UNA ARQUITECTURA PARA UNA APLICACIÓN INTELIGENTE DE EVALUACIÓN BASADA EN PATRONES DE DISEÑO, COMPONENTES Y AGENTES BAJO EL PARADIGMA DE WBE

R. Peredo Valderrama, I. Peredo Valderrama
Escuela Superior de Cómputo, IPN, México, D.F.
Av. Juan de Dios Bátiz S/N Esq. Miguel Othón de Mendizabal
Col. Linda Vista, 07738, México, D.F.
rperedo@ipn.mx, peredo@cic.ipn.mx

RESUMEN

La presente propuesta propone un enfoque novedoso hacia una arquitectura de evaluación inteligente para aplicaciones bajo el paradigma de Educación Basada en *Web (Web-Based Education, WBE por sus siglas en inglés)*. La arquitectura de evaluación de la aplicación *Web* está basada en: patrones de diseño de software, componentes, y agentes. Teniendo una capa de base de datos y una capa de base de conocimientos, para personalización dinámica basada en los planes: reconfiguración, composición y retroalimentación. Se hace uso de varios patrones de diseño para obtener una arquitectura robusta que nos permita manejar mejor el cambio y las actualizaciones. Se hace una propuesta de modificación a la arquitectura IEEE 1484 – LTSA, con la finalidad de serializar métricas de los estudiantes, colectadas a través del sistema, posibilitándonos transformar las métricas colectadas en conocimiento útil para estudiantes y profesores bajo el modelo de WBE. En la Vista del patrón MVC se uso el patrón de Composición, y por medio de componentes multimedia que denominamos: Componentes de Aprendizaje Reusables Inteligentes Orientados a Objetos (*Intelligent Reusable Learning Components Object Oriented*, IRLCOO por sus siglas en inglés), constituyendo una parte fundamental de la Aplicación Rica en Internet (*Rich Internet Application*, RIA por sus siglas en inglés).

Palabras claves*:* Patrones de diseño, Componentes, Sistema Multi-Agente y WBE.

RECONOCIMIENTOS

Los autores de este artículo agradecen al Instituto Politécnico Nacional (IPN) y a la Escuela Superior de Cómputo por su apoyo para este trabajo dentro del proyecto SIP: 20121741. Los autores desean reconocer a todos sus colegas y estudiantes que participaron en el diseño y desarrollo del software, y materiales de aprendizaje descritos en este artículo.

1. INTRODUCCIÓN

Las personas necesitan aprender a lo largo de su vida para adquirir nuevas habilidades, los diplomas de las escuelas ya no garantizan encontrar un empleo en estos días. La economía tradicional se ha transformado a una basada en el conocimiento, transformando una necesidad por trabajadores para manufactura a una por trabajadores generadores de conocimiento, estos últimos trabajadores requieren de una constante capacitación en periodos relativos cortos de tiempo, para adaptar sus habilidades a los cambios constantes, y además deben de estar preparados para cambiar varias veces de trabajo a lo largo de su vida. El aprendizaje a lo largo de la vida implica educación a lo largo de la vida, lo cual requiere profesores, recursos de aprendizaje, y un tiempo determinado para enfocarse al estudio. No siendo la educación tradicional la más adecuada para esta tarea.

La tecnología por si sola no puede revolucionar la educación, se requiere de un cambio en la sociedad, un mejor acceso al conocimiento, una capacitación constante de los profesores, una restructuración de las organizaciones educativas para adecuarlas a nuestros tiempos, y sistemas asistenciales basados en agentes que apoyen la revolución educativa.

Las computadoras han producido cambios en prácticamente todo, transformando entre otras: industria, cultura, y comunidades. Produciendo cambios vertiginosos en la mayoría de las disciplinas, revolucionando áreas como: ciencia, comunicación, economía, y comercio. Las Tecnologías de la Información (*Information Technology*, IT por sus siglas en inglés), sobresaliendo principalmente áreas como: software, hardware y redes, han generado un nuevo desafío en la educación. Siendo la educación un campo muy fértil para el uso de las IT, dentro de la cual los factores claves son: conocimiento, gente a educar, y los trabajadores del conocimiento.

El impacto de las IT en la educación producen efectos en muchas áreas. Los educadores pueden mejorar y redefinir los procesos de enseñanza/aprendizaje, al utilizar inteligencia artificial, ciencias cognitivas y aprovechar el poder completo de la Internet/*Web*. La reducción de los costos del hardware han resultan en un incremento de su uso en todas las disciplinas académicas [1].

Los educadores están incorporando herramientas en los salones de clase para estimular la curiosidad, y apoyar más a los estudiantes en su entendimiento. Los estudiantes en todos los niveles han respondido a las simulaciones computacionales, que hacen a los conceptos más atractivos y menos abstractos [2]. Los estudiantes que usan la tecnología obtienen un razonamiento más realzado, una mejor compresión y habilidades [3]. Las computadoras mejoran las actitudes de los estudiantes y sus intereses, por medio de un aprendizaje personalizable en un ambiente más interactivo y disfrutable [4].

La relación de la Inteligencia Artificial (*Artificial intelligence*, AI por sus siglas en inglés) y la educación comenzó en los 70's por varios líderes como: John Self, Jaime Carbonell, y William Clancey (1979). Esta tecnología tiene el potencial de producir materiales educacionales altamente personalizados, pedagógicos y accesibles, estos materiales pueden adecuarse mejor a las necesidades individuales de los estudiantes, e involucrar más a los estudiantes en un aprendizaje más efectivo. Esto debido a que tales sistemas son más sensibles a las necesidades individuales, pudiendo registrar las habilidades y resultados de los estudiantes de forma automática

y personalizada, además de poder ofrecerles diferentes estilos de aprendizaje.

El paradigma WBE tiene dos ventajas fundamentales respecto a la educación tradicional: los materiales educativos se pueden personalizar a las necesidades de los estudiantes en tiempo de ejecución, y los estudiantes pueden avanzar a su propio ritmo. Las áreas de investigación principales del paradigma WBE, son en la parte de desarrollo de materiales educativos: reusabilidad, accesibilidad, durabilidad, e interoperabilidad. Otra área de investigación importante son los ambientes virtuales. Hay tres iniciativas principales a nivel internacional bajo este paradigma: Consorcio de Aprendizaje Global (*Global Learning Consortium*, GLC por sus siglas en inglés), Aprendizaje Distribuido Avanzado (*Advanced Distribuited Learning*, ADL por sus siglas en inglés), e Iniciativa de Conocimiento Abierta (*Open Initiative Knowledge*, OKI por sus siglas en inglés) [5-7]. Nuestra propuesta se basa en la iniciativa ADL.

La evaluación automática ha sido un tema controversial, principalmente en modelos educativos totalmente a distancia. Muchas personas alegan que los estudiantes pueden hacer trampa, de diferentes maneras, pero esta preocupación aplica hasta en los modelos totalmente presenciales.

La evaluación en línea ha probado ser muy útil, donde la evaluación es relativamente simple: falso/verdadero, opción múltiple, y selección de área. Cuando el estudiante tiene que escribir una respuesta a ser evaluada, el sistema de evaluación debe de anticipar todas las respuestas posibles, y calificar las opciones correctas.

La evaluación en línea provee una forma rápida, precisa y consistente de llevar a cabo exámenes relativamente simples. Las herramientas de evaluación relativamente simples son frecuentemente construidas dentro de los Sistemas Manejadores de Aprendizaje (*Learning Management System*, LMS por sus siglas en inglés), pero hay herramientas especializadas de terceros, tales como QMark [8], que se concentran únicamente en las evaluaciones, ofreciendo evaluaciones más elaboradas.

El Ambiente de Aprendizaje Dinámico Orientado a Objeto Modular (*Modular Object Oriented Dynamic Learning Environment*, MOODLE por sus siglas en inglés) [9] es un LMS muy popular, para la gestión de cursos gratuito. MOODLE es un software basado en *Web* que posibilita a los instructores, entrenadores, y educadores crear cursos basados en *Web*. Provee un sistema robusto y organizado, con una interfaz fácil de usar basada en *Web*. Una de las principales ventajas de MOODLE es que los desarrolladores han mantenido la interfaz y la consistencia a lo largo de los años.

El módulo de evaluación de MOODLE provee una herramienta para crear evaluaciones con retroalimentaciones. Los instructores pueden crear evaluaciones y asignarlas a sus estudiantes, reduciendo la elevada complejidad de elaboración de exámenes para los materiales educativos. La evaluación es importante en entrenamiento y educación, para medir los resultados de los estudiantes, y proporcionar una retroalimentación personalizada a los estudiantes, en función de sus resultados obtenidos.

Esta propuesta presenta una arquitectura para una aplicación inteligente de evaluación bajo el paradigma WBE, la implementación se desarrollo usando el lenguaje de programación Java, patrones de diseño de software, componentes de software y agentes de software. La propuesta hace uso componentes de software, obteniendo los beneficios inherentes a estos: reducción de la complejidad, maximización del reuso y manejo del cambio. Los componentes multimedia avanzados denominados IRLCOO [10], consumen Servicios *Web* (*Web Service*, WS por sus siglas en inglés), implementados con el lenguaje de programación ActionScript 3.0 [11], para conformar una biblioteca de componentes para una plataforma RIA avanzada. Un Sistema Multi-Agente (Multi-Agent System, MAS por sus siglas en inglés) para la toma de decisiones automáticas del sistema, además de manejar la retroalimentación hacia los usuarios, implementado con los *frameworks: Java Agent DEvelopment* (JADE) [12], JADE eXtension (JADEX) [13], y Webbridge [13]. Las aportaciones que ofrece el sistema son: reducción de la complejidad de elaboración de las evaluaciones, maximización del reuso de los evaluaciones interactivas, mantenibilidad del sistema, simplificación de la interfaz de manejo de los componentes de evaluación, y subsistema asistencial inteligente basado en un MAS para retroalimentación dinámica. Resultando en una arquitectura robusta para manejar mejor los cambios y modificaciones, por medio de los patrones de diseño.

2. PATRONES DE DISEÑO

2.1 Introducción a los patrones de diseño

Los cambios a lo largo del desarrollo de un proyecto de software son inevitables. Los cambios en los requerimientos durante el desarrollo de un proyecto de software son constantes, requiriéndose efectuar cambios que el diseño original no contemplaba. La mejor manera para manejar el cambio en nuestros proyectos de software es por medio de un diseño robusto, que nos permita manejar cambios y modificaciones. La mejor manera de manejar los requerimientos cambiantes es por medio del manejo de las dependencias entre segmentos de código.

Los patrones de diseño nos dan una perspectiva mejor para el análisis y diseño. Además los patrones de diseño nos posibilitan mejores soluciones para maximizar el reuso de las partes de nuestros proyectos de software. La mayoría de los patrones hacen que el software se pueda modificar de una forma más sencilla, y nos posibilita un mejor mantenimiento del código a lo largo de la vida del proyecto. Esto debido a que son soluciones ya probadas y funcionales, que han ido mejorando a lo largo del tiempo en mejores arquitecturas, que manejan mejor el cambio de una manera más sencilla.

Los patrones de diseño son soluciones generales reusables para un problema común dentro de un contexto dado, siendo moldes para resolver problemas, que pueden ser usados en diferentes situaciones. Los patrones de diseño encapsulan conocimiento de experimentados desarrolladores de software, de tal manera que permiten aplicar ese conocimiento a problemas similares.

2.2 Patrón Singleton

El patrón Singleton se enfoca en asegurar que únicamente haya una única instancia de una clase en memoria, y se tenga acceso global al objeto. Teniendo como objetivo fundamental limitar a una única instancia de un objeto, y proveer un punto de acceso global. En la Figura 2 en la sección de patrón MVC, entre la Vista y el Modelo se encuentra el

patrón Singleton, en nuestro caso es implementado por medio de un Java Bean implementando la interface Serializable, y usando la etiqueta <use:Bean> en las Páginas Servidor Java (*Java Server Pages*, JSP por sus siglas en inglés), esto garantiza una sola instancia del Java Bean para el manejo del modelo.

2.3 Patrón Método de fábrica

Los acoplamientos fuertes entre clases ocasionan que los cambios a una clase requieran cambios en la otra. La situación se multiplica si los clientes de una clase requieren de múltiples cambios de código en diferentes secciones del código. Una solución es reducir la dependencia entre los clientes y las clases concretas. Es en estos casos donde el método de fábrica ofrece un diseño robusto de solución. Al introducir un intermediario entre el cliente y las clases concretas. El intermediario se denomina clase creador, ésta nos posibilita al cliente accesar a objetos sin especificar el tipo de objeto que será creado. Esto se lleva a cabo delegando la creación a un método separado dentro del creador denominado *factory*, el propósito principal es instanciar objetos y retornarlos. Los métodos de fabrica parametrizados toman un parámetro que especifican el tipo de objeto que será creado. En la Figura 1 se muestra el diagrama de clases de nuestro método de fábrica, sirviéndonos para implementar la lógica de negocios del examen en los componentes IRLCOO, esta es instanciada en el componente IRLCOO del lado del cliente.

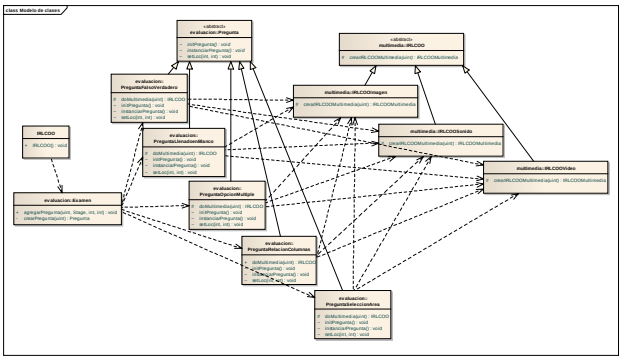

Figura 1. Diagrama de clases de nuestro método de fábrica.

2.4 Patrón Observador

Hay aplicaciones donde una fuente de información necesita difundirse a diferentes receptores, siendo la fuente de información un punto central que envía información a las instancias subscriptoras. Siendo eficiente para la difusión de la información a las instancias subscriptoras, además de que la fuente central de información garantiza que cada instancia subscriptora obtenga la misma información. La característica fundamental del patrón es la captura del cambio de estado en un punto central y enviarlo a todas las instancias subscritas. En la Figura 2 en la sección de patrón MVC, se muestra el patrón Observador, en este caso el objetivo del patrón es mantener las Vistas actualizadas.

2.5 Patrón Composición

El patrón de composición nos provee de un diseño robusto para la construcción de sistemas complejos, construyendo componentes complejos en base a componentes más pequeños. Las estructuras complejas se basa en arboles jerárquicos. Los componentes pueden ser indivisibles o complejos, estos últimos contienen una colección de otros componentes. Otro punto clave de este patrón es la simplificación de la Interfaz de Programación de Aplicación (*Application Programming Interface*, API por sus siglas en inglés), permitiendo manipular de forma indistinta tanto componentes indivisibles como complejos.

Los IRLCOO usan el patrón de composición, usando un componente IRLCOO contenedor, que tiene 2 subcomponentes IRLCOO: contenido/evaluación y navegación. Estos subcomponentes utilizan a su vez componentes IRLCOO indivisibles para la carga de multimedios: Video, Imágenes, Sonido y Animaciones [14]. La Figura 2 muestra el patrón de composición en la Vista del patrón MVC, desplegada en el navegador *Web* del cliente, mostrando al componente IRLCOO compuesto, en la forma de una evaluación. El uso de este patrón posibilito la construcción de evaluaciones modificables en tiempo de ejecución, posibilitando la personalización de las mismas a los estudiantes por medio de composición y secuenciación dinámica.

2.6 Patrón Modelo-Vista-Controlador

El patrón Modelo-Vista-Controlador (*Model-View-Controller*, MVC por sus siglas en inglés) es un patrón compuesto de múltiples patrones trabajando de manera conjunta, para crear aplicaciones complejas. El patrón consiste de tres componentes: Modelo, Vista y Controlador. El Modelo contiene los datos de la aplicación, y la lógica de negocios para manejar el estado de la aplicación, la Vista representa la interfaz hacia el usuario y muestra el estado de la aplicación, por último el Controlador maneja la entrada del usuario y cambia el estado de la aplicación. La clave fundamental de la gran flexibilidad del patrón es directamente atribuible a la separación de los tres componentes que lo conforman, sin traslapar sus responsabilidades, permitiendo a cada componente llevarlas a cabo, pero estos componentes trabajan de forma conjunta, comunicándose entre ellos.

El patrón MVC se ha utilizado en varios de nuestros proyectos [15-18], posibilitando el desarrollo de arquitecturas reutilizables, consistentes y de fácil mantenimiento. Esto posibilita que la aplicación inteligente de evaluación pueda manejar mejor el cambio y las actualizaciones. El *framework* para la implementación fue Struts 2, que pertenece a la fundación Apache [19]. Struts 2 implementa el patrón MVC, y cuenta con una gran cantidad de librerías para desarrollar aplicaciones *Web*.

Se utilizo para la estructuración de los documentos etiquetados el Lenguaje de Marcado Extensible (*eXtensible Markup Language*, XML por sus siglas en inglés), Xerces para implementar el Modelo Objeto Documento (*Document Object Model*, DOM por sus siglas en inglés), mientras que para la persistencia en archivos XML se utilizo JDOM [20-21]. En la parte de serialización de meta datos se utilizo el Marco de Descripción de Recursos (*Resource Description Framework*, RDF por sus siglas en inglés), y por último se hizo uso del *framework* JENA para la persistencia y recuperación RDF.

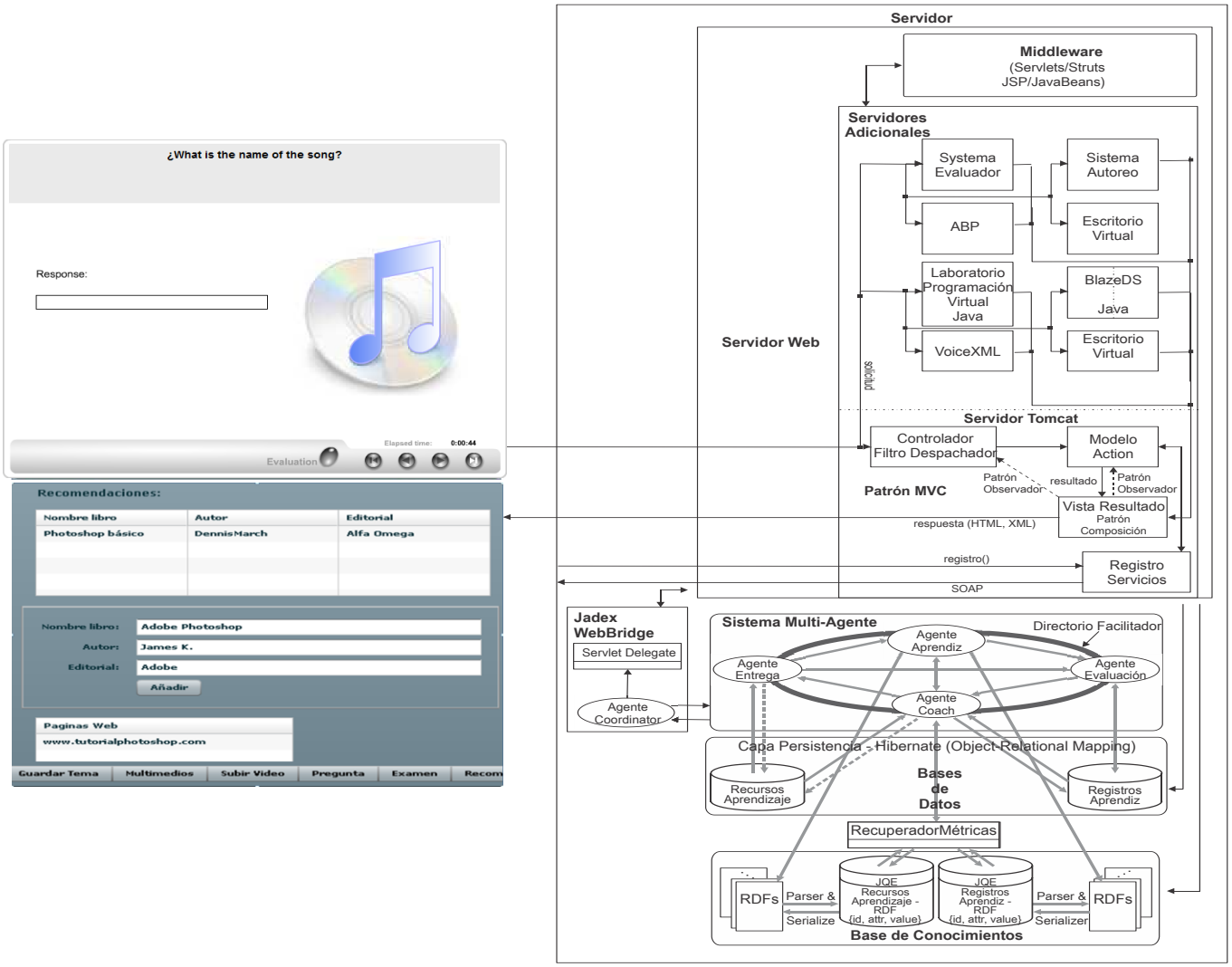

Figura 2. Arquitectura para la aplicación inteligente de evaluación.

3. ARQUITECTURA BASADA EN COMPONENTES

3.1 Introducción

El bloque fundamental de construcción para el desarrollo de software basado en componentes se denomina componente. Los componentes son bloques de construcción. Los componentes de software son vistos como una solución a la denominada crisis del software [15] que consiste en: baja reusabilidad de las partes de un proyecto, tiempos de entrega rebasados, y la calidad del software no esta garantizada. La idea fundamental es construir sistemas grandes y complejos en base a componentes más pequeños. El patrón de composición resulta ideal en el manejo de componentes de software, como ya se describió anteriormente. Los componentes tienen tres ventajas inherentes: reducción de complejidad, maximización de las reusabilidad, y manejo del cambio. Esto ha traído a los componentes a la escena principal del desarrollo de software. En varios trabajos anteriores hemos definido lo que es un componente de software [16-17, 22].

3.2 Intelligent Reusable Learning Components Object Oriented

La implementación de la última versión de los Componentes de Aprendizaje Reusables Inteligentes Orientados a Objetos (*Intelligent Reusable Learning Components Object Oriented*, IRLCOO por sus siglas en inglés) se ha implementado con el lenguaje de programación ActionScript 3.0. La API de comunicación de los IRLCOO con el LMS, en nuestro caso basado en ADL, emplea la clase *ExternalInterface* para hacer uso de la API-ADL por medio de las funciones de JavaScript [6].

3.3 Servicios *Web*

Los componentes IRLCOO consumen WS, para la implementación de los WS del sistema usamos el enfoque de la arquitectura: Transferencia de Estado Representacional (*REpresentational State Transfer*, REST por sus siglas en inglés), donde los sistemas que siguen los principios REST se denominan con frecuencia RESTful. RESTful es una arquitectura de software para sistemas hipermedia distribuidos como la *World Wide Web* (WWW). RESTful proporciona una fácil implementación y desarrollo. Para el lenguaje Java existe la Java API para Servicios *Web* RESTful (*Java API for RESTful Web Services*, JAX-RS) [23], siendo una API oficial incluida dentro del lenguaje Java, que nos proveyó de soporte para crear nuestros WS bajo la arquitectura RESTful. JAX-RS usa

anotaciones, para simplificar el desarrollo. La arquitectura RESTful hace hincapié en los recursos, una URL única para poder acceder a los recursos y a los distintos tipos de recursos. A través del ID que proporcionamos se construyen los Localizadores de Recurso Uniforme (*Uniform Resource Locator*, URL por sus siglas en inglés), desde donde se puede acceder a los diversos servicios.

4. SISTEMA MULTI-AGENTE

4.1 Introducción

Los agentes son un tema de investigación de vanguardia, y solo recientemente se ha comenzado a explotar dentro de aplicaciones prácticas. El término MAS es usado para describir a un sistema compuesto de múltiples agentes inteligentes interactuando. Los MAS están siendo usados en una amplia variedad de aplicaciones.

4.2 Arquitectura Creencias-Deseos-Intenciones

La arquitectura Creencias-Deseos-Intenciones (*Belief-Desire-Intention*, BDI por sus siglas en inglés) implementa los principales aspectos de la teoría de razonamiento práctico humano propuesto por Bratman [24], los conceptos inicialmente propuestos por Bratman fueron adoptados por Rao y Georgeff [25], para conformar un modelo más formal. El modelo ha sido adoptado y transformado dentro de una teoría formal dentro de un modelo de ejecución de agentes, basados en creencias, metas y planes [25]. Considerando los fundamentos teóricos, el número de implementaciones, y sistemas exitosos, esta arquitectura es la más ampliamente adoptada.

4.3 Middleware

El middleware JADE [12] es ampliamente usado para implementar el paradigma de agentes. Otro middleware complementario importante es JADEX, siendo un *framework* orientado a agentes usando XML y Java. [13].

El concepto de capacidad posibilita conjuntar un subconjunto de creencias, planes y metas dentro del módulo de un agente. El Directorio Facilitador (*Directory Facilitator*, DF por sus siglas en inglés) y Webbridge son capacidades en nuestra aplicación *Web*.

4.4 Marco de Descripción de Recursos

Los metadatos es información describiendo otra. La implementación de los metadatos de la arquitectura es por medio de tecnologías como: XML, esquemas XML, RDF, y esquemas RDF. Las métricas colectadas por los componentes IRLCOO del estudiante, permiten conformar la capa de base de conocimientos de nuestra aplicación, sirviendo como fuente de información de las actividades llevadas a cabo por el estudiante, y sirve para lograr una mejor personalización. La serialización RDF de las métricas del estudiante se hace por medio del *framework* JENA [26].

5. APLICACIÓN INTELIGENTE DE EVALUACIÓN

5.1 Arquitectura para la aplicación inteligente de evaluación

La arquitectura para la aplicación inteligente de evaluación toma como base la arquitectura IEEE 1484 – LTSA [14, 27-29], sustituyendo los procesos por agentes de software implementados con JADEX. También se propuso un nuevo flujo de datos, no considerado originalmente en la arquitectura original de IEEE 1484 - LTSA, consistiendo en un registro automático de métricas de los estudiantes, y depositándolas en las bases de conocimientos. Esto permite un registro de preferencias del usuario desde los componentes IRLCOO de forma automática, hacia los registros del usuario en las bases de conocimientos. Con la finalidad de registrar algunas métricas del estudiante de forma automática como: tareas completadas, trayectoria del estudiante en los materiales de evaluación, tareas incompletas, uso de herramientas de colaboración, actividades completadas, actividades incompletas, estadísticas de materiales más visitados, tiempo en cada pregunta de la evaluación, tiempo total de la evaluación, trayectoria del estudiante en los materiales de evaluación, etc. Todas estas métricas del estudiante son almacenadas en los registros del usuario de la bases de conocimientos, para su posterior análisis por medio del MAS, con la finalidad de generar una retroalimentación personalizada, identificación de posibles problemas de aprendizaje, y reconfiguración de trayectoria de aprendizaje en función de los resultados de la evaluación [30].

6. RESULTADOS

La Figura 3 en la parte superior muestra una evaluación tipo RIA de llenado en blanco, el profesor la crea en el ambiente virtual de profesor, donde tiene que seleccionar el tipo de evaluación, y configurar la pregunta y las opciones correspondientes a la misma. En la parte inferior se ve el módulo de recomendaciones a la pregunta, donde el profesor puede recomendar libros y páginas *Web* que el profesor considera adecuadas. Esta información es recuperada por medio del MAS para crear una retroalimentación más personalizada para el estudiante, en función de los planes: reconfiguración, composición y retroalimentación. El estudiante ingresa a su ambiente, y cuenta con una lista de evaluaciones desarrolladas por sus profesores, pasando a realizarlas, y obteniendo su retroalimentación personalizada.

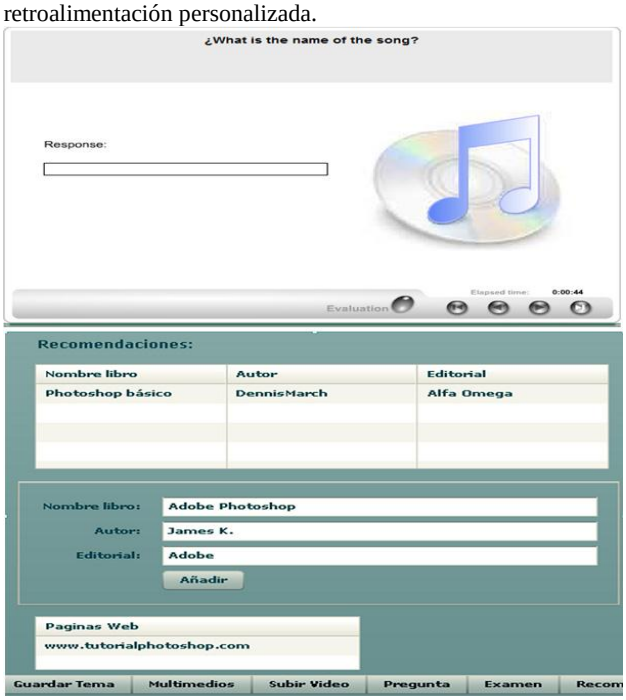

Figura 3. Componente IRLCOO de evaluación tipo RIA e interfaz de recomendaciones.

7. CONCLUSIONES

La arquitectura propuesta para la aplicación inteligente de evaluación esta basada en la arquitectura IEEE 1484 – LTSA, se hizo una modificación a la arquitectura original, agregando un nuevo flujo de datos, posibilitando la captura de métricas del estudiante sin pasar por el agente del Coach, capturando las acciones del estudiante en el ambiente virtual de aprendizaje por medio de los componentes IRLCOO, además de agregar una nueva capa de base de conocimientos. El uso de los patrones de diseño en la aplicación, posibilitando un mejor manejo del cambio y las actualizaciones de la aplicación *Web*. El patrón Singleton se utilizo para manejar el acceso a la base de datos, por medio de una sola instancia de un Java Bean. El patrón método de fábrica nos posibilito implementar la lógica de negocios del examen del lado del cliente, permitiéndonos desacoplar la instanciación de las representaciones de los componentes IRLCOO en memoria, permitiendo aprovechar mejor los recursos. El patrón Observador nos posibilito mantener actualizadas las Vistas del patrón MVC en la aplicación *Web*. El patrón composición nos posibilito crear componentes IRLCOO complejos en base a componentes más simples, además de simplificar la API de programación de los componentes IRLCOO. Además de posibilitarnos la retroalimentación, reconfiguración de los componentes IRLCOO y sus secuencias de manera dinámica, en tiempo de ejecución, en función de los resultados de la evaluación, basándose en los planes: reconfiguración, composición y retroalimentación, además de posibilitarnos separar contenido y navegación, maximizando la reusabilidad de los componentes IRLCOO. El patrón MVC nos permitió maximizar las partes del proyecto, y mejorar el mantenimiento del proyecto a lo largo del tiempo, simplificando su mantenimiento.

RECONOCIMIENTOS

Los autores de este artículo agradecen al Instituto Politécnico Nacional (IPN) y a la Escuela Superior de Cómputo por su apoyo para este trabajo dentro del proyecto SIP: 20121741. Los autores desean reconocer a todos sus colegas y estudiantes que participaron en el diseño y desarrollo del software, y materiales de aprendizaje descritos en este artículo.

8. REFERENCIAS

1. Marlino, M.R., Sumner, T.R., Wright M.J., Report of a workshop sponsored by the National Science Foundation (University Corporation for Atmospheric Research 2004)
2. Manduca, C.A., Mogk, D.W., Using Data in Undergraduate Science Classrooms (Carleton College 2002)
3. Roschelle, J., Pea, R., Hoadley, C., Gordin, D.N., Means, B., Changing How and What Children Learn in School with Computer-Based Technologies (The Future of Children 10 2000)
4. Valdez , G. , M , M. , Foertsch , M. , Anderson , M. , Hawkes , M. and Raack , L., Computer-Based Technology and Learning: Evolving Uses and Expectations (North Central Regional Laboratory 2000)
5. Global IMS Learning Consortium, URL: http://www.imsproject.org/
6. Advanced Distributed Learning Initiative, URL: http://www.adlnet.org
7. Open Knowledge Initiative, MIT, URL: http://web.mit.edu/oki/
8. Questionmark...getting results, QMark, URL: http://www.questionmark.com/esp/index.aspx
9. Moodle.org: open-source community-based tools for learning, MOODLE, URL: http://moodle.org/
10. R. Peredo Valderrama, Leandro Balladares Ocaña, L. Sheremetov, Development of Intelligent Reusable Learning Objects for *Web*-Based Education systems, Expert Systems with Applications, 28(2): 273-283, Pergamon Press, 2005
11. ActionScript 3.0 Language and Components Reference, Adobe, URL: http://livedocs.adobe.com/flash/9.0/ActionScriptLangRefV3/
12. JADE Agent Platform, Available at: http://jade.tilab.com/
13. JADEX Agent Platform, Available at: http://jadex-agents.informatik.uni-hamburg.de/xwiki/bin/view/About/Overview
14. R. Peredo Valderrama, A. Canales Cruz, I. Peredo Valderrama: Un primer enfoque hacia una arquitectura para sistemas educativos basada en tecnologías de *Web* semántica para educación basada en *Web*. CISCI 2011.
15. Rubén Peredo Valderrama, Alejandro Canales Cruz: Aplicaciones *Web* basadas en componentes de software para Educación Basada en *Web*. CNCIIC-ANIEI 2008.
16. R. Peredo Valderrama, I. Peredo Valderrama, L. Balladares Ocaña, Sistema evaluador usando *Web* semántica para educación basada en *Web*, 5ta. Conferencia Iberoamericana en Sistemas, Cibernética e Informática (CISCI 2006), IIIS, 2006
17. R. Peredo Valderrama, I. Peredo Valderrama, L. Balladares Ocaña, Sistema generador de contenidos multimedia interactivos didácticos usando componentes de software para educación basada en *Web*, 6ta. Conferencia Iberoamericana en Sistemas, Cibernética e Informática (CISCI 2007), IIIS, 2007
18. Rubén Peredo, Alejandro Canales, Alain Menchaca, Iván Peredo, Intelligent *Web*-based education system for adaptive learning, Expert Systems with Applications, 38(12): 14690–14702, Pergamon Press, 2011
19. Struts 2 (2010), URL: http://struts.apache.org/2.1.6/index.html.
20. DOM specification, URL: http://www.w3.org/DOM/
21. JDOM, URL: http://www.jdom.org/
22. C.Szyperski, Component Software -Beyond Object-Oriented Programming (Addison-Wesley 1999)
23. The Java Community Process(SM) Program - JSRs: Java, JAX-RS, URL: http://jcp.org/en/jsr/detail?id=311
24. M. Bratman. Intention, Plans, and Practical Reason. Harvard University Press. Cambridge, MA, USA. 1987.
25. A. Rao and M. Georgeff. BDI Agents: from theory to practice. V. Lesser. Proceedings of the First International Conference on Multi-Agent Systems (ICMAS'95). The MIT Press. Cambridge, MA, USA. 1995. pp.312-319.
26. JENA Semantic *Web Framework*, Available at: http://jena.sourceforge.net/
27. IEEE Learning Technology Standards Committee. Available at: http://ltsc.ieee.org/
28. IEEE 1484.11.2 Standard for Learning Technology – ECMAScript Application Programming Interface for Content to Runtime Services Communication. November 10, 2003 Available at: http://ltsc.ieee.org/
29. IEEE 1484.12.1-2002 Learning Object Metadata Standard. Available at: http://www.ieee.org/
30. C. Bauer & G. King, Hibernate in Action (Manning Publications 2004)