2.1. El conocimiento de AdapTeach

El conocimiento de AdapTeach está dividido en varios componentes que representan lo que sistema conoce y lo que ha aprendido en el tiempo. Los componentes que contienen el conocimiento se describen a continuación.

Ontología: OntoTeach

La ontología es la base de conocimiento que contiene el conocimiento a enseñar: el contenido de las lecciones, los objetos de aprendizaje, los ejercicios, preguntas y sus respuestas válidas, y los pre-requisitos de cada learning object.

Sobre los learning objects, la ontología contiene tanto su contenido como su importancia. La importancia representa qué tan importante es ese objeto de aprendizaje para la adquisición del conocimiento total del dominio que se desea transmitir. Esta importancia se tiene en cuenta en el cálculo de las sugerencias del sistema. Más adelante se mencionará que relación existe entre esta importancia conceptual y el algoritmo de cálculo del próximo learning object a sugerir.

La ontología es instanciable para que AdapTeach pueda utilizarse para cualquier contenido que se desee enseñar. Más adelante se detallará la estructura de la ontología y la forma de instanciar la misma.

Restricciones generales y de contenido.

Existen diferentes tipos de restricciones, aquellas que son generales, que se aplican a cualquier contenido a enseñar. Y aquellas restricciones que dependen del conocimiento del dominio, a las que se ha llamado restricciones de contenido.

Los pre-requisitos para el aprendizaje de un determinado learning object, podrían considerarse como *restricciones del contenido*. Es decir, un pre-requisito representa qué learning objects deberían haber sido aprendidos para poder aprender otro. No todos los learning objects tendrán pre-requisitos. Si todos tuvieran pre-requisitos o si los requisitos fueran muchos, el algoritmo podría estar muy limitado, teniendo muy pocas opciones de recomendación, y en el peor de los casos, solo podría recomendar una sola opción, cayendo en una secuencia fija de lecciones como la que se da en un sistema de e-learning tradicional.

Las *restricciones generales* son reglas a tener en cuenta en el razonamiento de AdapTeach, a la hora de sugerir un siguiente learning object, pero no tienen que ver con un contenido particular.

Algunos ejemplos de estas restricciones son:

- ◊ No sugerir un learning object si el usuario está situado en ese learning object.
- ◊ No sugerir un learning object si el learning object sugerido por AdapTeach tiene un pre-requisito que no ha sido cumplido.
- ◊ No sugerir un learning object sugerido por AdapTeach si existe alguna preferencia del usuario que está en conflicto con ese learning object. Si hubiera conflicto podría considerarse preguntar al usuario si desea acceder igual.

Las restricciones generales estarán representadas a modo de condiciones en el algoritmo de sugerencia.

Base de casos: la experiencia de AdapTeach.

La base de casos contiene la experiencia de AdapTeach. La base de casos será cargada inicialmente por un experto en el tema a enseñar, pero no solo en el tema, sino experto en la "enseñanza" de ese tema. Esto es importante porque según cómo sea ingresada inicialmente la base de casos, el sistema tendrá más posibilidades de acertar con sus sugerencias cuando sea utilizado.

El experto cargará los casos iniciales indicando qué learning object se debería sugerir ante determinado estado del usuario. Cuantos más casos se ingresen en un comienzo, más experiencia inicial tendrá el sistema y más acertadas pueden ser sus sugerencias.

Una base de casos completa permitiría tener más de un learning object a sugerir ante determinado estado del usuario. Se espera que el sistema sugiera siempre más de un learning object, aunque siempre el sistema establecerá las prioridades entre estos.

La base de casos irá creciendo a medida que los diferentes usuarios utilicen el sistema. Más adelante en este capítulo se describirá la estructura de la base de casos y se mostrará la forma de aprender de AdapTeach.

Perfil del usuario.

El perfil del usuario contiene todo aquello que ha sido indicado explícitamente por el usuario (como sus preferencias), y todo aquello que el sistema ha registrado durante el proceso de aprendizaje del usuario: los puntajes obtenidos en cada learning object, las sugerencias que aceptó, las que no aceptó, la historia de navegación, etc.

Estas cuestiones podrían clasificarse en dos categorías:

- ◊ los hechos de los cuales el sistema no tiene dudas (porque fueron indicados explícitamente por el usuario)
- ◊ aquellas cosas que considera que son ciertas pero no puede estar totalmente seguro, es decir, las conclusiones "inferidas".

Podríamos llamar a estas cuestiones *hechos* y *creencias*, dado que estos términos son parte de la terminología en los enfoques de diseño de agentes y en este caso particular representan justamente lo que dichos términos representan en esos enfoques.

Creencias

Como se ha mencionado anteriormente, una *creencia* es aquello que el sistema asume como verdadero, aunque no existe una certeza absoluta porque no ha sido indicado explícitamente por el usuario.

AdapTeach tendrá en cuenta las siguientes situaciones a modo de creencias:

- ◊ un determinado learning object fue una buena sugerencia si el usuario obtiene un puntaje razonable en el mismo y opina que la sugerencia fue aceptable.
- ◊ el usuario no está interesado en el learning object sugerido, si luego de una sugerencia decide acceder directamente a otro learning object.
- ◊ el usuario comprendió bien un learning object si obtiene un puntaje razonable en los ejercicios.
- ◊ un caso es bueno si se ha aplicado con éxito en una cantidad de usuarios. Que el caso se aplique con éxito significa que el usuario obtuvo un puntaje razonable en el learning object sugerido y además opinó que la sugerencia fue adecuada.
- ◊ el learning object no fue aprendido si se quedó menos de un tiempo determinado (no estuvo el tiempo suficiente para leer)o se quedó más de un tiempo en él (se quedó tanto tiempo que podría asumirse que no estaba leyendo sino haciendo otra actividad).
- ◊ El usuario no aprendió un learning object si no respondió ninguna pregunta ni realizó ejercicio.

Hechos

Un *hecho* es aquello de lo cual el sistema no tiene dudas porque ha sido indicado explícitamente por el usuario. Un hecho puede haber sido una creencia que el usuario confirmó explícitamente. Ejemplos de hechos en AdapTeach son:

- ◊ El sistema sabe que a un usuario le pareció adecuada una sugerencia por su feedback explícito. En este caso se trata de una creencia que se transforma en hecho.
- ◊ El sistema conoce las preferencias explícitas de un usuario: idioma que prefiere, color, etc.

4.2.2. La interfaz

La interfaz es el componente mediante el cual el usuario interactúa con el sistema. Tiene dos grandes componentes:

- ◊ El observador
- ◊ El comunicador

También en este punto se ha adoptado la terminología utilizada en los enfoques de diseño de agentes.

Observador

El observador se encarga de registrar todo lo que el usuario hace, es decir, "observa" las acciones y las registra. Algunos ejemplos de acciones observadas son:

- ◊ Solicitud de un learning object para visitar.
- ◊ Navegación hacia a un learning object sugerido.
- ◊ Hora de acceso a un learning object.
- ◊ Hora de salida de un learning object.
- ◊ Navegación hacia un learning object no sugerido, evitando la sugerencia del sistema.
- ◊ Ingreso de la opinión del usuario respecto al learning object sugerido, para indicar qué grado de dificultad tuvo.
- ◊ Ingreso y resolución de respuestas a preguntas y ejercicios.
- ◊ Solicitud de evaluación del learning object.
- ◊ Cambio de valores de las preferencias del usuario.

Comunicador

El comunicador es el encargado de informar a usuarios y expertos que administran los cursos que se dictan con AdapTeach. Informa a usuarios y expertos. Al usuario informa todo lo que éste necesita para su proceso de aprendizaje, y al experto que entrena el sistema, informa todo lo que necesita para ajustar los razonamientos del sistema.

Qué se informa al usuario alumno?

- ◊ Learning object sugeridos en orden de importancia.
- ◊ Contenido del learning object a visitar cuando el usuario decide acceder.
- ◊ Razonamiento llevado a cabo por el sistema para sugerir.
- ◊ Estado mental: creencias, hechos, perfil de usuario, restricciones, base de casos, etc.
- ◊ Learning objects sugeridos para la navegación a partir del learning object actual.
- ◊ Estado inicial y actual del usuario con los puntajes obtenidos en cada learning object.
- ◊ Learning objects que se encuentran definidos para el curso con el cual se está trabajando.

Qué se informa al experto entrenador?

- ◊ Casos de la base de casos utilizados para el razonamiento.
- ◊ Razonamientos realizados: situación, learning object sugerido y resultados.
- ◊ Historia de navegación de un alumno particular.
- ◊ Sugerencias no aceptadas por el alumno, con el caso que dio origen a la sugerencia.

4.2.3. El cerebro

El cerebro contiene todos los mecanismos para poder tomar decisiones. Interactúa con el estado mental tomando de él todo lo que necesita para tomar las decisiones, y también lo alimenta con nuevos razonamientos y conclusiones.

Los componentes del cerebro son:

- ◊ Razonador.
- ◊ Analizador.
- ◊ Controlador de ontología.
- ◊ Controlador principal.

Razonador

Está compuesto por el motor que utiliza Razonamiento basado en casos (CBR) para determinar los siguientes learning objects a sugerir. Como se ha mencionado antes, el razonador podría ofrecer más de un learning object, ya que ante un determinado estado de puntajes, puede haber más de un posible learning object a visitar, aunque generalmente con diferentes prioridades.

El razonador debe tener en cuenta las restricciones generales y de contenido mencionadas anteriormente, a la hora de sugerir un learning object.

Más adelante se explicará cómo funciona el algoritmo de razonamiento basado en casos para determinar el próximo learning object a sugerir.

Controlador de ontología

El controlador de la ontología se encarga de interactuar con la misma para obtener todo el conocimiento sobre el contenido de un curso. El controlador interviene en las siguientes situaciones:

- ◊ Cuando el usuario selecciona un curso, solicita los learning objects correspondientes a ese curso.
- ◊ Cuando el algoritmo está calculando el próximo paso a seguir, solicita los pre-requisitos de los learning objects obtenidos.
- ◊ Cuando el usuario acepta una sugerencia de learning object para continuar, solicita el contenido del mismo a la ontología

Controlador principal

El controlador principal es el que interactúa con los tres componentes del cerebro de AdapTeach: el razonador, el analizador y el controlador de la ontología; y con los componentes que interactúan con el usuario. Este componente es una especie de delegador de funciones, según lo que debe hacer el sistema, delega responsabilidades a los demás componentes.

Analizador

El analizador toma lo registrado por el observador y decide qué incluir en el conocimiento de AdapTeach, es decir, decide qué cosas pueden resultarle de interés en el futuro para ajustar sus decisiones. Básicamente el analizador puede hacer tres cosas:

- ◊ Agregar un caso en la base de casos.
- ◊ Actualizar el perfil del usuario.
- ◊ Detectar si el usuario leyó o no el learning object actual, para poder determinar si mostrar nuevamente un learning object que es clave en el aprendizaje, mostrando un mensaje de advertencia.

Agregar un caso en la base de casos

El analizador determinará si un nuevo caso, obtenido de la interacción de un usuario particular, es útil para ser agregado en la base de casos. Más adelante se explicará de qué manera funciona el aprendizaje de AdapTeach.

Actualizar el perfil de usuario

El analizador podría detectar algún cambio del perfil del usuario, que no ha sido indicado explícitamente por este. Por ejemplo, supongamos que el usuario definió como preferencia que desea ver en modo demo los PLO. Sin embargo, cada vez que se le ofrece este modo, accede al modo interactivo. Entonces, el analizador podría asumir que su preferencia ha cambiado. En este caso, podría preguntar al usuario si desea cambiar dicha preferencia.

Detectar si un usuario leyó un learning object

Si el usuario solicita un learning object sin haber respondido ninguna pregunta o ejercicio, habiéndose quedado en el learning object más de un tiempo *x* o menos de un tiempo *y*, el sistema puede inferir que el usuario no leyó el contenido. En este caso, podría incluirse en el estado mental que el learning object no fue leído, a modo de creencia. Si el learning object no fue leído, el razonador podría sugerirlo de nuevo indicando con un mensaje esta creencia, para que el usuario decida si acceder a la sugerencia igual.

Si el usuario solicita un learning object habiendo respondido preguntas, habiendo solicitado su evaluación, y el tiempo de estadía en el learning object está en un rango adecuado, entonces se asume que lo leyó.

4.3. La Ontología OntoTeach

La ontología construida lleva el nombre de OntoTeach (Ontology For Teaching Systems). Ésta contiene, tal como se ha mencionado anteriormente, todo el conocimiento que se desea enseñar, estructurado en objetos de aprendizaje o Learning Objects.

La Ontología ha sido construida utilizando el editor de Ontologías Protégé, que permite construir ontologías en un entorno gráfico y generar el archivo en lenguaje OWL que será luego procesado por el sistema adaptativo.

4.3.1. Estructura de OntoTeach

Una ontología consta de *clases* para la definición del conocimiento a representar. En este caso OntoTeach modela cursos adaptativos de enseñanza, por lo cual las clases que representan ese dominio incluyen, entre otras: AdaptiveCourse, Lesson, LearningObject, LearningObjectContent.

Cada clase tiene propiedades o *slots*, que describen su estructura. Así, por ejemplo, la clase LearningObject tiene slots como: hasID, hasName, hasScoreWeight, hasLearningObjectContents, y otros.

A continuación se muestra la estructura de clases y slots de OntoTeach, sin incluir instancias, de las que se hablará más adelante.

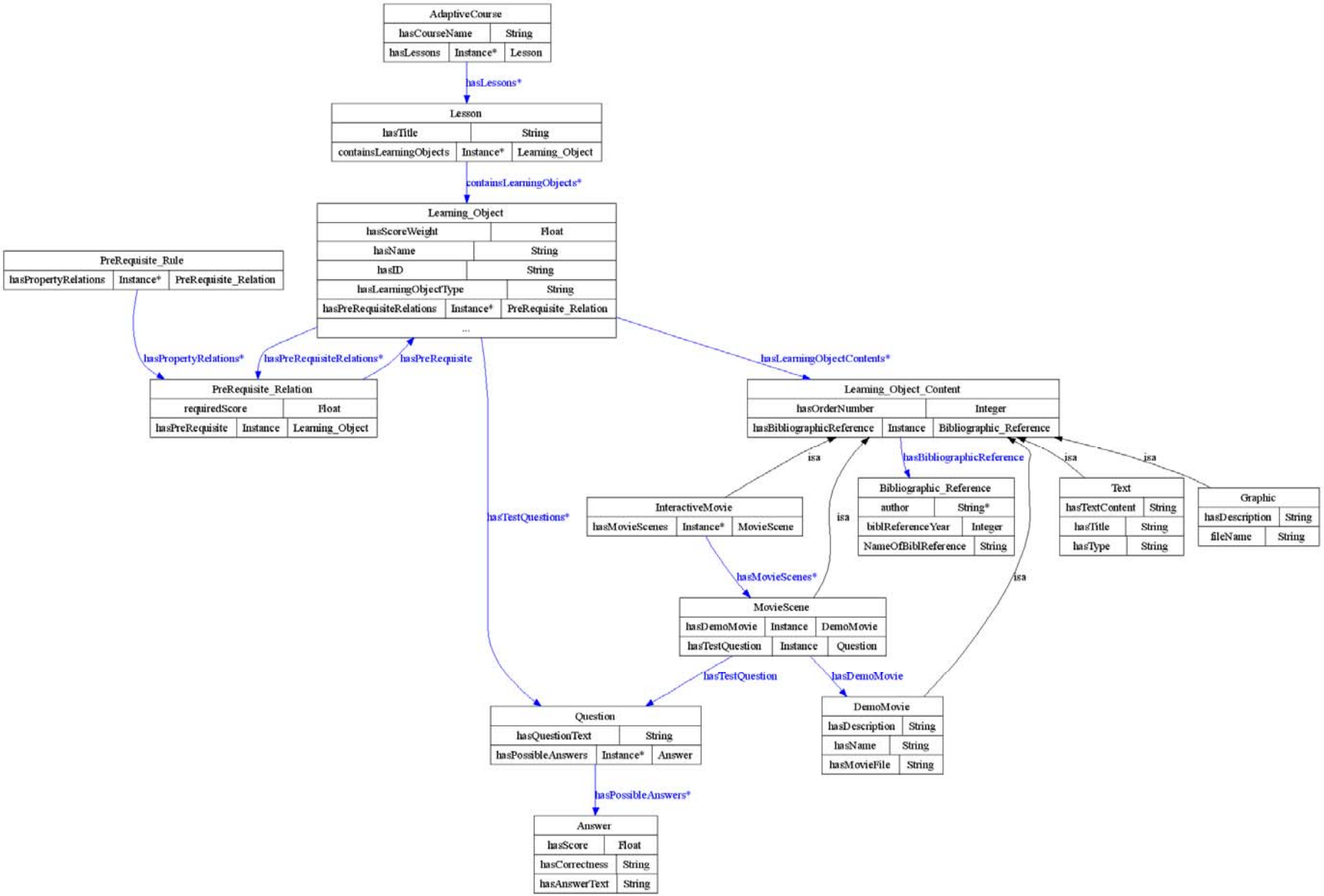


Figura 7: Estructura general de la Ontología OntoTeach, sin instancias

Para poder utilizar la ontología en el sistema de enseñanza, debemos crear las instancias de las clases definidas. Así la clase *AdaptiveCourse* tendrá varias instancias, una por cada curso definido en nuestra ontología. Y la clase *LearningObject* tendrá muchas instancias, una para cada learning object dentro de un curso definido. Las instancias se relacionan entre sí, indicando por ejemplo, que una determinada instancia de *Learning Object*, pertenece a una determinada instancia de *Lesson* y ésta a su vez pertenece a una instancia de *AdaptiveCourse*.

El corazón de la ontología está formado por las clases LearningObject y LearningObjectContent.

Una instancia de LearningObject está vinculada a muchas instancias de LearningObjectContent porque un learning object puede tener varios componentes, de diferentes tipos (texto, gráfico, película, etc.).

En el capítulo 5, que describe la forma de instanciar OntoTeach, se mencionarán las diferentes clases y slots que integran la ontología.

4.4. El perfil de usuario de AdapTeach

Cada alumno que utilice AdapTeach, tendrá definido un perfil de usuario en el cual se guardarán todas sus preferencias y su historia de aprendizaje. Esto es, todos los puntajes obtenidos en los learning objects evaluados y la historia de navegación.

El perfil de un usuario se puede almacenar, a modo de ejemplo, en un archivo XML. Cada usuario podría tener asociado un archivo XML diferente. A continuación se muestra el contenido de un usuario a modo de ejemplo, en lo que se refiere a los puntajes obtenidos y la historia de navegación:

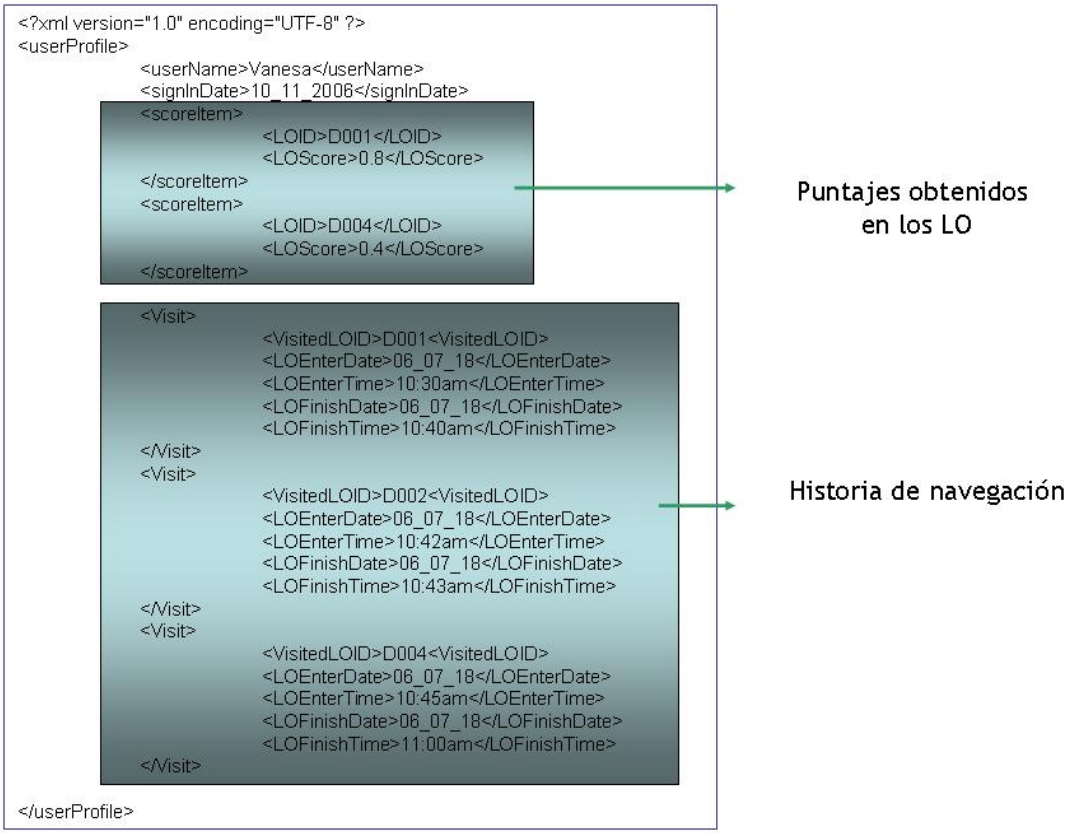


Figura 8: Ejemplo de XML correspondiente a un usuario

Cada alumno tendría asociado un XML con su perfil. Este XML es procesado por AdapTeach a la hora de calcular la próxima sugerencia de learning objects, ya que ésta se calcula en base a los puntajes obtenidos en los diferentes learning objects.

Cuando el usuario accede a un learning object, el perfil se actualiza con una nueva sección de Visita, almacenando hora y fecha de ingreso al learning object.

Cuando el usuario solicita un nuevo learning object o abandona el sistema, el perfil se actualiza con los puntajes obtenidos en el learning object actual en la sección correspondiente al learning object (por ejemplo, <LOID: D001>, y con la fecha y hora de salida del learning object en la sección de Visita.

4.5. La inteligencia de AdapTeach

Como se ha mencionado anteriormente, el algoritmo de sugerencia de próximos learning objects a visitar, está basado en una técnica conocida como Razonamiento basado en casos o en inglés, CBR (Case-Based Reasoning).

Esta técnica se basa en simular la experiencia de un experto en forma de casos. Los casos, en esta aplicación, se refieren al estado de un alumno en un momento dado del aprendizaje y el paso a seguir según el punto de vista de un docente experto.

4.5.1. Aplicación de CBR para calcular la próxima sugerencia

Los casos en su conjunto representan la experiencia de un docente experto, quien determina qué debería explicar a un alumno que ha aprendido ciertos conocimientos y no otros.

Un caso entonces tiene información sobre ciertos puntajes obtenidos en los diferentes learning objects. Algunos casos tendrán puntajes correspondientes a un usuario particular, la sugerencia realizada por el sistema y el resultado de esa sugerencia. Otros, en cambio, tendrán puntajes de usuarios y sugerencias, pero no de un usuario particular sino de un usuario genérico, es decir, que representan la experiencia del docente en general. Más adelante se explican las diferencias entre los casos.

A los distintos parámetros o datos que se guardan en un caso, se los denomina "dimensiones del caso".

4.5.2. Dimensiones del Caso

Las dimensiones de un caso para el cálculo del próximo paso en el aprendizaje, se enumeran a continuación:

- ◊ Número de caso.
- ◊ Tipo de caso.
- ◊ Puntajes de los learning objects en una situación particular.
- ◊ Learning object sugerido en esa situación descripta por los puntajes.
- ◊ Motivo por el cual el learning object debería ser sugerido.
- ◊ Fecha de creación: fecha de creación del caso.
- ◊ Hora de creación: hora de creación del caso.
- ◊ Nombre de usuario: nombre del usuario que dio origen al nuevo caso.
- ◊ Caso más similar: este es el caso más similar al nuevo caso agregado en la base.
- ◊ Resultado de sugerencia: indica si el resultado es positivo o negativo.

A continuación se muestra un ejemplo de definición de un caso en formato XML.

```
<?xml version="1.0" encoding="UTF-8" ?>
<CaseBase>
……

        <case>
                <CaseNumber>5</CaseNumber>
                <CaseType>Training</CaseType>
                <Scores>
                        <LOScore>
                                <LOID>D002</LOID>
                                <Score>0.8</Score>
                        </LOScore>
                        <LOScore>
                                <LOID>D003</LOID>
                                <Score>0.8</Score>
                        </LOScore>
                        <LOScore>
                                <LOID>D004</LOID>
                                <Score>1.0</Score>
                        </LOScore>
                        <LOScore>
                                <LOID>D008</LOID>
                                <Score>0.9</Score>
                        </LOScore>
                </Scores>
                <SuggestedLO>
                        <SuggestedLOID>P002</SuggestedLOID>
                        <MostSimilarCase>0</MostSimilarCase>
                        <Why>
                                The user has learned correctly what is a contract and its
                                example. He is able to understand a more difficult example
                                or to know how to build a contract in its three modes. It's
                                best to see the demo first, so this view is suggested
                        </Why>
                </SuggestedLO>
                <Result> Positive  </Result>
        </case>
…….

</CaseBase>
```

Figura 9: Ejemplo de caso en XML

En el anexo I se describen con mayor detalle las dimensiones de los casos, su importancia y de donde se obtienen los datos de cada una de ellas.

4.5.3. Adaptabilidad navegacional con CBR en AdapTeach

En una clase real, el siguiente paso en la enseñanza (qué concepto, ejercicio, etc.) es determinado por el docente según su experiencia, porque debe identificar qué necesita repasar o visitar el alumno para reforzar cierto conocimiento o simplemente para avanzar en el aprendizaje.

En el sistema que simula la experiencia del docente, la forma en que esa experiencia se introduce en el sistema es clave ya que puede aumentar la exactitud con la que el sistema recomienda los learning objects que el alumno realmente necesita.

Obtención de los casos más similares a la situación actual.

La sugerencia del learning object que un usuario debería visitar en determinado momento tiene que ver con:

- ◊ Los puntajes obtenidos hasta el momento en cada uno de los learning object.
- ◊ Las preferencias que puedan determinar la selección de un learning object u otro.
- ◊ La experiencia de AdapTeach.

En cada momento del aprendizaje es posible que el sistema tenga varios learning objects para sugerir como paso siguiente. El problema radica en identificar cuales y en qué orden se sugieren.

La aplicación de CBR permite tener una experiencia y evaluar una situación actual según esa experiencia. Así, si una situación actual no coincide exactamente con un caso, el algoritmo CBR permite identificar el más parecido a ella y aplicar la solución del caso.

4.5.4. La comparación entre los casos y la situación nueva.

Para poder calcular la similitud entre dos casos, es necesario comparar cada una de las dimensiones del mismo.

El razonamiento está dado por las funciones de similitud de las dimensiones definidas en el caso CBR. Las funciones de similitud determinan qué tan similar es un valor de una dimensión entre un caso y otro.

Según cual sea el contenido de una dimensión, la función de similitud puede variar. En una primera versión de Adapteach, solo está considerada la función de similitud que evalúa similitud entre puntajes obtenidos en un learning object. Sin embargo, es posible extender el diseño y considerar diferentes tipos de funciones de similitud.

Función de similitud para evaluar el puntaje de un LO.

Como se ha mencionado, la función de similitud de puntajes permite evaluar qué tan similar es una dimensión de un caso respecto de la misma dimensión de otro caso. Así, por ejemplo, podemos comparar el puntaje obtenido en un learning object de un usuario particular en un momento dado, con un caso guardado en la base de casos.

Supongamos los siguientes valores:

- ◊ Valor 1= puntaje obtenido en el LO, caso nuevo
- ◊ Valor 2= puntaje obtenido en el LO, caso recuperado de la base de casos

Para comparar se aplica el siguiente algoritmo:

Valor1 +/- i = Valor2 >> similitud = (10 – i) / 10

Con esta fórmula obtenemos los siguientes ejemplos de similitud:

Valor1 = Valor2	→ similitud = 1;
Valor1+1 = Valor2 ó Valor1-1= Valor2	→ similitud=0.9;
Valor1+2 = Valor2 ó Valor1-2= Valor2	→ similitud =0.8;
Valor1+3 = Valor2 ó Valor1-3= Valor2	→ similitud =0.7;
Valor1+4 = Valor2 ó Valor1-4= Valor2	→ similitud =0.6;
Valor1+5 = Valor2 ó Valor1-5= Valor2	→ similitud =0.5;
Valor1+6 = Valor2 ó Valor1-6= Valor2	→ similitud =0.4;
Valor1+7 = Valor2 ó Valor1-7= Valor2	→ similitud =0.3;
Valor1+8 = Valor2 ó Valor1-8= Valor2	→ similitud =0.2;
Valor1+9 = Valor2 ó Valor1-9= Valor2	→ similitud =0.1;
Valor1+10= Valor2 ó Valor1-10= Valor2	→ similitud =0;

Determinación de la similitud entre dos casos

El resultado de la función de similitud determina el porcentaje de similitud de una sola dimensión. Si tomamos todas las dimensiones de un caso, podemos evaluar la similitud entre dos casos.

Así, por ejemplo, tenemos un caso guardado en nuestra base (caso1), ingresado en modo entrenamiento, que indica qué caso es el más recomendable ante determinados valores de puntajes.

Y continuando con el ejemplo, tenemos un caso nuevo (caso 2) que representa los puntajes obtenidos por un usuario real que está utilizando el sistema y que desea que el mismo le sugiera el próximo paso a seguir.

Para calcular la similitud entre dos casos, aparece un nuevo parámetro que representa la importancia que tiene cada learning object para el proceso de aprendizaje. La importancia, en AdapTeach, está definida en la ontología, junto con la definición de cada learning object.

El cálculo de similitud entre casos se realizará sumando el valor de la función de similitud de cada dimensión multiplicada por la importancia del learning object. Dividiremos esta suma por la suma de las importancias de los learning objects.

$$\frac{\begin{array}{l} \text{Sim Dim1(caso1/caso2)} * \text{impDim1} \\ + \text{Sim Dim2(caso1/caso2)} * \text{impDim2} \\ + \ldots\ldots \\ + \text{Sim DimN(caso1/caso2)} * \text{impDimN} \end{array}}{\text{ImpDim1} + \text{ImpDim2} + \ldots + \text{ImpDimN}}$$

Cálculo de similitud entre dos casos

La división por la suma de las importancias nos permite obtener un valor de similitud entre 0 y 1, y de esta manera no solo podemos comparar las diferentes similitudes para saber qué caso es más parecido al que estamos comparando, sino también nos permite determinar en qué porcentaje es similar un caso con otro.

Por ejemplo, supongamos que luego de hacer todas las comparaciones, tenemos los siguientes valores de similitud con el caso nuevo

Sim (Caso nuevo – Caso 1):	0.96
Sim (Caso nuevo – Caso 10):	0.73
Sim (Caso nuevo – Caso 14):	0.72
Sim (Caso nuevo – Caso 2):	0.50
Sim (Caso nuevo – Caso 4):	0.42
…..	
Sim (Caso nuevo – Caso 5):	0.22
Sim (Caso nuevo – Caso 12):	0.12

Podemos observar que el caso 1 tiene una similitud de 0.96, que es la más alta. Por lo tanto podemos saber que el caso 1 es el más similar al caso nuevo. Pero además, sabemos que el caso nuevo con el caso 1 es un 96% similar, lo cual nos da información muy valiosa.

AdapTeach permite configurar un valor umbral para obtener los casos similares, que permite definir hasta qué porcentaje de similitud es aceptable. Es decir, si definimos este umbral igual a 0.7, estaremos diciendo que solo consideraremos similares aquellos casos que superen ese valor, al ser comparados con el caso nuevo.

Si en un determinado momento el usuario solicita un nuevo learning object, y la comparación con los casos de la base no da como resultado algún caso que supere el umbral, el sistema deberá tener en cuenta algún otro mecanismo de sugerencia. Podría ser que en este caso se sugiera un learning object fijo que seguiría en una secuencia típica del contenido que se está enseñando.

Extensibilidad del framework para el cálculo de similitud

Como se ha mencionado antes, el framework está definido de manera tal que puede ser extensible en lo que respecta al razonamiento que puede realizar. El sistema puede extenderse definiendo nuevas dimensiones a los casos, con sus respectivas funciones de similitud. El uso del patrón de diseño Strategy en el diseño de AdapTeach permite llevar a cabo la extensibilidad, simplemente agregando una clase *SimilarityFunction* que calcule la función de similitud para una dimensión dada. El algoritmo de cálculo de la similitud dependerá de los valores posibles de la dimensión.

4.5.5. La selección de los mejores Learning objects a sugerir.

Una vez realizada la comparación del caso nuevo (situación de un alumno que solicita un nuevo learning object a visitar), contra todos los casos de la base de casos, podemos ordenarlos según el valor de similitud. Aquellos con mayor valor serán los más parecidos.

Tomamos todos aquellos casos cuya similitud con la nueva situación superen el umbral de calidad que definimos.

Que un caso tenga una similitud aceptable no implica que sea elegido por AdapTeach, ya que a la hora de mostrar el learning object, debemos chequear algunas otras condiciones. De esta manera deberíamos recorrer la lista de learning objects obtenidos en la comparación, desde el que más similitud tiene, y chequear los siguientes puntos:

- ◊ Si el caso seleccionado tiene resultado negativo, se debe evitar sugerir el learning object del mismo.
- ◊ Si el learning object obtenido es el que el usuario está visitando actualmente, debería evaluarse si ofrecerlo o no.
- ◊ Si existe un pre-requisito de contenido que hace que no se pueda mostrar el learning object obtenido, no se debe mostrar ese learning object

Si alguna de las tres situaciones se da, el caso debe saltearse y continuar con el siguiente según la similitud.

Analizamos caso por caso a continuación:

El caso tiene un resultado negativo

Si el caso tiene el valor "*negative*" en la dimensión de resultado, significa que ante un alumno con los puntajes definidos en las dimensiones de puntajes ("*score*"), debe evitarse la sugerencia de el learning object que contiene la dimensión "*suggestedLearningObject*".

El learning object obtenido es el mismo que el que está siendo visitado:

En este caso no es conveniente sugerir el mismo learning object al actual, ya que el usuario puede no estar interesado en él. Para decidir si se debe sugerir o no, podemos tratar de identificar si el usuario realmente leyó o no el contenido del learning object actual. Para esto se podrá tomar la hora de ingreso al learning object. Si el tiempo transcurrido es menor a cierto valor (no tuvo tiempo de leer) o mayor a cierto valor (estuvo tanto tiempo que se podría inferir que estuvo haciendo otra tarea fuera del sistema), entonces se podría sugerir el learning object con un aviso, dejando la opción al usuario de acceder o no.

Existe un pre-requisito de contenido

En este caso, el learning object obtenido no puede sugerirse al usuario, porque no completó los pre-requisitos necesarios, es decir, no obtuvo el puntaje necesario en los learning objects requeridos.

Los pre-requisitos se definen en la ontología junto con la definición de los learning objects. Sin embargo, una buena base de casos que contenga la experiencia del docente, no debería obtener learning objects que tengan pre-requisitos no cumplidos, ya que esto implicaría que el caso no está bien definido.

Sin embargo, los pre-requisitos son una forma de identificar casos que deben refinarse, ya que el sistema guardará los learning objects sugeridos que no pueden mostrarse, a modo de historia. Si se diera el caso una varias veces, el experto podría decidir eliminar el caso de la base o modificarlo.

4.5.6. Aprendizaje de AdapTeach: evolución de la base de casos

Un sistema adaptativo debería aprender con el tiempo. En este caso particular, el sistema debería ajustar sus sugerencias, es decir, aprender a sugerir mejor cada vez más, en base a su experiencia.

La experiencia de AdapTeach está representada en la base de casos. Cada caso representa una situación particular de un alumno con el learning object a sugerir ante esa situación.

La experiencia representada en la base de casos puede ser experiencia del docente que entrenó el sistema, o bien experiencia que logró a través de la interacción con los usuarios.

Durante el uso del sistema, se puede determinar el agregado de un caso nuevo a la base, teniendo en cuenta ciertos parámetros para asegurar la validez del caso. Más adelante se detalla la forma en que AdapTeach analiza el agregado de una nueva experiencia o caso.

Casos de entrenamiento y casos aprendidos.

Cuantos más casos haya en la base, más apropiada será la sugerencia y mejor será el aprendizaje del sistema en el futuro.

Existen dos tipos de casos en la base de casos:

- ◊ Casos de entrenamiento (Training cases)
- ◊ Casos aprendidos (Learned cases)

La diferencia entre estos tipos de casos, tiene que ver con si el caso fue incluido en la base en la fase de entrenamiento del sistema o si fue aprendido por el mismo durante la ejecución.

Los *casos aprendidos* son aquellos que fueron incorporados a la base durante el funcionamiento del sistema, a partir de la interacción con los usuarios y las inferencias realizadas por AdapTeach.

Los *casos de entrenamiento* representan la experiencia inicial del sistema. Son los casos incluidos por el entrenador o experto. Sin los casos de entrenamiento, el sistema no podría sugerir de manera confiable, ya que no tendría ninguna experiencia.

En su estructura no existe diferencia entre ambos tipos de casos. Sin embargo, existe una diferencia conceptual que puede ser utilizada en el futuro para analizar si AdapTeach aprende de manera correcta y para analizar la eficacia de los diferentes tipos de casos. Por ejemplo, se podría obtener como conclusión que los casos generados por un experto son mejores que los casos identificados por AdapTeach o viceversa.

Cabe aclarar que el agregado de un nuevo caso, siempre requiere la intervención del experto. Lo que hace AdapTeach es identificar posibles casos para agregar, pero no los agrega sin aprobación del experto.

¿Cuando se agrega un caso aprendido a la base de casos?

Cada vez que un usuario solicita un learning object, el algoritmo determina cuales son los posibles pasos a seguir. Si el usuario accede al learning object sugerido, el sistema registra los siguientes datos (al salir del learning object):

- ◊ Puntajes al anteriores a la sugerencia
- ◊ Caso obtenido por el algoritmo
- ◊ Learning object sugerido
- ◊ Puntaje obtenido en el learning object sugerido
- ◊ Opinión del usuario sobre la sugerencia

Cuando el sistema ha sido utilizado por varios usuarios, este log tiene información muy valiosa para ajustar el entrenamiento del sistema, es decir, para que aprenda de la experiencia.

AdapTeach aprende con casos positivos y negativos.

Los *casos positivos* son aquellos en los cuales la sugerencia fue aceptable. Una sugerencia es aceptable si la misma tuvo un efecto positivo en la enseñanza, es decir, si una cantidad considerable de usuarios obtuvo un puntaje aceptable en el learning object y además opinaron que la sugerencia había sido buena.

Los *casos negativos* son aquellos que indican que debe evitarse sugerir un learning object ante determinada situación. Estos casos negativos podrían ser ingresados en modo entrenamiento, representando lo que el docente considera que debe evitarse. O podrían aparecen como consecuencia del uso del sistema, donde varios usuarios no tuvieron un buen desempeño en el learning object sugerido y/o su opinión no fue aceptable.

Debido al grado de subjetividad que puede influir a la hora de evaluar un caso, y debido a los diferentes grados de dedicación de los usuarios y de sus opiniones, AdapTeach no toma cada nueva situación como un nuevo caso, sino que arma un nuevo caso en base a muchas situaciones almacenadas en el tiempo.

Por ejemplo, podríamos tener los siguientes datos que representan la sugerencia a varios usuarios.

Usuario	Fecha y hora	Puntajes anteriores a sugerencia	ID Caso obtenido	Learning object sugerido	Puntaje obtenido LO Sugerido	Opinión del usuario	Resultado
Nicolás	7/11/06 10:30	D001: 0.9 D004: 0.5 P001: 0.8	Case 1	D002	0.7	Difícil pero aceptable	Positivo
Franco	10/10/06 11:02	D001: 0.8 D004: 0.6 P001: 0.6 P002: 0,7	Case 1	D002	0.9	Difícil pero aceptable	Positivo
Juan Manuel	10/01/06	D001: 0.9 D002: 0.1 D003: 0.2 P001: 0.7	Case 1	D002	1.0	Fácil pero aceptable	Positivo

Si un mismo caso hubiera sido utilizado ante situaciones parecidas con un resultado positivo, AdapTeach podría sugerir agregarlos en la base de casos, ya que permitirían ajustar más las sugerencias futuras.

Las situaciones son parecidas si fue utilizado el mismo caso para la sugerencia. En la tabla anterior podemos detectar diferentes situaciones en las que fue tomado como referencia el mismo caso, por lo tanto son situaciones parecidas (pero no necesariamente iguales).

Al analizar las diferentes situaciones, AdapTeach verifica que el resultado haya sido igual (positivo o negativo). Si por ejemplo, se detecta que el resultado es positivo, AdapTeach sugiere agregar todos las situaciones nuevas como caso. De esta manera se puede ajustar la sugerencia en un futuro ya que habrá casos parecidos al original (Case 1) o casos parecidos a los nuevos casos, de los cuales ya se tiene más precisión sobre su resultado. Para ser agregados en la base de casos, AdapTeach siempre solicita intervención del experto.

La opinión del usuario

La opinión del usuario se toma como una medida subjetiva, que es tenida en cuenta a la hora de aceptar la inclusión de una situación como caso en la base de casos.

Cada vez que el usuario responde las preguntas y ejercicios de un learning object, tiene la posibilidad de dar su opinión sobre la sugerencia indicando uno de los siguientes valores:

- ◊ Muy fácil, no adecuado.
- ◊ Muy difícil, no adecuado.
- ◊ Fácil, pero adecuado.
- ◊ Difícil pero adecuado.

Este valor, junto con el puntaje obtenido en el learning object, permiten el experto evaluar una situación para considerar su inclusión como caso en la base de casos.

5. AOOTS: enseñando Orientación a Objetos utilizando AdapTeach.

Se presentará en este capítulo, una ejemplo particular que utiliza el framework AdapTeach para la enseñanza de Análisis y Diseño Orientado a Objetos (AOOTS: Adaptive OO Teaching System). Se mostrará una pequeña parte de un curso en este dominio pero se describirá en detalle cómo se instancia el framework para este contenido en particular.

5.1. Instanciación de AdapTeach.

Cuando se desea utilizar Adapteach para crear un curso en particular, debemos realizar las siguientes tareas:

- ◊ Instanciar la ontología con los learning objects (contenido a enseñar), pre-requisitos e importancias de cada learning object.
- ◊ Definir los casos de entrenamiento inicial de Adapteach.

AOOTS es una instancia de AdapTeach. En este capítulo se detallará el proceso de creación de AOOTS.

5.2. Instanciación de la ontología OntoTeach.

En el capítulo 4.3 se ha descrito la estructura de la ontología OntoTeach. OntoTeach tiene una estructura que permite ser instanciada con cualquier contenido que se desee enseñar. Es decir que si deseamos crear un curso sobre cualquier tema, podemos instanciar OntoTeach con los learning objects necesarios, sus importancias, pre-requisitos, etc.

5.2.1. Creación de instancias

Para instanciar la ontología es necesario crear todas las instancias de las clases que componen su estructura. A continuación se detalla cada una de ellas.

Creación de la instancia de Curso (AdaptiveCourse) AOOTS

Para instanciar OntoTeach con el contenido del curso, debemos primero crear una instancia de la clase "*AdaptiveCourse*", en este caso la instancia se llama AOOTS_OOAnalysisDesign.

Creación de instancias de Lecciones

Una vez creada la instancia de curso, debemos crear las instancias de las lecciones que contendrán a los learning objects.

Luego de crear las lecciones debemos vincular estas a la instancia de Curso creada anteriormente.

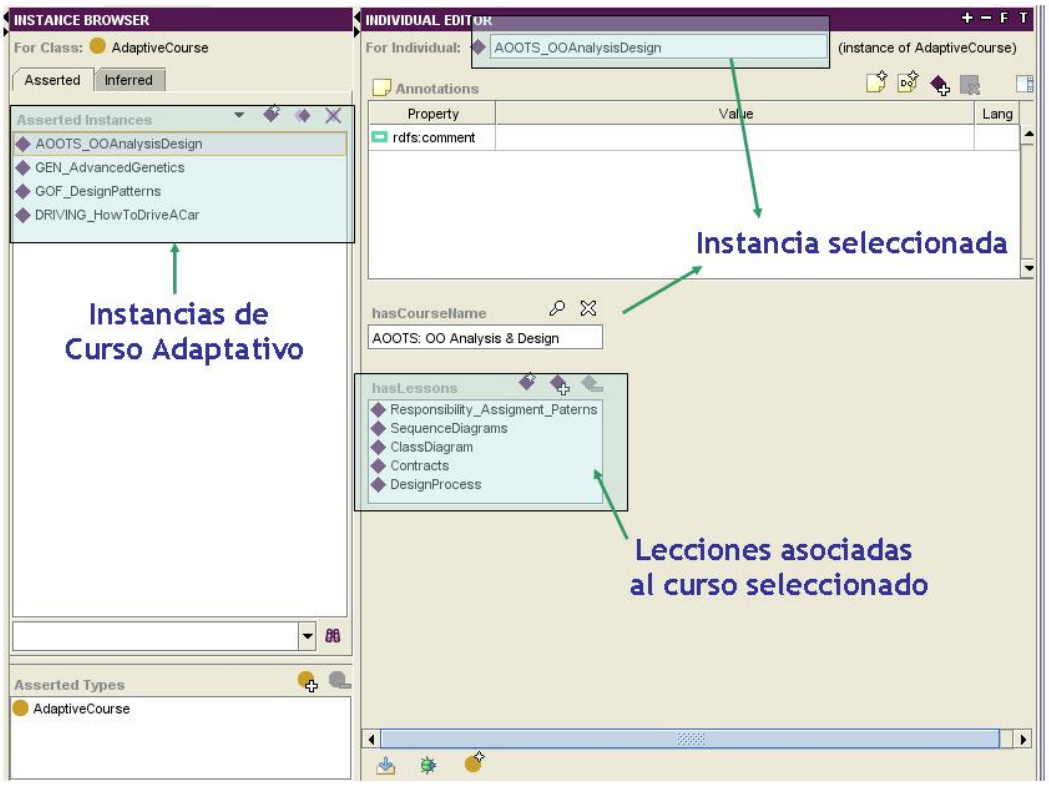


Figura 10: definición de la instancia de Curso con las lecciones asociadas

Creación de instancias de Learning Objects

Luego de definir la instancia de curso y lecciones, deben crearse los Learning Objects, que son las unidades de aprendizaje.

Cada learning object debe vincularse a una lección de las creadas antes.

La siguiente imagen muestra la lección con los learning objects asociados:

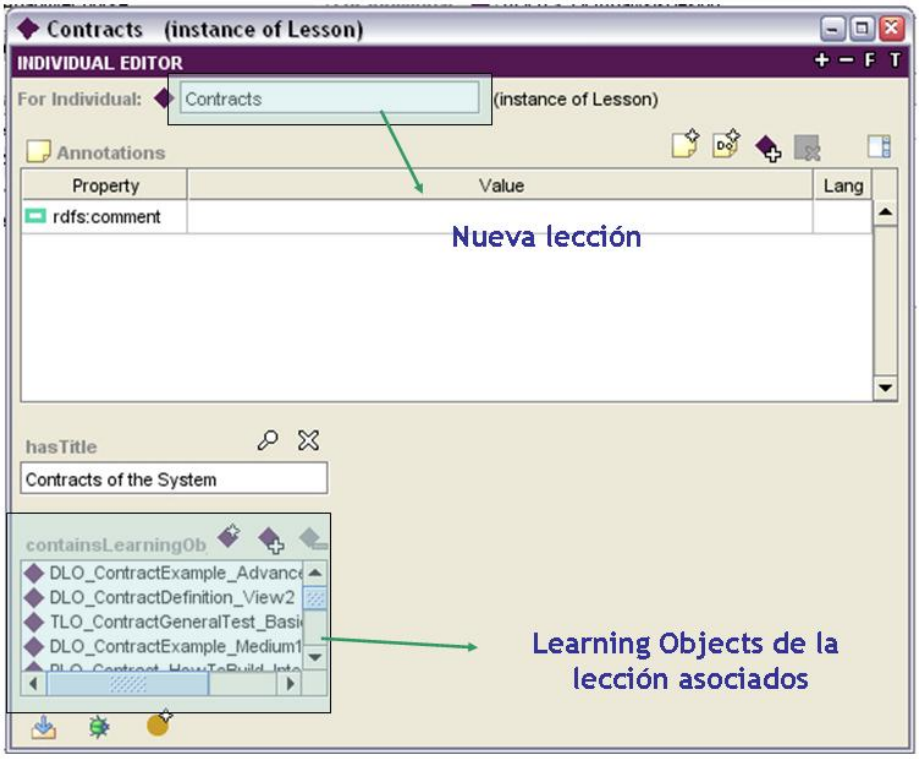


Figura 11: Instancia de Lección

A continuación se muestra una pantalla de Protégé con la definición de un Learning object de proceso (PLO) a modo de ejemplo:

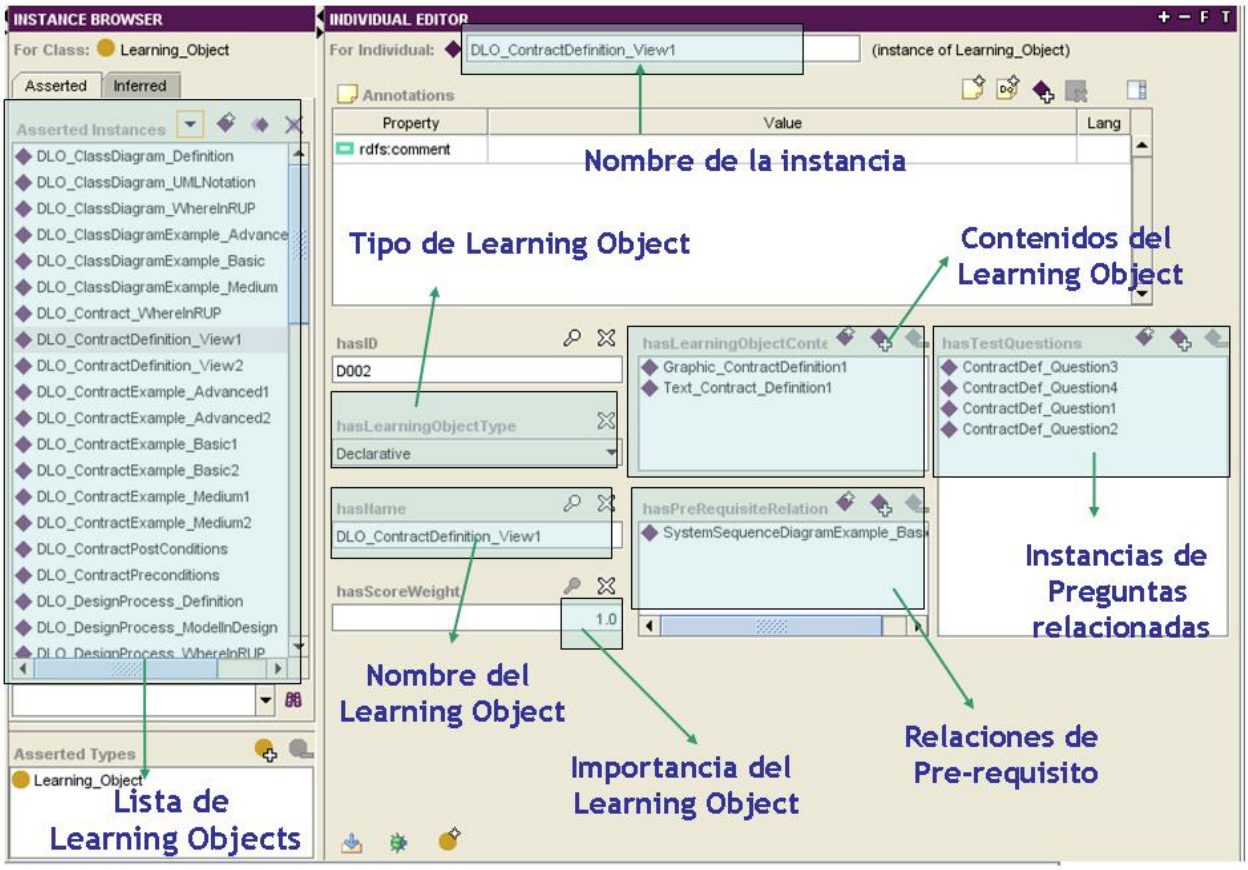


Figura 12: definición de una instancia de Learning Object

Como se puede ver en el gráfico de definición de un learning object, se deben proveer las referencias a los contenidos de ese learning object.

Definición de las importancias de los learning objects

La importancia de cada learning object es fundamental, ya que representa la importancia de un contenido particular para el aprendizaje de un dominio.

La importancia se tiene en cuenta en el cálculo de la similitud entre una nueva situación y un caso almacenado en la base.

Como podemos ver en la imagen anterior, la importancia se define en la ontología en el slot *hasScoreWeight*. En el ejemplo anterior, la importancia del learning object es de 0.8. Este se multiplicará por el puntaje obtenido en ese learning object a la hora de calcular las similitudes.

Definición de los pre-requisitos

Los pre-requisitos representan los learning objects que deben ser aprendidos para que otro learning object pueda ser visitado.

Los pre-requisitos representan restricciones de contenido. Para que el sistema adaptativo cumpla con su objetivo, un experto no debe excederse en la definición de estas restricciones.

Como se ve en la figura, los pre-requisitos se definen en cada learning object, con una referencia a todos los learning objects que deben completarse antes.

Para crear un pre-requisito, primero debe crearse la instancia de Relación de pre-requisito (PreRequisiteRelation). Esto es porque un mismo learning object puede ser pre-requisito de diferentes learning objects, pero con diferente importancia. Es decir, pueden variar en el porcentaje que se necesita para acceder.

En el ejemplo de la figura, un pre-requisito del learning object *PLO_Contract_HowToBuild_Static_Basic*, es el learning object *Contract_Definition_View1*, con un 70%. Esto significa que se necesita tener un puntaje de por lo menos 70, en el segundo para poder visitar el primero.

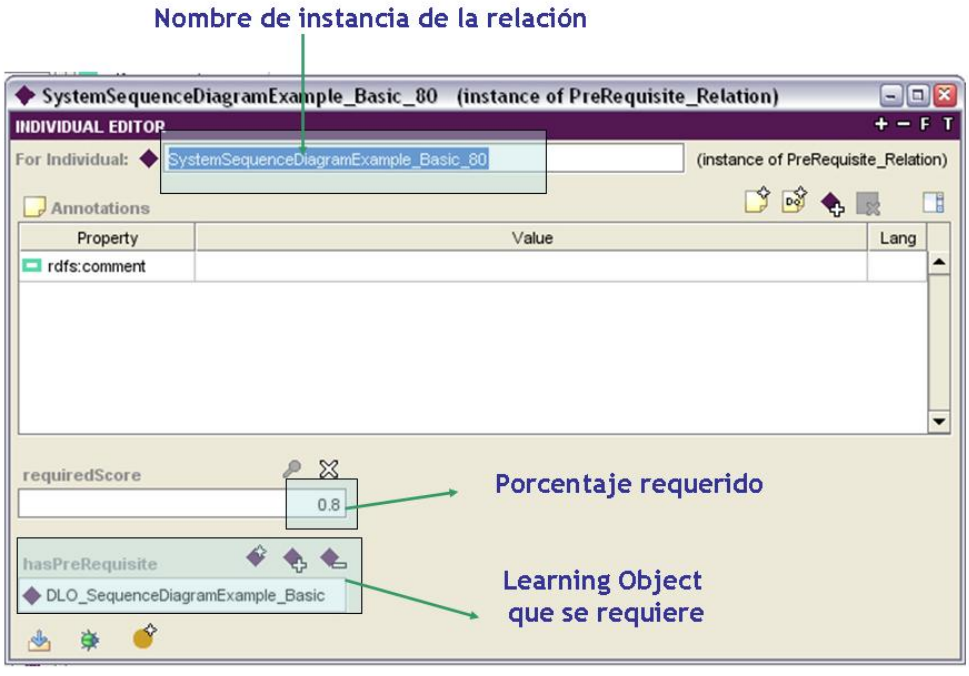


Figura 13: Instancia de relación de pre-requisitos entre learning objects

Creación de instancias de contenidos de Learning Objects

Un learning object puede tener varios contenidos, que son la unidad mínima de conocimiento que tiene la ontología. Así, por ejemplo, un learning object puede tener un texto y un gráfico que explican un contenido determinado. Entonces tendrá dos contenidos de diferentes tipos. Más abajo se describen los tipos de contenido posibles.

En la definición de los contenidos de los learning objects, es importante también definir un *número de orden*. El número de orden servirá para poner en pantalla los diferentes contenidos, en el orden establecido por el experto, ya que no es lo mismo mostrarlos en cualquier orden.

Los contenidos de los learning objects pueden tener una *referencia bibliográfica* de donde fueron obtenidos, por lo tanto se pueden instanciar dichas referencias y vincularlas al contenido.

Los contenidos de un learning object pueden ser de diferentes tipos[2]:

- ◊ Texto (Text)
- ◊ Gráfico (Graphic)
- ◊ Película Demo (DemoMovie)
- ◊ Película interactiva (InteractiveMovie)
- ◊ Escena de película (MovieScene)

Como ejemplo se mostrará la definición de un contenido de tipo texto y un contenido de tipo gráfico. El resto de los contenidos se define de forma similar con diferentes atributos o slots.

En el caso de los textos, se provee el contenido a enseñar dentro de la ontología.

La siguiente figura muestra la definición de un contenido de texto:

[2] Se mencionan también en inglés debido a que la ontología está definida en ese idioma

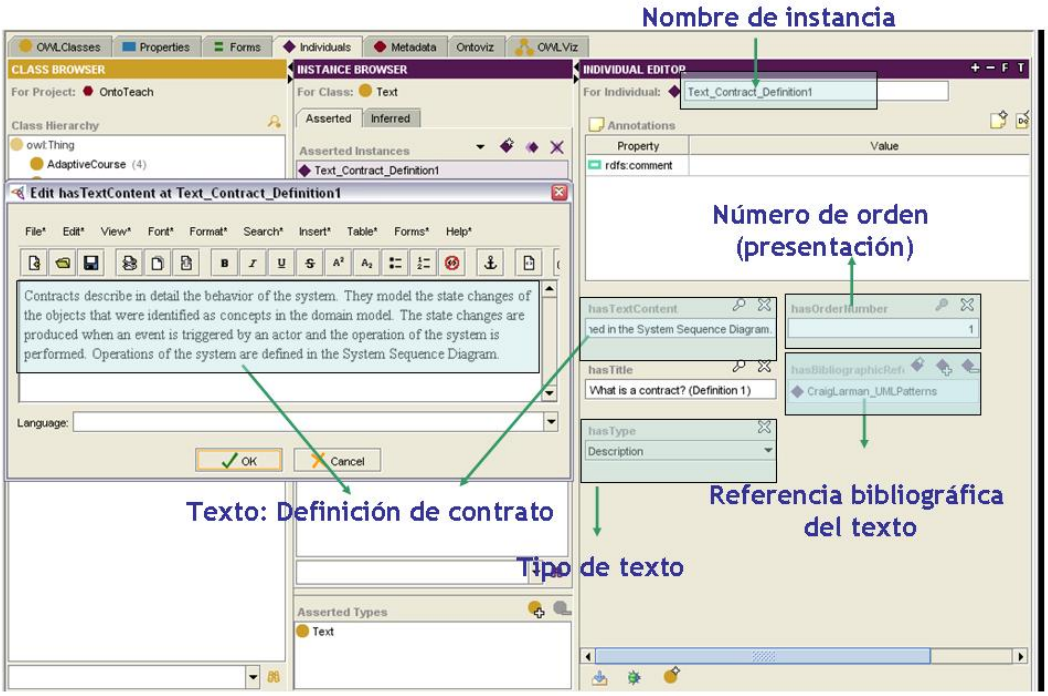


Figura 14: creación de una instancia de contenido tipo Texto

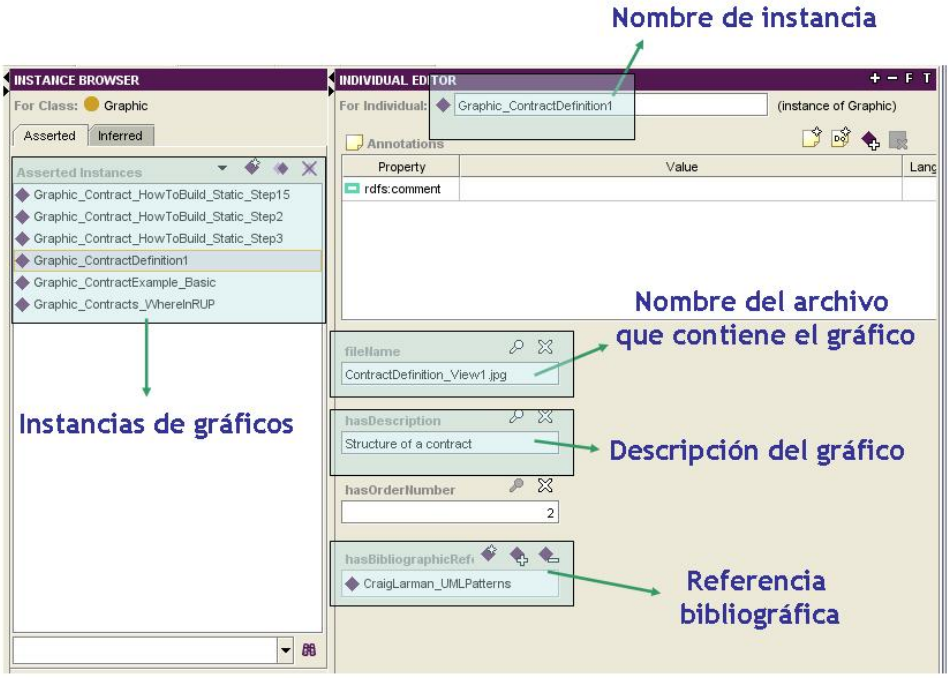


Figura 15: creación de una instancia de contenido de tipo Gráfico

Creación de instancias de contenidos de preguntas y respuestas.

Para que se pueda ofrecer al alumno diferentes learning objects, se deberá tener alguna manera de evaluarlo, de manera que se pueda ir actualizando su estado, a través de los puntajes que va obteniendo. La siguiente figura muestra la forma en que se definen preguntas para un learning object.

Cada pregunta tiene un nombre, un texto que contiene la pregunta en sí misma, y además tiene vínculos a posibles respuestas. Las respuestas también deben ser instanciadas para ser vinculadas a las preguntas.

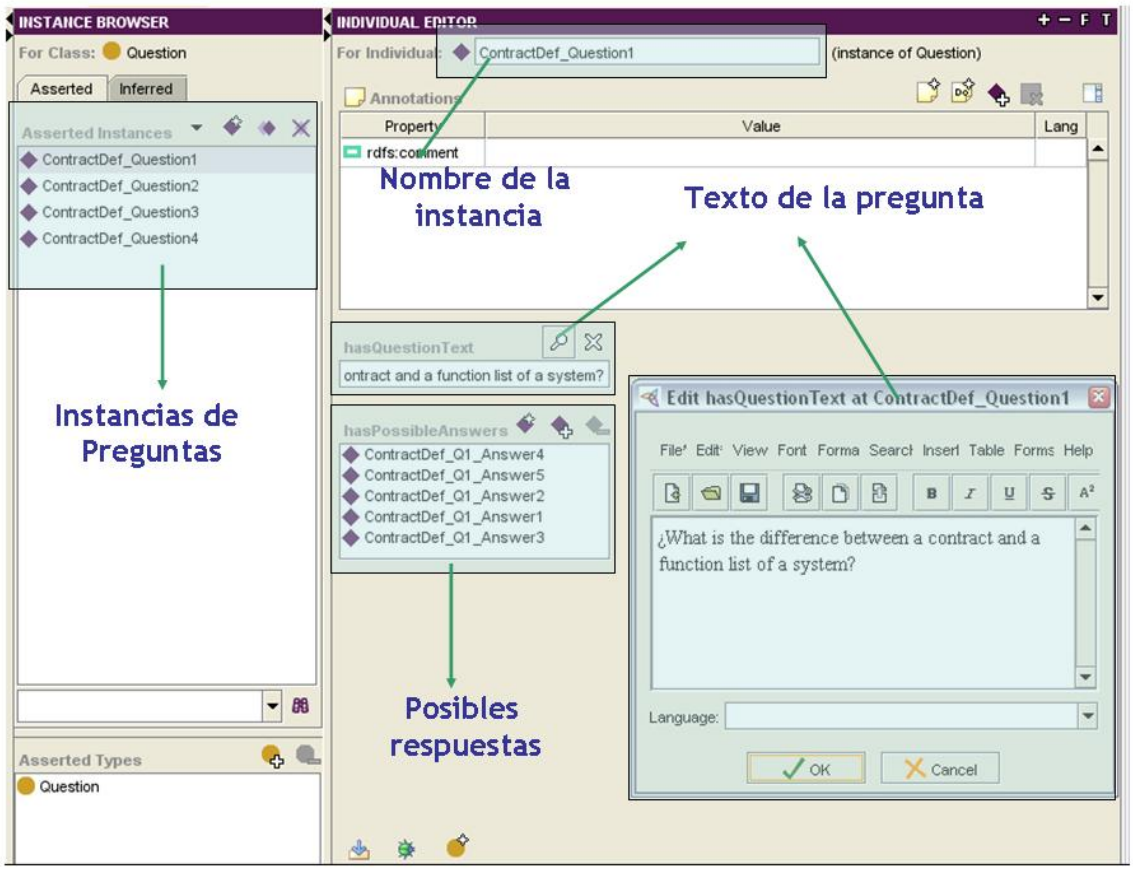


Figura 16: Definición de una instancia de Pregunta (Question)

En la siguiente figura se muestra cómo se instancia una respuesta. Una respuesta tiene un nombre de instancia y un texto. Pero además se requiere definir si esa respuesta es correcta o incorrecta.

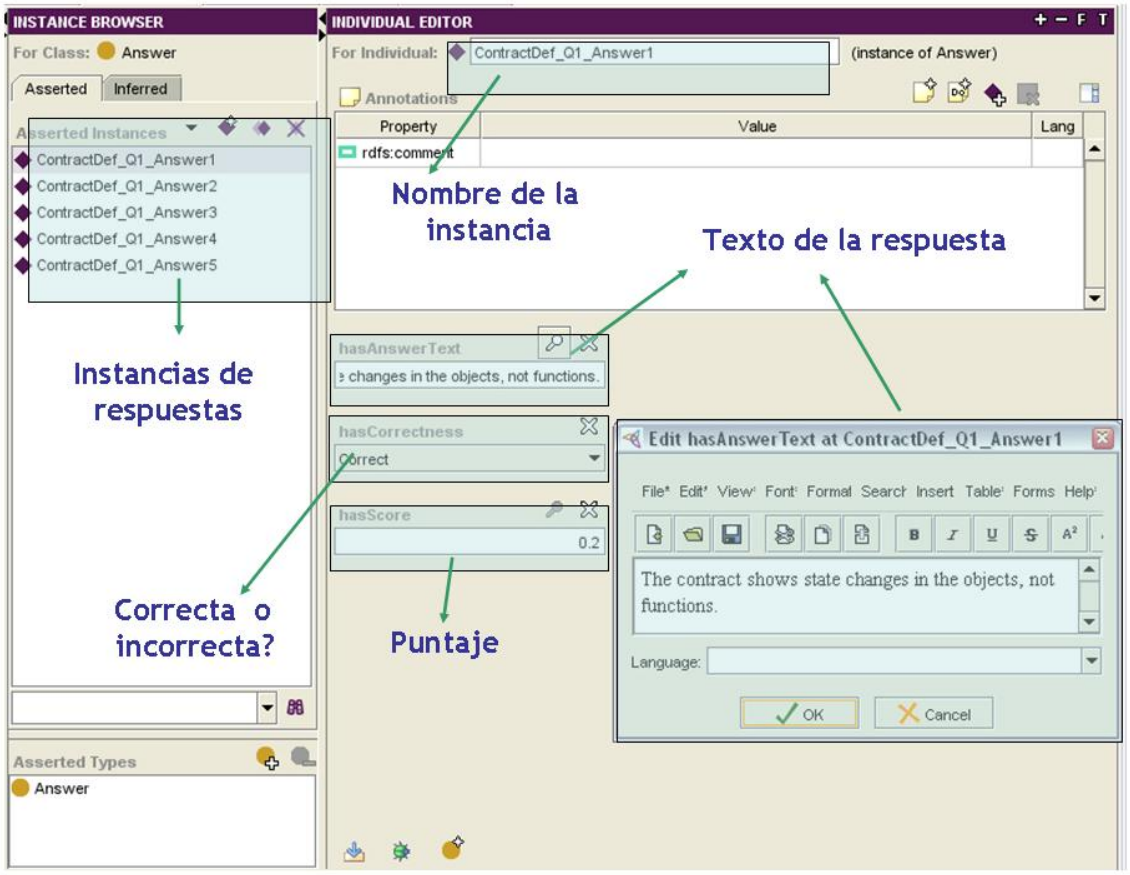


Figura 17: Definición de una instancia de Respuesta (Answer)

5.3. Configuración de CBR

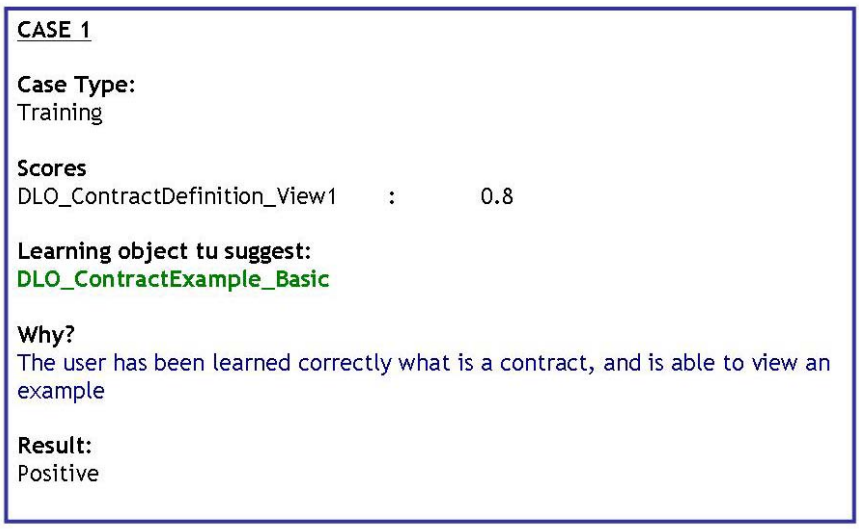
5.3.1. Definición de los casos de entrenamiento

La definición de los casos de entrenamiento es una tarea clave para el correcto funcionamiento del sistema. La base de casos representa la experiencia del docente, por lo tanto, tener una base de casos con casos incorrectos, podría asemejarse a tener un docente que no puede identificar las necesidades de los alumnos, ni su nivel de comprensión. Como consecuencia el docente explicaría algo que el alumno no está en condiciones de aprender.

Para crear una base de casos, el entrenador del sistema debe conocer sobre el dominio a enseñar, pero también debe conocer cómo enseñar ese dominio.

Como se ha mencionado en el capítulo anterior, un caso representa una experiencia particular, donde dada una serie de conocimientos obtenidos por un alumno, se le debería ofrecer determinado learning object para que el proceso de aprendizaje sea el más adecuado para ese alumno.

Un caso ejemplo de AOOTS es el siguiente:


CASE 1

Case Type:
Training

Scores
DLO_ContractDefinition_View1 : 0.8

Learning object tu suggest:
DLO_ContractExample_Basic

Why?
The user has been learned correctly what is a contract, and is able to view an example

Result:
Positive

Ejemplo de Caso de entrenamiento para AOOTS

Para simplificar la explicación, se considera una lección a modo de ejemplo y los puntajes obtenidos en esa lección solamente. Sin embargo, se asume que el alumno tendrá también los puntajes en los learning objects de lecciones anteriores.

Suponemos en este ejemplo, solo para simplificar, que el usuario obtuvo el puntaje máximo en todos los learning objects de lecciones anteriores, por lo cual está preparado para navegar en esta lección actual sin necesidad de volver atrás. Por ello es que se

mostrarán ejemplos donde siempre se recomiendan learning objects dentro de la lección "Contratos".

Analizamos a continuación cada componente del caso mostrado anteriormente a modo de ejemplo:

- ◊ **Tipo de caso** (Case Type): En este caso particular representa que el caso fue ingresado en modo de entrenamiento (Training)
- ◊ **Puntajes obtenidos** (Scores): el usuario ha obtenido 0.8 en el learning object que describe la definición de contrato en su primera versión (Contract Definition View 1)
- ◊ **Learning object a sugerir** (Learning Object ti Suggest): a un usuario que ha comprendido la definición de contrato, se recomienda sugerirle un ejemplo básico de contrato.
- ◊ **Motivo de la sugerencia** (Why): contiene el motivo por el cual se recomienda ofrecer el learning object al usuario en primera instancia.
- ◊ **Resultado de la sugerencia** (Result): un resultado positivo indica que el learning object debe sugerirse. A diferencia de esto, un resultado negativo (no es este el caso) indicaría que debe evitarse.

5.3.2. Base de casos

A modo de ejemplo, en el anexo II, se presentan 7 casos posibles que servirán para determinar la navegación sugerida a un usuario dentro de la lección que se está tomando como ejemplo.

Cabe aclarar que una base de casos debería ser mucho más completa, pero se toman solo algunos casos para ilustrar la forma en que el sistema recomienda.

5.3.3. Ejemplo de funcionamiento del algoritmo CBR en AOOTS

Para ilustrar el funcionamiento de AdapTeach en AOOTS, se describirá a continuación un proceso de aprendizaje a modo de ejemplo, dentro de la lección "Contratos".

Imaginemos que el usuario se encuentra en el learning object que da una definición de contrato (*DLO_ContractDefinition_View1*) y que ha obtenido 0.9 de puntaje en dicho learning object. ¿Qué learning object debería ofrecésele a continuación? Esa es la pregunta que se resolverá con el razonamiento de AdapTeach explicado a continuación.

La siguiente figura (Figura 18) muestra todos los learning objects que podrían sugerirse con una flecha azul punteada. Se puede observar que las relaciones de pre-requisitos tienen un puntaje, por lo cual no se podría ingresar a un learning object si el puntaje obtenido en el learning object actual (0.9) no fuera mayor al puntaje pedido por la relación de pre-requisito.

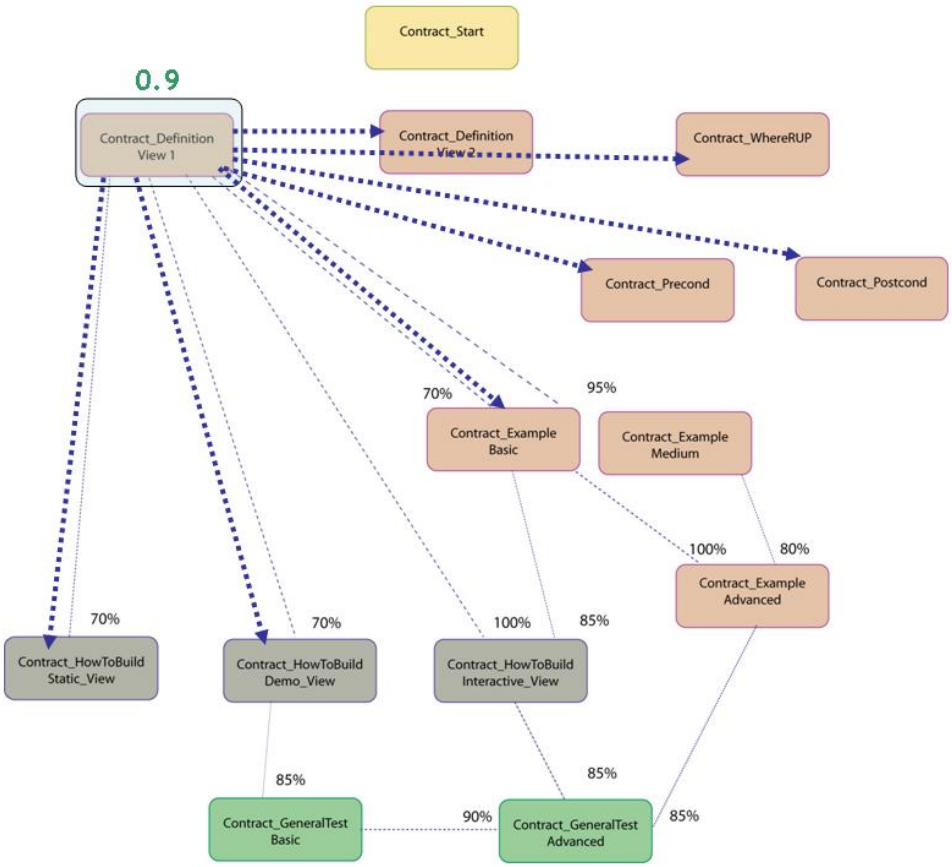


Figura 18: Posibles learning objects a sugerir desde un learning object

De todos los caminos posibles, AdapTeach debería sugerir uno de ellos en primer lugar. Para eso tiene en cuenta la experiencia, de manera que analizando el estado actual, se puede buscar una situación parecida y sugerir el learning object que se sabe que da resultado en esa situación encontrada.

El estado del usuario actual es:

DLO_ContractDefinition_View1: 0.9

Esto significa que solo consideraremos este puntaje en el cálculo de la sugerencia. Nuevamente cabe aclarar que el ejemplo está simplificado para una explicación del proceso, ya que en teoría, el usuario debería tener además todos los puntajes de lecciones anteriores visitadas.

Para obtener la sugerencia, AdapTeach evalúa todos los casos de la base de casos (ver anexo II con los casos ejemplos definidos), y obtiene todos los casos que sean similares a la situación actual. Los casos similares son aquellos son similares en un porcentaje umbral definido. Supongamos que en este caso el umbral que definimos es del 80%, lo

que significa que todos los casos que no sean similares en un 80% por lo menos, serán descartados.

El cálculo se realiza comparando la nueva situación con cada caso de la base, en este caso tenemos 7 comparaciones. A continuación se muestra una tabla con la información de los casos, y más abajo otra tabla con el cálculo de las similitudes de la situación nueva con cada uno de ellos.

Caso	DLO_Contract Definition_View1	DLO_Contract Definition_View2	DLO_COntract Example_Basic	DLO_Contract Postconditions	…	Learning Object	Resultado
	Importancia: 1.0	Importancia: 0.8	Importancia: 1.0	Importancia: 0.8	…		
1	0.8	0	0	0		DLO_Conrtract Example_Medium	Positivo
2	0.4	0	0	0		DLO_Conrtract Definition_View2	Positivo
3	0.8	0.8	0.5			DLO_Contract Postconditions	Positivo
4	0.8	0.8	0.5	0.9		DLO_Contract Example_Advanced	Positivo
5	0.8	0.8	1	0.9		PLO_Contract HowToBuild_Demo	Positivo
6	0.8	0.8	0.5	0.3		DLO_Contract Example_Advanced	Negativo
7	0.6	0	0	0		DLO_Contract Example_Basic	Positivo

Base de casos ejemplo (reducida)

Caso	Cálculo de similitud	Resultado
1	(Sim (0.8;0.9) * 1.0 + Sim (0;0.1) * 0.8 + Sim (0;0)* 1.0 + Sim (0;0) *0.8) / 3.6	0.95
2	(Sim (0.4;0.9) * 1.0 + Sim (0;0.1) * 0.8 + Sim (0;0)* 1.0 + Sim (0;0) *0.8) / 3.6	0.84
3	(Sim (0.8;0.9) * 1.0 + Sim (0.8;0.1) * 0.8 + Sim (0.5;0)* 1.0 + Sim (0;0) *0.8) / 3.6	0.68
4	(Sim (0.8;0.9) * 1.0 + Sim (0.8;0.1) * 0.8 + Sim (0.5;0)* 1.0 + Sim (0.9;0) *0.8) / 3.6	0.48
5	(Sim (0.8;0.9) * 1.0 + Sim (0.8;0.1) * 0.8 + Sim (1.0;0)* 1.0 + Sim (0.9;0) *0.8) / 3.6	0.34
6	(Sim (0.8;0.9) * 1.0 + Sim (0.8;0.1) * 0.8 + Sim (0.5;0)* 1.0 + Sim (0.3;0) *0.8) / 3.6	0.61
7	(Sim (0.6;0.9) * 1.0 + Sim (0;0.1) * 0.8 + Sim (0;0)* 1.0 + Sim (0;0) *0.8) / 3.6	0.89

Cálculo de similitud de la situación nueva con cada caso de la base

El caso que está marcado en rojo indica que tiene un resultado negativo. En este caso particular no influye ya que la similitud no supera el umbral requerido.

Si observamos los resultados de similitud y suponemos un umbral del 80% vemos que hay 3 casos que superan el umbral:

Caso 1: 95% similitud → "DLO_ContractExample_Medium".
Caso 7: 89% similitud → "DLO_ContractExample_Basic".
Caso 2: 84% similitud → "DLO_ContractDefinition_View2".

Ninguno de los tres casos tiene un resultado negativo. Si así fuera deberíamos evitar mostrar el learning object del caso. Como los tres son positivos, podemos verificar si existen relaciones de pre-requisitos entre el learning object actual y los posibles a sugerir:

- ◊ Entre *DLO_ContractDefinition_View1* y *DLO_ContractExample_Medium* hay una relación de pre-requisito del 95%. El puntaje del usuario es de 0.9, por lo cual *NO* es posible mostrar este Learning object.
- ◊ Entre *DLO_ContractDefinition_View1* y *DLO_ContractExample_Basic* hay una relación del 70%, por lo cual también se puede mostrar.
- ◊ Entre *DLO_ContractDefinition_View1* y *DLO_ContractDefinition_View2* no hay pre-requisito.

Eliminamos de la lista el que no cumple con la relación de pre-requisitos:
Caso 1: *DLO_ContractExample_medium.*

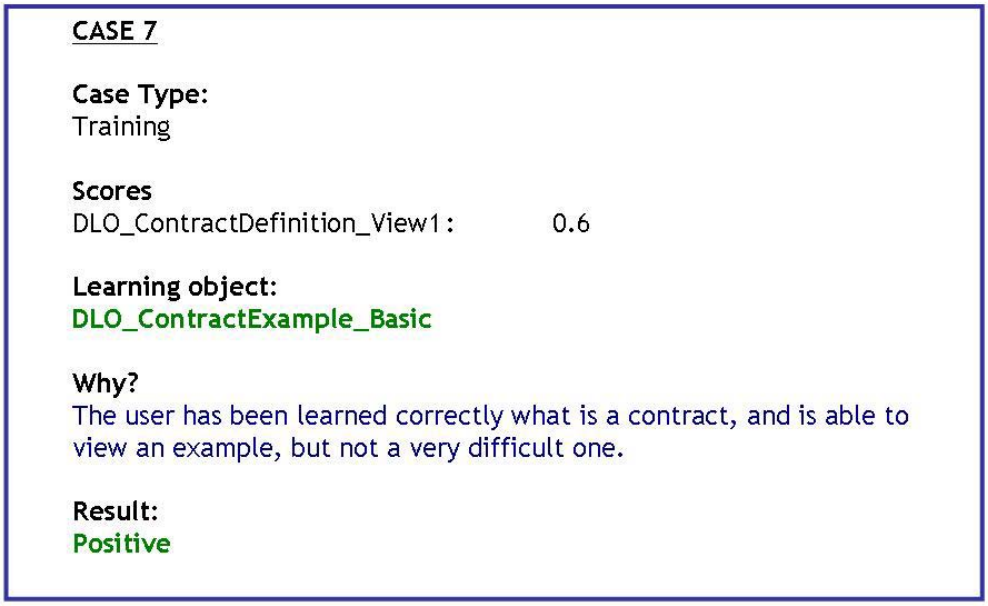
De esta manera solo nos quedan dos posibles learning objects a sugerir:

Caso 7: DLO_ContractExample_Basic
Caso 2: DLO_ContractDefinition_View2

De los dos, AdapTeach sugiere el primero de la lista por tener mayor similitud. Si quisiéramos ver el motivo pedagógico por el cual se debería sugerir ese learning object (ingresado por el entrenador), podemos verlo en el caso 7 en la sección *Why*, que en español sería:

"El usuario ha aprendido correctamente qué es un contrato, y está en condiciones de ver un ejemplo, pero no demasiado complejo"

Para referencia se muestra a continuación el caso seleccionado:



Caso más similar: Caso 7

Como hemos podido observar en este capítulo, AdapTeach permite sugerir el próximo paso a seguir en el aprendizaje teniendo en cuenta la experiencia anterior, tanto positiva como negativa. Un requisito fundamental es que el sistema sea entrenado correctamente. Si no se tuviera suficiente experiencia para entrenar el sistema, es necesario realizar ajustes teniendo en cuenta toda la información que almacena AdapTeach cada vez que un usuario accede o sale de un learning object. Esta información se describió en el capítulo anterior, como parte del aprendizaje de AdapTeach.

6. Implementación de AdapTeach: prueba de concepto

AdapTeach ha sido definido conceptualmente como framework. Algunos componentes de Adapteach se han implementados para comprobar la posibilidad de implementación de la propuesta. La implementación completa, que incluye una interfaz gráfica amigable y que permite mostrar los contenidos de los learning objects instanciados en la ontología, está en desarrollo y se menciona en la sección de trabajos futuros en el capítulo 8.

6.1. Componentes implementados para la prueba de concepto

Los componentes implementados a modo de prueba de concepto para el desarrollo de este trabajo son:

- ◊ **Ontología Ontoteach**: implementada en el software Protégé (creado por la Universidad de Stanford), con el plugin para creación de ontologías en OWL (Ontology Web Language). La ontología está implementada en forma completa, pero está instanciada solo para la prueba de concepto, habiendo definido solo algunos learning objects de un curso adaptativo (AOOTS).
- ◊ **Controlador de Ontología**: permite interpretar el código OWL a través de un framework llamado JENA, también creado por la Universidad de Stanford. JENA provee las facilidades para manipular ontologías con objetos en lenguaje Java. La implementación para la prueba de concepto solo incluye la lectura de la ontología, la construcción del modelo en Java, y la lectura de algunos contenidos de los learning objects instanciados.
- ◊ **Base de casos**: implementada con formato XML. Para bases de casos muy grandes, se sugiere la implementación con alguna tecnología de bases de datos que permita indexar los casos y buscar con más rapidez para encontrar similitudes.
- ◊ **Perfil del usuario**: implementado con formato XML, definiendo un archivo XML para cada usuario. Solo se ha implementado la información sobre puntajes obtenidos, aunque en el perfil deberían también incluirse las preferencias e historia de navegación (no tenidas en cuenta en la prueba de concepto).
- ◊ **Razonador (Motor CBR)**: en la prueba de concepto se ha implementado el motor CBR, es decir el algoritmo que calcula qué caso es el más parecido a la situación actual del alumno. El algoritmo implementado interpreta los archivos XML de perfil de usuario y de la base de casos, calcula las similitudes, encuentra el caso más parecido y devuelve el identificador de learning object que sugiere el caso.
- ◊ **Interfaz**: Se muestran en este trabajo algunas posibles pantallas. Para la prueba de concepto se ha utilizado el framework ZK [frameworkZK], que permite construir rápidamente pantallas en formato XML. ZK provee facilidades para la construcción de interfaces en Java que incluyan gráficos, animaciones, películas construidas en flash y videos.

Diagrama de clases de la prueba de concepto

A continuación se muestra un diagrama de clases de la prueba de concepto, donde se pueden observar los diferentes controladores mencionados anteriormente y todas las clases que colaboran con ellos. Cabe destacar que las implementaciones realizadas han sido construidas para comprobar la posibilidad de implementar el framework, pero el desarrollo completo es un trabajo que queda fuera del alcance de esta tesis, y que será desarrollado como un trabajo futuro.

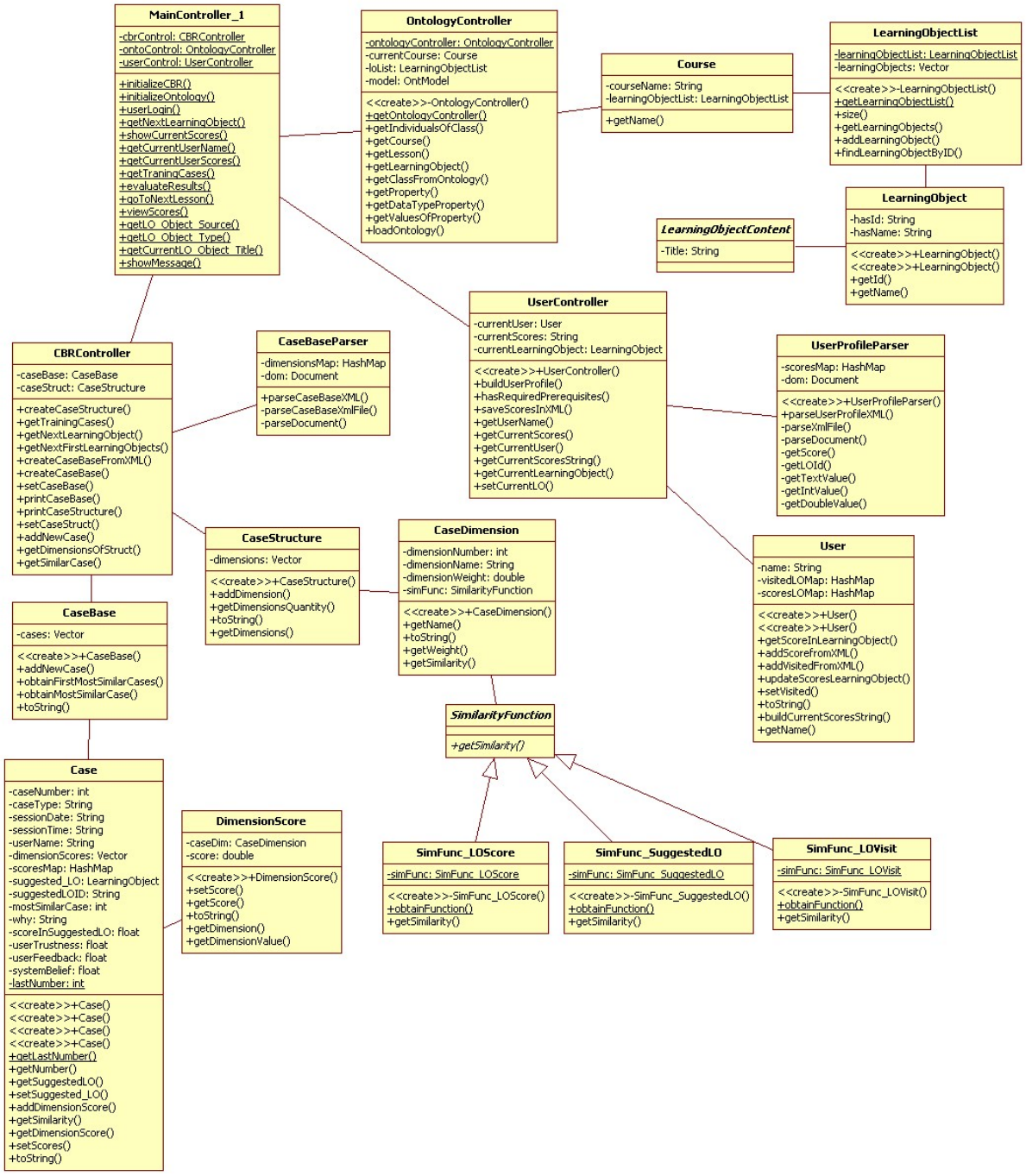


Figura 19: Diagrama de clases de la prueba de concepto

6.2. Componentes no implementados y mejoras propuestas

Como se ha mencionado antes, los componentes implementados han permitido definir conceptualmente el framework. Sin embargo, para observar el mismo en forma completa, es necesario realizar un desarrollo que incluya los componentes que se mencionan en esta sección. Este desarrollo está en marcha y se menciona en la sección de trabajos futuros.

Ontología

La ontología está implementada en forma completa, pero queda pendiente la instanciación de todo el contenido de un curso para ponerla en funcionamiento.

Interfaz gráfica

La interfaz debería:

- ◊ Mostrar el contenido de los learning objects instanciados en la ontología, en los diferentes tipos de medios que existen actualmente (imágenes, videos, animaciones, películas flash, etc.)
- ◊ Mostrar las preguntas y ejercicios de los learning objects, instanciados en la ontología
- ◊ Registrar los datos ingresados por el usuario en la evaluación de los learning objects
- ◊ Registrar el feedback del usuario en lo que respecta a opinión sobre las sugerencias.
- ◊ Mostrar el estado actual del usuario: puntajes y preferencias definidas en el perfil
- ◊ Mostrar el proceso de razonamiento de la última sugerencia
- ◊ Mostrar la base de casos actual, diferenciando entre los casos de entrenamiento y aprendidos.
- ◊ Mostrar la historia de navegación del usuario

Base de casos

En un curso completo, existirá un gran número de learning objects, y por lo tanto, puede haber muchas combinaciones de situaciones que se desean incluir en la base de casos, tanto a modo de entrenamiento como casos creados en el aprendizaje.

Se propone la implementación de la base con alguna tecnología que permita la indexación de los casos para la búsqueda del más similar de forma rápida.

Razonador

Además del motor CBR para el cálculo del próximo paso a seguir, el razonador debería:

- ◊ sugerir nuevos casos a agregar en la base, en función de las experiencias obtenidas con el uso del sistema.
- ◊ evaluar las relaciones de pre-requisitos entre learning objects para evitar sugerencias restringidas por el contenido.
- ◊ evaluar restricciones generales para evitar sugerir learning objects que no pueden ser sugeridos.

Analizador

El analizador debe ser el encargado de:

- ◊ Actualizar el perfil del alumno con los nuevos puntajes obtenidos en el learning object actual
- ◊ Actualizar el perfil del alumno para registrar su historia de navegación cuando éste navega por los diferentes learning objects, indicando si ha accedido a ellos por recomendación del sistema o por decisión propia.
- ◊ Actualizar el perfil del alumno si éste modifica sus preferencias.
- ◊ Decidir si un caso aprendido por el razonador, debe ser agregado o no a la base de casos. Esto es realizado pidiendo aprobación al usuario experto.

6.3. AdapTeach Classroom: Entorno propuesto para la implementación

En esta sección se describe el entorno propuesto para la implementación de AdapTeach y se muestran algunas pantallas a modo de ejemplo. Estas pantallas constituyen solo un prototipo, dado que la prueba de concepto solo incluye la implementación del motor, para comprobar la posibilidad de implementar la propuesta. Sin embargo, estas pantallas darán una idea del entorno completo en funcionamiento.

El entorno de AdapTeach (AdapTeach classroom) tiene 4 secciones principales:

- ◊ **Contents**: muestra contenidos del learning object actual.
- ◊ **User Status**: muestra información sobre el usuario logueado.
- ◊ **Learning Objects**: muestra los learning objects del curso en el que se encuentra el usuario, en forma de lista.
- ◊ **AdapTeach Reasoning**: muestra el razonamiento de adapTeach para llegar al learning object actual.

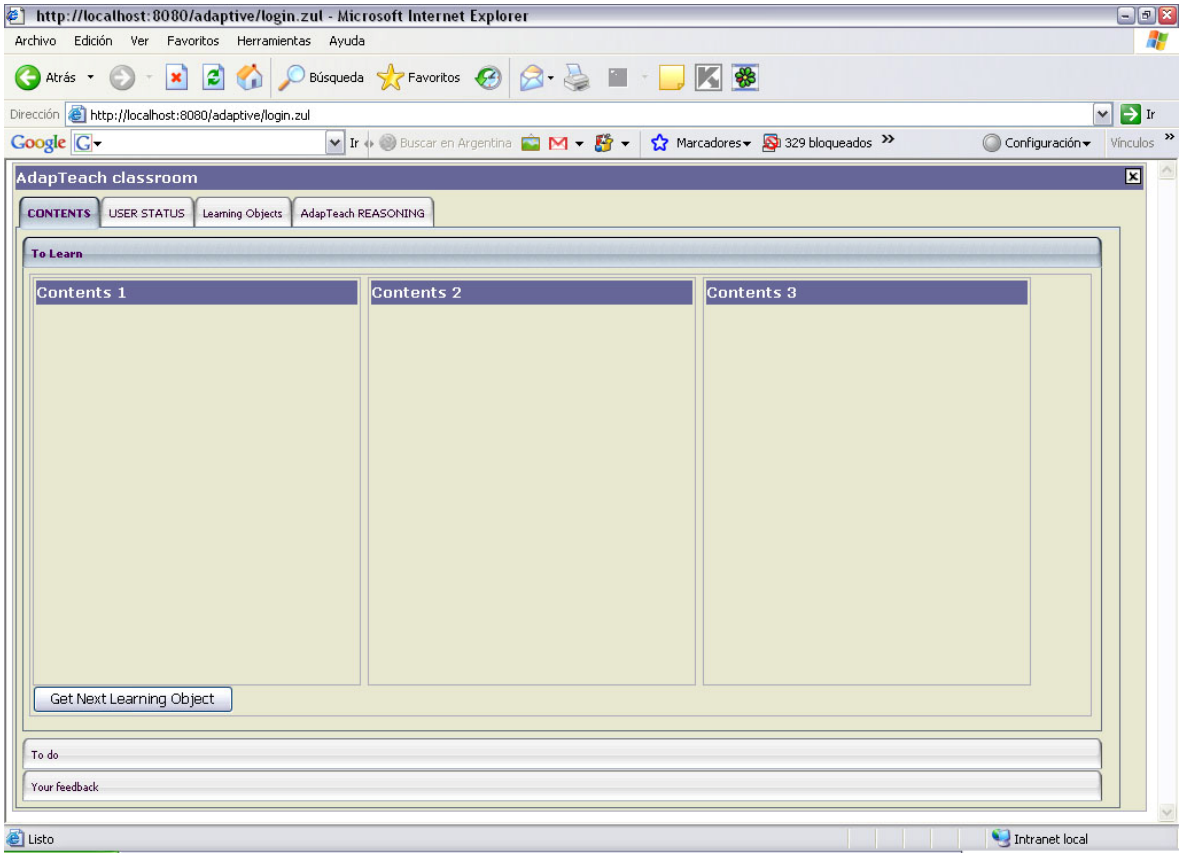


Figura 20: Entorno básico de AdapTeach: AdapTeach Classroom

6.3.1. Contents (Contenidos)

Este es el lugar donde se muestran los contenidos a enseñar, y las preguntas que el usuario debe responder. Tiene tres solapas:

To learn: contiene el contenido a enseñar. Es el lugar donde se muestra el learning object actual y sus contenidos. Tiene un botón que le permite al usuario acceder al siguiente learning object sugerido por Adapteach.
To do: contiene las preguntas y ejercicios a responder por el usuario.
Your feedback: en este sector el usuario puede dar su opinión sobre el learning object actual, indicando si le pareció una buena sugerencia o no.

Figura 21: Pantalla básica de contenidos de AdapTeach

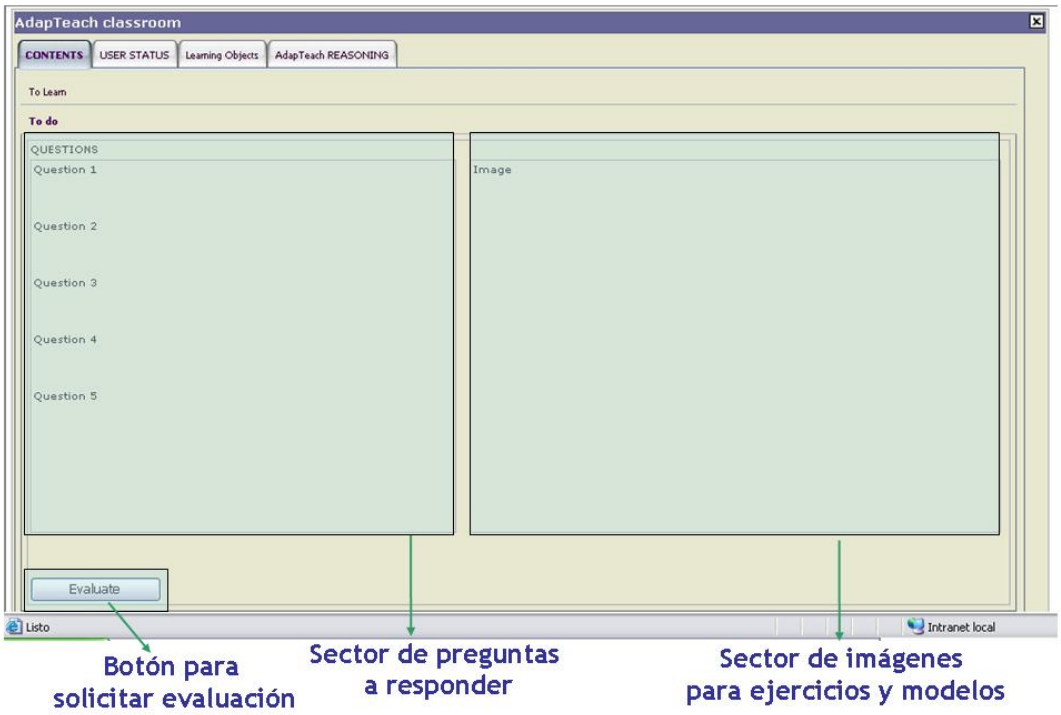


Figura 22: Pantalla básica de preguntas y ejercicios en AdapTeach

6.3.2. User Status (Estado e información del usuario)

En esta sección el sistema muestra toda la información del usuario:

- ◊ Preferencias seleccionadas.
- ◊ Historia de navegación.
- ◊ Puntajes obtenidos en los learning objects.

6.3.3. Learning Objects (Objetos de aprendizaje)

En esta sección el sistema muestra la lista de Learning Objects que componen el curso. El usuario podría ir a cualquiera de ellos a partir de aquí, evitando la sugerencia del sistema.

6.3.4. AdapTeach Reasoning (Razonamiento de AdapTeach)

En esta sección se muestra el razonamiento que ha hecho AdapTeach para llegar a sugerir el learning object actual, o bien el último que ha sugerido en caso de que al actual el usuario no haya llegado por sugerencia del sistema.

6.4. Tecnologías y herramientas utilizadas

Editor de Ontologías Protégé

Como ya se ha mencionado antes, se ha utilizado el editor de ontologías Protégé, con el plugin para código OWL (Ontology Web Language).
A continuación se muestra la definición de la clase *LearningObject* en el editor Protégé.

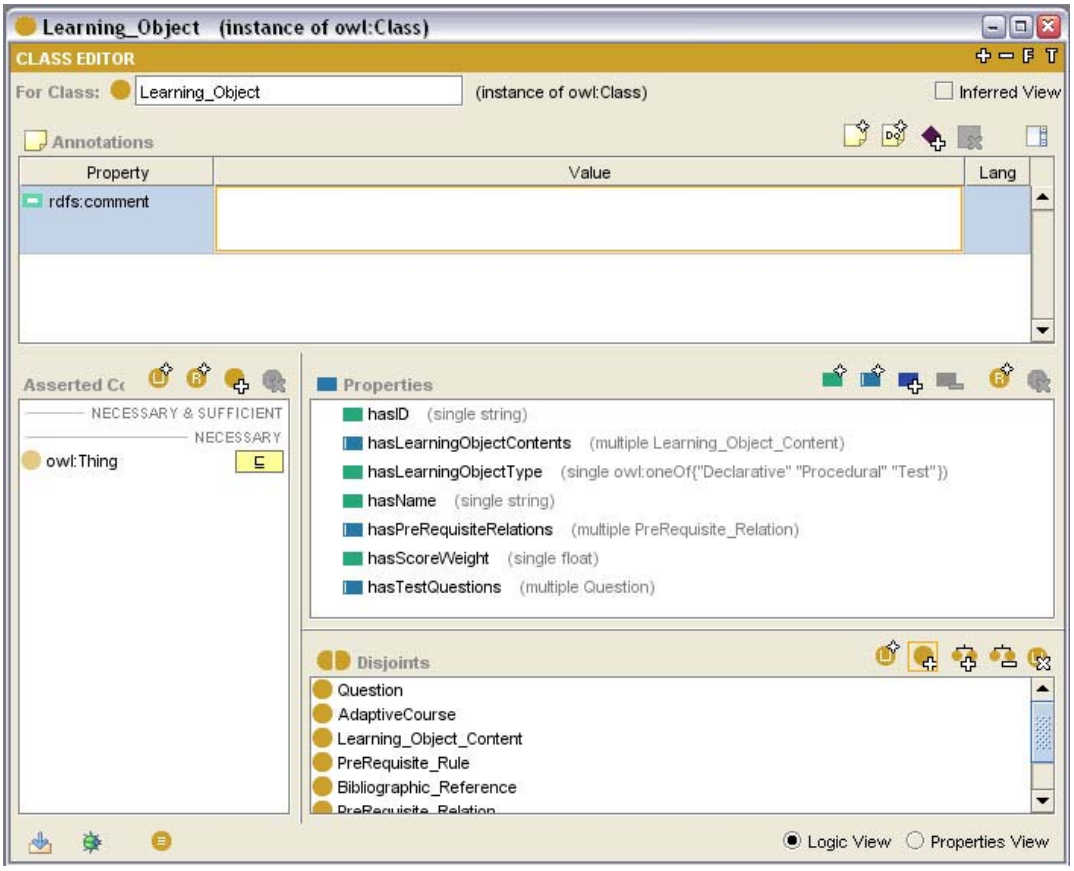


Figura 23: Pantalla de instanciación de la ontología en Protégé

A partir de la definición de la ontología construida en la herramienta Protégé, se genera el código OWL que será procesado por AdapTeach.

Framework ZK para la construcción de interfaces Java

La interfaz de usuario está construida utilizando un framework llamado ZK, que permite definir archivos XML para construir las interfaces. Estos archivos XML tienen tags para la definición de las ventanas, botones, y todos los elementos gráficos de la interfaz. El framework interpreta el XML y construye la interfaz gráfica que se observa al ejecutar la aplicación.

A continuación se muestra, a modo de ejemplo, una porción de código XML, guardado con la extensión .zul, que corresponde a la pantalla de login de usuarios.

```
<?xml version="1.0" encoding="UTF-8"?>
<?xml-stylesheet href="ptp.css" type="text/css"?>
<!--
LOGIN
-->
<window id="login" title="Welcome to AdapTeach" border="normal" width="90%" closable="true" use="MainController">
 <hbox  width="99%">
                        <vbox>
                                        <image id="image" src="/images/AdapTeachTitle.jpg" width="800px"/>
                                        <separator spacing="50px"/>
                        </vbox>
 </hbox>
                        <hbox>
                        Select a course:
                        <listbox mold="select" rows="1">
                                        <listitem label="Object Oriented Analysis and Design" selected="true"/>
                                        <listitem label="Design Patterns"/>
                                        <listitem label="Advanced Genetics" />
                                        <listitem label="How to drive a car"/>
                        </listbox>
                        <separator spacing="40px"/>
                        </hbox>

<hbox  width="60%">
                        <grid>
                        <columns>
                        <column width="200px"/>
                        </columns>
                        <rows>
                        <row>
                        UserName: <textbox id="userName" value=""/>
                        </row>
                        <row>
                        Password: <textbox value=""/>
                        </row>

                        </rows>
                        </grid>
                        <separator spacing="30px"/>
 </hbox>
                         <hbox>
                        <separator spacing="30px"/>
                        <button id="btnLogin" label="Enter" width="125px" onClick="userLogin();"/>
                          </hbox>
                        <zscript>
                          void userLogin(){
                        login.visible=false;
                        MainController.userLogin (userName.value);
                        final Window win = (Window) Executions.createComponents("principal.zul",null,null);
                        win.doEmbedded();
                          }
                        </zscript>
</window>
```

Figura 24: Definición XML de la pantalla de Login con ZK (login.zul)

La siguiente pantalla es la que se puede visualizar una vez interpretada por el framework ZK:

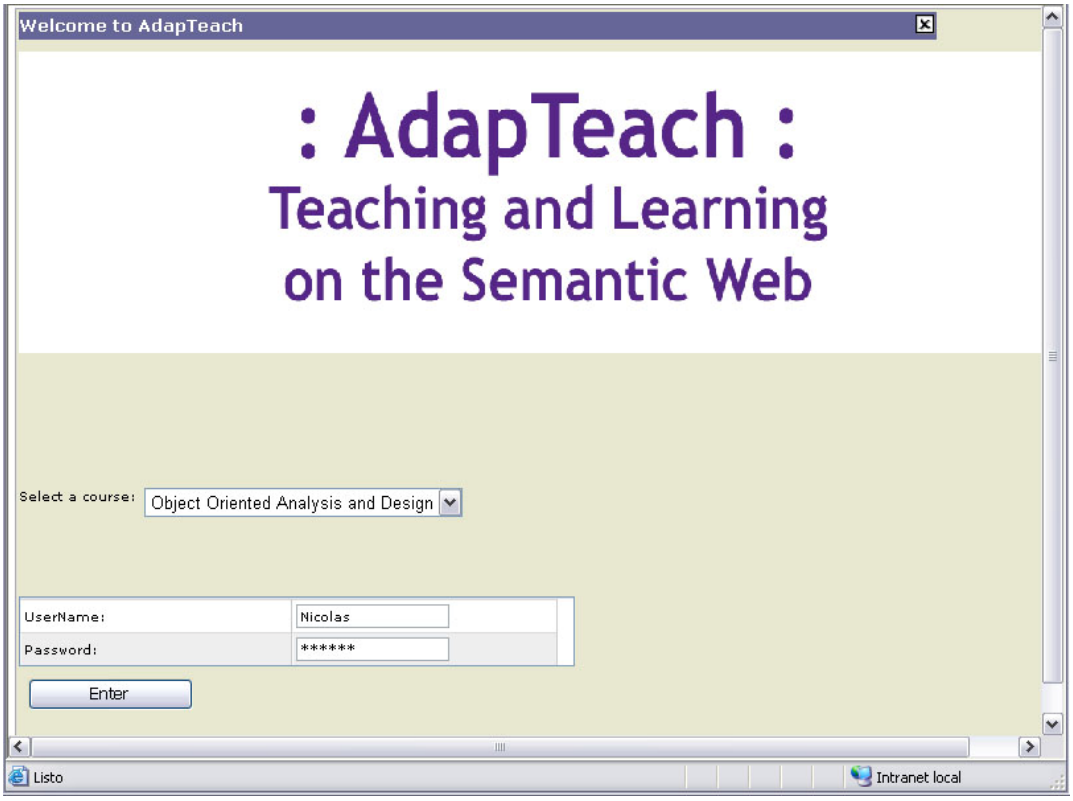


Figura 25: Pantalla básica de Login de usuarios en AdapTeach.

7. Trabajos relacionados

El desarrollo de la Web semántica ha impactado notablemente en el futuro del e-learning. En los últimos años se han realizado avances en estándares para el desarrollo de learning objects, como por ejemplo Learning objects Meta-Data o IMS. Además, se han creado grandes repositorios de learning objects tales como Ariadne y otros. Ha ocurrido un cambio de foco, pasando de ambientes cerrados de e-learning a ambientes abiertos, en los cuales pueden integrarse learning objects de múltiples fuentes [Henze1].

La mayoría de las técnicas inteligentes aplicadas pueden clasificarse en tres grandes grupos [Brusilovsky1]:

- ◊ secuenciamiento de currículum o contenidos.
- ◊ soporte para la resolución interactiva de problemas.
- ◊ análisis inteligente de las soluciones del alumno.

Estas tecnologías están orientadas al soporte de las tareas "inteligentes" de un docente, que no pueden ser soportadas por herramientas no inteligentes tradicionales utilizadas en sistemas de tutoría [Brusilovsky1].

A continuación se mencionan algunos trabajos que tienen relación con el presente.

Personal Learning Object Readers

Autor/es: Nicola Henze [Henze1].

Objetivo y descripción:

Este trabajo se basa en la utilización de tecnologías actuales de Web Semántica para la definición de un framework basado en servicios para la enseñanza personalizada. El objetivo de este framework es establecer la funcionalidad de personalización a través de servicios Web semánticos.

Se provee la infraestructura para mantener diferentes tipos de servicios Web: servicios para crear la interfaz de usuario (Visualization services) y servicios que ofrecen cierta funcionalidad de personalización (Personalization services). El componente principal es el servicio de conexión (Connector Service), que pasa la información y las peticiones entre la interfaz de usuario y los servicios de personalización, y además muestra la información de perfil de usuario y los metadatos de cursos, learning objects, etc. Para ver el contenido de Learning objects utiliza Learning Object Readers.

Adaptabilidad:

Los servicios de personalización utilizan técnicas de adaptabilidad como las mencionadas en [Brusilovsky].

Ejemplos de servicios de personalización: ofrecer un learning object, mostrar quizzes, ejercicios, y mostrar links a información más detallada.

Definición de contenidos:

Se definen learning objects, descripción de cursos, ontologías de dominio y perfiles de usuario usando estándares. Para definir un curso se debe:

- ◊ Crear anotaciones en el curso de acuerdo al estándar LOM (obligatorio)
- ◊ Proveer una ontología o el link a una ontología que contenga el conocimiento de dominio del curso (opcional)
- ◊ Proveer un punto de entrada al curso, desde el cual los alumnos deben acceder a él (obligatorio).

Open Authoring Toolkit

Autor/es: Declan Dagger, Owen Conlan, Vincent P. Wade [Dagger1]

Objetivo y descripción:

Define una arquitectura para facilitar el reuso de recursos de aprendizaje en cursos adaptativos. Explica el proceso de creación de un curso y el proceso de selección de los recursos candidatos en tiempo de ejecución. Presenta una herramienta para la construcción de cursos adaptativos.

Este enfoque presenta grupos candidatos que son utilizados para agrupar los recursos de aprendizaje que son similares entre sí. Por ejemplo, un grupo puede tener que ver con un determinado concepto, conteniendo recursos que cubren el concepto de diferentes maneras. Los recursos pueden variar en aspectos pedagógicos o técnicos, pueden enfocar el concepto desde diferentes perspectivas, o también pueden mostrar el material de formar diferentes según diferentes topologías de red, ancho de banda, dispositivos de usuarios finales, o diferentes habilidades del alumno (por ejemplo cuando hay limitaciones físicas).

Cada grupo candidato tiene asociados metadatos que describen el rol del grupo y los identificadores de los recursos de aprendizaje que están dentro de él.

Existen varias metodologías que se pueden aplicar para la agrupación de recursos de aprendizaje, como por ejemplo similitud en el estilo de aprendizaje, similitud en conocimiento que se debe poseer como pre-requisitos, etc. La información abstracta de cada grupo es inferida por el motor de adaptabilidad durante el proceso de selección del recurso de aprendizaje.

Los metadatos se almacenan en un repositorio de recursos de aprendizaje. Para la definición de los metadatos se utiliza MetaSCO, una forma de XML. Para la agrupación de recursos se utiliza la técnica de clustering que consiste en agrupar objetos similares (en este caso recursos de aprendizaje) dada una serie de entradas.

Para los recursos de aprendizaje se debe tener una ontología o jerarquía de ontologías disponible que contenga la información sobre los recursos de aprendizaje. Esta ontología interactúa con el repositorio de recursos. Se utiliza MetaSCO para capturar la información contextual antes de usar el recurso, se define en el momento de construcción del repositorio para tener una descripción detallada de los recursos.

Adaptabilidad:

El enfoque de multi-modelos interactúa e interpreta N modelos para producir la adaptabilidad.

La selección se realiza mediante tres modelos: modelo narrativo, selector de candidatos, y ejecución de narrativas.

ELM-ART: Intelligent Tutoring System on the WWW

Autor/es: Peter Brusilovsky, Elmar Schwarz, and Gerhard Weber [Brusilovsky1].

Objetivo y descripción:

Está diseñado para el soporte de la enseñanza de programación LISP. Está basado en ELM-PE, que es un entorno de aprendizaje inteligente que soporta programación basada en ejemplos, análisis inteligente de solución a problemas y facilidades de testeo y debugging.
ELM-ART provee tanto los contenidos como un entorno para la resolución de problemas, mediante el cual el alumno puede investigar ejemplos y resolver problemas on-line.

Adaptabilidad:

Provee adaptabilidad navegacional, de manera que el alumno puede tener muchas alternativas para navegar a diferencia de un sistema de e-learning tradicional. La desventaja es que el alumno tiene muchas probabilidades de perderse en el espacio de navegación. Para la adaptabilidad el sistema provee *adaptive annotation* y ordenamiento adaptativo de links. *Adaptive annotation* significa que el alumno utiliza elementos visuales para ver el tipo y el estado de cada link.

Se mantiene además un modelo de usuario para cada usuario registrado. El sistema sabe cuando un contenido puede ser comprendido por el alumno, o cuando todavía no está listo (hay algún pre-requisito sin cumplir).

El ordenamiento de links es realizado a través de la similitud entre casos, que permite a los alumnos reusar el código desarrollado en ejemplos anteriores en nuevos problemas. El sistema ordena los posibles links evaluando la similitud con el actual.

Definición de contenidos**:**

ELM-ART está diseñado para enseñanza a distancia y provee todos los contenidos (presentación de conceptos, tests, ejemplos y problemas) en una forma hipermedial.

Limitaciones

- ◊ El usuario puede perderse en el espacio de navegación.
- ◊ El sistema está diseñado para un tipo particular de enseñanza: la programación en Lisp.

8. Conclusiones y trabajos futuros

En el ámbito de los sistemas de enseñanza hay mucho por hacer en la tarea de lograr que estos sistemas soporten el proceso de enseñanza y aprendizaje que se da en la vida real entre docentes y alumnos.

Las personas son diferentes, sus formas de pensar y de aprender también lo son. Como consecuencia, un sistema de enseñanza que se comporte siempre de la misma manera con cualquier usuario, puede verse limitado en su eficacia para lograr el aprendizaje de un alumno. Si bien no se pretende que los sistemas de enseñanza reemplacen al docente, es importante que los mismos estén dotados de cierta inteligencia, para poder aprovechar al máximo las oportunidades de aprendizaje y lograr el objetivo de enseñar o servir de soporte en la enseñanza a diferentes tipos de alumnos.

Este trabajo ha consistido de un análisis de la estructura que debería tener un curso adaptativo y la definición de un framework o marco de trabajo para construir sistemas que se adapten a usuarios, para la enseñanza. El framework define conceptualmente un curso adaptativo. Como se ha visto en el desarrollo de este trabajo, se han definido las siguientes cuestiones:

- ◊ la identificación de diferentes tipos de adaptabilidad que podrían darse en un sistema adaptativo para la enseñanza.
- ◊ la forma de estructurar y almacenar el conocimiento a enseñar y sus restricciones.
- ◊ la forma de razonar para sugerir la secuencia de contenidos (adaptabilidad navegacional) más adecuada al usuario y su situación actual.
- ◊ la forma de representar el estado de los usuarios y su nivel de aprendizaje.
- ◊ la forma de representar la experiencia del docente que el sistema utilizará para sugerir la secuencia de contenidos.
- ◊ todo aquello que el sistema debería observar de los usuarios.
- ◊ la forma que tiene el sistema de aprender o adquirir nuevas experiencias.
- ◊ algunas limitaciones que se han detectado en el framework definido.
- ◊ las diferentes herramientas e implementaciones que serían necesarias para poner en práctica el framework.

Esta definición conceptual ha sido probada en lo que se ha llamado la prueba de concepto, con la cual se ha concluido que es posible implementar el framework para construir sistemas de enseñanza adaptativos.

La eficacia de sistemas construidos con este framework dependerá en gran medida de la forma en que el framework sea instanciado y el sistema entrenado.

La representación del conocimiento en una ontología utilizando lenguaje OWL, que puede ser procesada con herramientas tales como Jena [Jena2], permite la implementación del framework en la Web semántica. Este es un aspecto interesante, considerando que el futuro de la Web tiene que ver con la representación de la semántica de la información y su procesamiento con agentes inteligentes.

Algunas limitaciones del framework tienen que ver con la naturaleza de los sistemas de enseñanza adaptativos, tales como:

- ◊ la dificultad de determinar si un alumno realmente aprendió un conocimiento o no.
- ◊ la dificultad de definir casos de entrenamiento efectivos, plasmando la experiencia docente en estos casos.
- ◊ la dificultad de representar explícitamente el conocimiento de un dominio a enseñar.
- ◊ la dificultad de representar explícitamente la forma en que un docente enseña un dominio dado.
- ◊ la dificultad de hacer explícito y representar en forma completa el proceso de enseñanza y aprendizaje, simulando lo que naturalmente ocurre en el aula entre docentes y alumnos.

Otras limitaciones tienen que ver con la definición de un alcance definido para el desarrollo de este trabajo. Ejemplos de estas limitaciones son:

- ◊ La ontología OntoTeach está definida en forma completa en estructura, pero su instanciación con diferentes dominios de conocimiento podrían permitir el refinamiento de la estructura básica.
- ◊ La definición conceptual de los casos de entrenamiento está definida pero sería de gran valor hacer un análisis de la efectividad de los casos en el uso real del sistema. Esto podría hacerse acumulando grandes cantidades de casos de entrenamiento y aprendidos en un sistema que implemente el framework, y analizando la información recopilada para detectar posibles debilidades y fracasos en la sugerencia de la secuencia de contenidos.

Se han identificado una variedad de trabajos de interés para esta área, que de ser desarrollados, permitirían refinar el modelo conceptual creado a partir de la acumulación de experiencias que surgen del uso del framework. Algunos de los trabajos identificados ya están en desarrollo.

Un sistema adaptativo que utilice este framework y esté diseñado para registrar todo lo que se puede observar en un proceso de enseñanza por parte del docente, permitiría obtener conclusiones interesantes sobre la forma en que debería aprender este sistema.

El Framework ha sido construido para simular la interacción entre docentes y alumnos, y por tratarse de un proceso adaptativo en la realidad, mucho se puede aprender del uso por parte de diferentes personas que tengan diferentes formas de aprender y diferentes niveles de aprendizaje.

Trabajos futuros

A continuación se mencionan algunos trabajos futuros que permitirían implementar AdapTeach en forma completa para su prueba, refinamiento y uso. Algunos de estos proyectos se encuentran en desarrollo. La lista completa de trabajos futuros identificados es la siguiente:

- ◊ Construcción de una aplicación que tome el conocimiento de la ontología, implemente en forma completa el motor CBR y sugiera a un alumno con un perfil determinado, los siguientes learning objects que debería visitar. Esta aplicación tendría una interfaz amigable que permitiría mostrar el contenido a enseñar, ofrecer los ejercicios y preguntas, evaluar el desempeño de usuario y actualizar su perfil.

- ◊ Construcción de una instancia completa de OntoTeach que enseñe un dominio en particular que permita a un alumno aprender un tema, considerando todas las características mencionadas en el framework.

- ◊ Construcción de una aplicación para el uso del usuario experto o docente, que permita la creación de casos de entrenamiento en un dominio determinado y el seguimiento del aprendizaje del sistema, pudiendo agregar nuevos casos a la base de casos y sugerirlos. Esta aplicación podría evaluar el desempeño del sistema y sugerir al experto nuevos casos para agregar a la base.

- ◊ Construcción de una aplicación que permita instanciar en forma amigable la ontología. Esta aplicación permitiría instanciar OntoTeach y verificar el ingreso de todos los contenidos necesarios para el funcionamiento del sistema adaptativo (pre-requisitos, importancias, etc.).

- ◊ Construcción de una aplicación para visualizar el estado de un alumno en forma gráfica. Esta aplicación debería mostrar de la mejor manera posible en qué estado del aprendizaje dentro de un curso se encuentra el alumno, utilizando herramientas visuales.

- ◊ Construcción de una aplicación de análisis de la base de casos y las historias de navegación para determinar casos que no representan una buena experiencia.

- ◊ Definición de la forma de tratar otros tipos de adaptabilidad, tales como adaptabilidad de contenido, de preferencias de usuario y de presentación, dentro del framework Adapteach.

Bibliografía

[Balani] *Naveen Balani.* The future of thye web is semantic. Oct 2005. http://www-128.ibm.com/developerworks/xml/library/wa-semweb/

[Brusilosvsky1] *Brusilovsky, Peter; Elmar Schwarz, and Gerhard Weber*. ELM-ART: An intelligent tutoring system on the WWW. Department of Psychology, University of Trier. 1996.

[Brusilovsky2] *Brusilovsky, Peter*. Knowledge Tree: A distributed architecture for adaptive e-learning. School of Information sciences. Pittsburg University.New York 2004.

[Brusilovsy3] *Brusilovsky, Peter***.** Methods and Technics for adaptive hypermedia. 1996

[Dagger1] *Declan Dagger, Owen Conlan, Vincent P. Wade*. An Architecture for Candidacy in Adaptive eLearning Systems to Facilitate the Reuse of Learning Resources. Trinity College Dublin. Republic of Ireland.

[Henze1] *Henze, Nicola*. E-Learning in the Semantic Web: The Personal Learning Object Raeders. University of Hannover.

[Jena 2] *Hewlett Packard Laboratories.* Sitio web de Jena. Jena 2: A semantic Web Framework. http://jena.sourceforge.net/

[Larman] *Larman, Criag*. UML y Patrones. Segunda Edición. Prentice Hall. 2002.

[Knublauch et al] *Holger Knublauch, Ray W. Fergerson, Natalya F. Noy and Mark A. Musen.* The Protégé OWL Plugin:An Open Development Environment for Semantic Web Applications. Stanford Medical Informatics, Stanford School of Medicine.

[Nodenot et al.] *Thierry Nodenot, Christophe Marquesuzaa, Pierre Laforcade, Christian Sallaberry*. Model based Engineering of Learning Situations for Adaptive Web Based Educational Systems.

[Protégé] Sitio Web del software Protégé. Universidad de Stanford. http://protege.stanford.edu/

[Wiley] *David A. Wiley***.** Connecting learning objects to instructional design theory: A definition, a metaphor, and a taxonomy. Utah State University. Digital Learning Environments Research Group. The Edumetrics Institute.

Anexos

Anexo I: Descripción de las dimensiones de un caso CBR para AdapTeach

A continuación se describen las diferentes dimensiones del caso.

Tipo de caso

Tag XML: CaseType

Qué contienen
Esta dimensión indica el tipo de caso: de entrenamiento (training) o aprendido (learned).

Importancia

En principio, los casos aprendidos tienen la misma importancia que los casos de entrenamiento. Sin embargo, la experiencia en el uso de AdapTeach permitirá definir si esto debe ser así.

De donde se obtienen los valores

Si el caso es de entrenamiento, el caso tendrá el valor "Training". Si el caso es aprendido, al crear el caso se le asignará el valor "Learned".

Puntajes en los LO en una situación particular

Tag XML: LO_Score

Qué contienen

Estas dimensiones indican el puntaje que el usuario (particular o genérico, según el tipo de caso) en cada uno de los learning objects.

Importancia

Estas dimensiones tienen una importancia relativa a la importancia del contenido. Por ejemplo, si es una definición o algo clave en el aprendizaje del tema, seguramente tendrá una importancia alta. Si es un LO procedural modo demo o interactivo, también tendrán importancia alta porque son los que contienen el conocimiento más importante.
En cambio, LO que contienen información adicional, no tan relevante o que no son tan importantes, tendrán una importancia menor

La importancia de las dimensiones de contenido es definida por el entrenador del sistema, cuando configura el framework para una aplicación particular.

De donde se obtienen los valores

Cuando el caso es de entrenamiento, los valores son ingresados por un entrenador que simula a un usuario en particular en un estado del aprendizaje.

Cuando el caso es aprendido, estas dimensiones corresponden al estado actual del usuario que está usando el sistema.

Learning Object sugerido

Tag XML: Suggested LO

Qué contiene
Esta dimensión indica el LO que ha sido sugerido en ese caso particular. En los casos de entrenamiento es el nodo que el entrenador considera que debería sugerirse ante la combinación de valores del caso.

Importancia
No tiene importancia asociada, porque es el resultado del caso.

De donde se obtienen los valores
Cuando el caso es de entrenamiento, esta dimensión es llenada por el entrenador. Cuando es un caso agregado en modo de uso, esta dimensión contiene el LO al que el usuario accedió como una sugerencia del sistema.

Motivo de la sugerencia

Tag XML: Why

Qué contiene
Indica el motivo por el cual debe sugerirse el LO indicado en la dimensión LO sugerido.

Importancia
No tiene importancia asociada, porque es informativo

De donde se obtienen los valores
Esta dimensión es explicativa y debe ser llenada por el experto en los casos de entrenamiento.
En los casos aprendidos se copia el contenido de esta dimensión que está presente en el LO más similar.

Caso más similar

Tag XML: MostSimilarCase

Qué contiene

Cuando se crea un nuevo caso con los puntajes de un usuario y se lo compara con todos demás casos, se obtiene el más similar y se registra en el nuevo caso. Más adelante se explicará cómo se comparan los casos.

El valor que contiene es el ID del learning object más similar.

Importancia

Esta dimensión es simplemente informativa, no se usa para el cálculo del próximo paso a seguir, pero es importante a la hora de analizar qué tan buenos son los casos aprendidos por el sistema, si se deseara ajustar este aprendizaje (no en modo uso, sino en modo administración del sistema).

De donde se obtiene el valor

Se toma el Identificador del learning object más parecido, identificado por el algoritmo

Resultado

Tag XML: Result

Qué contiene

Esta dimensión indica si la experiencia representada en el caso ha sido positiva negativa. Una experiencia positiva indica que la sugerencia del learning object ante la situación del alumno es buena, o sea que es recomendable aplicar el learning object. Una experiencia negativa indica que debe evitarse sugerir el learning object en la situación planteada.

Importancia

La importancia de esta dimensión siempre es alta, porque es lo que el sistema ha obtenido como conclusión sobre el resultado de una sugerencia.

Anexo II: Base de casos ejemplo de AOOTS (reducida)

Los siguientes son algunos ejemplos de casos de la base de casos de AOOTS.

CASE 1

Case Type:
Training

Scores
DLO_ContractDefinition_View1 : 0.8

Learning object:
DLO_ContractExample_Medium

Why?
The user has been learned correctly what is a contract, and is able to view an example

Result:
Positive

CASE 2

Case Type:
Training

Scores
LO_ContractDefinition_View1 0.4

Learning object:
DLO_ContractDefinition_View2

Why?
Why? The user has not learned correctly what is a contract, another way to explain is necessary

Result:
Positive

CASE 3

Case Type:
Training

Scores

DLO_ContractDefinition_View1	0.8
DLO_ContractDefinition_View2	0.8
LO_Contract_Example_Basic	0.5

Learning object:
DLO_Contract_PostConditions

Why?
The user has learned correctly what is a contract, but he has not understood the example, so it is necessary to add detailed information. It can be done here with the pre and postconditions LO. Postconditions are best to be learned first.

Result:
Positive

CASE 4

Case Type:
Training

Scores

DLO_ContractDefinition_View1	0.8
DLO_ContractDefinition_View2	0.8
DLO_Contract_Example_Basic	0.5
DLO_Contract_PostConditions	0.9

Learning object :
DLO_Contract_Example_Advanced

Why?
The user has learned correctly what is a contract, He did'n understood the example, but he did understood postconditions. The current LO is Postconditions so he may be able to understand with another example.

Result:
Positive

CASE 5

Case Type:
Training

Scores

DLO_ContractDefinition_View1	0.8
DLO_ContractDefinition_View2	0.8
DLO_Contract_Example_Basic	1
DLO_Contract_PostConditions	0.9

Learning object:
PLO_Contract_HowToBuild_DemoView

Why?
The user has learned correctly what is a contract and its example. He is able to understand a more difficult example or to know how to build a contract in its three modes. It's best to see the demo first, so this view is suggested

Result:
Positive

CASE 6

Case Type:
Learned

Scores

DLO_ContractDefinition_View1:	0.8
DLO_ContractDefinition_View2:	0.8
DLO_Contract_Example_Basic:	0.5
DLO_Contract_PostConditions:	0.3

Learning object :
DLO_Contract_Example_Advanced

Why?
The user has learned correctly what is a contract, He did'n understood the example, but he did understood postconditions. The current LO is Postconditions so he may be able to understand with another example.

Result:
Negative

CASE 7

Case Type:
Training

Scores
DLO_ContractDefinition_View1 : 0.6

Learning object:
DLO_ContractExample_Basic

Why?
The user has been learned correctly what is a contract, and is able to view an example, but not a very difficult one.

Result:
Positive