UNIVERSIDAD DON BOSCO

VICERRECTORÍA DE ESTUDIOS DE POSTGRADO

TRABAJO DE GRADUACIÓN

Diseño de un sistema de portafolio de asignatura con un enfoque por competencia y por objetivos, utilizando metodología de desarrollo ágil.

PARA OPTAR AL GRADO DE:

MAESTRO EN ARQUITECTURA DE SOFTWARE

ASESOR:

MG. CARLOS FILIBERTO ALFARO CASTRO

PRESENTADO POR:

JOSE ALEXANDER BARRERA ALGUERA

CESAR CELESTINO ESPINOZA PEÑA

JORGE ALBERTO MARTINEZ CRUZ

Antiguo Cuscatlán, La Libertad, El Salvador, Centroamérica

Septiembre de 2013

AGRADECIMIENTOS

Quiero agradecerla a Dios todo poderoso, por darme la oportunidad de vivir y por estar conmigo en cada paso que doy, por fortalecer mi corazón e iluminar mi mente y por haber puesto en mi camino a aquellas personas que han sido mi soporte y compañía durante todo el periodo de estudio.

Agradezco a mis padres, hermanos, esposa e hija, por apoyarme siempre en todo y estar ahí con palabras de aliento y comprensión.

A mis amigos y compañeros de la maestría, que con su experiencias profesionales me ayudaron a adquirir nuevos conocimientos.

A mis queridos compañeros de tesina, que pesar que los fastidie mucho para realizar los avances, lograron entenderme. Solamente déjenme decirles gracias por todo.

José Alexander Barrera Alguera

Primeramente quiero darle gracias a Dios, porque a pesar de todas las dificultades y adversidades, siento que él siempre está cuidándome, haciéndome el camino más fácil, para que yo pueda aprovechar las oportunidades y todas las bendiciones que me ha brindado; pues es él, quien ha permitido que yo tenga una familia maravillosa, una esposa extraordinaria con quien he procreado dos preciosas hijas, rodeado de buenos amigos, un trabajo digno que me da el espacio para desarrollarme profesionalmente. No encuentro palabras de gratitud que puedan expresar mi enorme agradecimiento hacia él.

Agradezco a mis padres, por haberme apoyado en mi educación, que con gran sacrificio y paciencia, lograron orientarme para finalizar mi carrera universitaria; sin ellos no hubiera sido posible obtener este importante logro en mi carrera profesional.

A mi esposa por apoyarme en las decisiones y en la comprensión de hacer un esfuerzo para que yo pueda lograr este nuevo éxito.

A mis compañeros de la maestría de Arquitectura de Software de quienes aprendí mucho y la buena relación que tuvimos.

César Celestino Espinoza Peña

Luego de concluir otra etapa importante en mi vida, quiero dar las gracias en primer lugar a Dios todo poderoso por todas las bendiciones que me ha brindado a lo largo de mi vida y en especial por permite culminar una meta más en mi vida, y a la virgen María por guiarme y darme fortaleza de seguir adelante con mis objetivos.

Además agradezco a mi esposa, mi hijo y mi hija por apoyarme en el logro de mis metas y sobre todo por brindarme el apoyo, consejos, paciencia y por compartir junto a mí las dificultades que se nos presentaban para ver reflejado este triunfo que hoy se me da, ya que sin ellos este logro no sería posible.

Agradezco especialmente a mis papas ya que ellos iniciaron el camino en mi vida para estudiar y me dieron la oportunidad de poder convertirme en un profesional, así como también, me brindaron su apoyo incondicional en cada momento que solicite, de igual manera agradezco a mis hermanos y hermana por animarme a seguir estudiando y por todas sus muestras animo hacia mi persona y mi familia.

Al Mg. Carlos Alfaro por la dedicación de su tiempo y conocimiento en diferentes etapas del desarrollo de la carrera que he concluido satisfactoriamente, a la Universidad Don Bosco por brindarme la oportunidad de haber estudiado esta carrera y a mis jefes por creer en mi persona y brindarme esta oportunidad.

Finalmente agradezco a mis compañeros de maestría en especial a mis compañeros y amigos de trabajo de graduación por confiar siempre en mí y sobre todo por tener siempre la convicción de que saldríamos ganadores en este proceso que iniciamos juntos y hoy terminamos juntos, ya que he aprendido mucho de sus experiencias y me han brindado su amistad.

Jorge Alberto Martínez Cruz

INDICE

INDICE DE TABLAS

INDICE DE FIGURAS

RESUMEN

La Universidad Don Bosco posee un plan académico el cual consiste en el desarrollo de un Portafolio de Asignatura basado en un enfoque por objetivos, la institución posee una herramienta informática que sistematiza la gestión del proceso enseñanza aprendizaje, el cual es utilizado por el 100% de docentes de pregrado. Esta herramienta es un sub sistema que forma parte de un portal Académico, que se encuentra alojado en la página principal de la Universidad como servicios académicos en línea.

El proyecto consiste en diseñar una propuesta de software, que incorpore en la herramienta actual componentes bajo el enfoque por competencia. Para el diseño de la herramienta haremos uso de una de las metodologías de desarrollo ágil, caracterizando las más conocidas y en base al estudio de ellas, justificar la metodología seleccionada.

I. INTRODUCCION

La Universidad ha emprendido un nuevo proyecto que consiste en incluir en su modelo educativo el enfoque de educación basado en competencias. La educación basada en competencias se refiere, en primer lugar, a una experiencia práctica y a un comportamiento que necesariamente se enlaza a los conocimientos para lograr sus fines. Deja de existir la división entre teoría y práctica porque de esta manera la teoría depende de la práctica, implica la exigencia de analizar y resolver problemas y de encontrar alternativas frente a las situaciones que plantean dichos problemas, la capacidad de trabajar en equipos multidisciplinarios y la facultad de aprender a aprender y adaptarse.

En el presente documento se explicará la problemática que existe en incorporar el enfoque por competencia al programa académico nombrado Portafolio de Asignatura implementado por la institución; comprender las necesidades planteadas y a raíz de eso crear un diseño de software que proponga una solución al problema.

Para diseñar la herramienta haremos uso de una de las metodologías de desarrollo de software ágil. En el área de Ingeniería de Software, el término metodología (Pressman, 2005) se refiere a un marco de trabajo usado para estructurar, planificar y controlar el proceso de desarrollo de sistemas computacionales.

Las Metodologías Agiles, presentan respuestas rápidas y efectivas al cambio; tienen un plan de proyecto flexible, y muestran simplicidad, de manera general, en el desarrollo. Sin embargo, tienen la desventaja de generar poca documentación y no hacer uso de métodos formales; debido a que estas metodologías han sido diseñadas para que se puedan implementar en proyectos en que el número de personas involucradas es reducido.

Así, se espera que al utilizar una metodología para el desarrollo del software, ésta pueda proveer un conjunto de prácticas y herramientas que faciliten el proceso de desarrollo, ofreciendo un producto con alta calidad y que satisfaga las expectativas del cliente.

II. MARCO TEORICO

La Universidad Don Bosco es una institución educativa de nivel superior, de utilidad pública, apolítica, de inspiración cristiana y sin fines de lucro.

Con una amplia oferta de carreras de pregrado y postgrado, cursos de especialización y formación continua e inspirada por el carisma salesiano, ha contribuido a la formación de profesionales integrales del más alto nivel.
La Universidad Don Bosco de El Salvador posee dos campus:

- Ciudadela Don Bosco, Soyapango, San Salvador.
- Antiguo Cuscatlán, La Libertad.

En 1998 nace la idea de desarrollo de un portafolio por docente, a raíz de esta iniciativa se implementó un muestreo por escuela, en donde el director de escuela selecciono a un docente para que desarrollara un portafolio. Al cabo de cinco años se creó un comité de portafolios, con la finalidad de analizar todas las muestras recopiladas y crear una propuesta de los elementos que debía tener el portafolio[6].

En el 2006 el comité de portafolios, inicio el proyecto denominado “Portafolio Docente” implementando una herramienta informática, que estandarizaba los elementos del Portafolio Docente y cada docente debía elaborar su portafolio. El comité siguió analizando la información y después de dos años, se lanzó una nueva propuesta como “Portafolio de Asignatura” que consiste en evidenciar los elementos académicos que componen la praxis de la docencia, estos elementos son: Planificación de la asignatura, Planes de clases, Enfoque metodológico, Planificación de evaluaciones, Materiales didácticos, Trabajos de Estudiantes, Asistencia, Calificaciones y Análisis reflexivo. Se creó una nueva herramienta que se implementó en el año 2008 y se ha convertido en el espacio de trabajo de todos los docentes que laboran en la institución, para las carreras de pregrado.

La Universidad pretende incluir en su modelo educativo el enfoque de educación por competencias y los elementos que componen el portafolio de asignatura, no proveen información de las competencias adquiridas por los estudiantes y se ha desarrollado

desde un enfoque constructivista de cumplimiento de objetivos. La Universidad requiere incorporar una solución que supla las necesidades en el portafolio de asignatura.

La primera cuestión a plantear a la hora de establecer la metodología sobe el proceso de enseñanza-aprendizaje es establecer las distintas modalidades de enseñanza que se van a tener en cuenta a la hora de impartir una materia para que los alumnos adquieran los aprendizajes establecidos. Consideraremos como modalidades de enseñanza al conjunto de actividades a realizar por los alumnos, de una forma secuencial izada, a lo largo de un curso y/o asignatura, haciendo hincapié en las técnicas, recursos y tiempos que necesitarán para su ejecución. Lógicamente diferentes modalidades de enseñanza reclaman tipos de trabajos distintos para profesores y alumnos y exigen la utilización de herramientas metodológicas también diferentes.

La práctica más habitual y característica en la enseñanza universitaria es la clase teórica, estrategia que, por sí sola, no es muy recomendable para el fomento de aprendizaje autónomo de los alumnos. Por ello es importante que al elaborar el diseño de la metodología de trabajo, además de los contenidos de las asignaturas, precisemos las modalidades de enseñanza que vamos a utilizar para organizar la trayectoria curricular y las experiencias de aprendizaje de los estudiantes.

Al menos, deberemos especificar aquellas actividades que son más habituales en la enseñanza superior y que tienen que ver con las siguientes modalidades presenciales y no presenciales. Entendemos por actividades presenciales aquellas que reclaman la intervención directa de profesores y alumnos como son las clases teóricas, los seminarios, las clases prácticas, las prácticas externas y las tutorías. Se consideran como modalidades no presenciales las actividades que los alumnos pueden realizar libremente bien de forma individual ó mediante trabajo en grupo.

Aunque no es fácil aceptar una conceptuación del término competencias podríamos reconocer que supone la combinación de tres elementos:

a) Información
b) Desarrollo de una habilidad
c) Aplicarlas en una situación inédita.

La mejor manera de observar una competencia, es en la combinación de estos tres aspectos, lo que significa que toda competencia requiere del dominio de una información específica, al mismo tiempo que reclama el desarrollo de una habilidad ó mejor dicho una serie de habilidades derivadas de los procesos de información, pero es en una situación problema, esto es, en una situación real inédita, donde la competencia se puede generar.

La competencia de una persona abarca la gama completa de sus conocimientos y sus capacidades en el ámbito personal, profesional ó académico, adquiridas por diferentes vías y en todos los niveles, del básico al más alto. La educación tradicional actual se basa exclusivamente en el uso y manejo de la palabra, el copiar, transcribir, resumir y memorizar. Las competencias en la nueva educación contienen el potencial para convertirse en un plan efectivo tendiente a mejorar el aprendizaje de los estudiantes y debe ser un reto que debemos aceptar e integrarlo en nuestra cultura académica en donde el docente deja de lado los objetivos tradicionales para sus cursos donde se dictaban conferencias y utilizaban métodos de evaluación cerrados, para dar paso a una figura mediadora y facilitadora donde será necesario dedicar la mayor parte de su tiempo a la observación del desempeño de los alumnos y a la asesoría ya que las acciones educativas se reconocerán a través de las certificaciones con esto tendríamos un vigoroso instrumento para enriquecer el currículum, fortalecer el aprendizaje y con ello acortar la distancia que se ha ido abriendo entre educación universitaria y práctica profesional.

Es probable que el enfoque de competencias pueda mostrar su mayor riqueza si se logra incorporar de manera real en la tarea docente, en la promoción de ambientes de aprendizaje escolares. En este sentido se trataría de pasar de los modelos centrados en la información hacia modelos centrados en desempeños. Los conceptos de movilización de la información, de transferencia de las habilidades hacia situaciones inéditas adquieren una importancia en esta perspectiva.

Al comprender la importancia que representa para el sistema educativo el enfoque por competencias, implica cambios y transformaciones profundas en los diferentes niveles educativos, y seguir este enfoque es comprometerse con una docencia de calidad, buscando asegurar el aprendizaje de los estudiantes, es por esto que se plantea el proyecto para iniciar el diseño del portafolio asignatura tomando en cuenta el enfoque por competencia, evolucionando así el sistema del portafolio de asignaturas, para que tanto

docentes como estudiantes interactúen con la herramienta para enriquecer de mejor manera el proceso enseñanza-aprendizaje, y el diseño de dicho proyecto se realizara haciendo uso de las metodologías de desarrollo ágil que mejor se acoplen al desarrollo del sistema en cuestión.

III. TECNOLOGIAS

Es muy importante que para todo proyecto y en especifico para el desarrollo de software es necesario contar con tecnología, que nos permita crear un producto, confiable duradero y sobre todo que satisfaga las necesidades para lo cual es construido, es de esta forma que para desarrollar el presente proyecto, nos apoyamos en las tecnologías que a continuación se detallan:

A. Portal Académico.

Portal web de la Universidad Don Bosco donde están alojados la mayoría de sistemas web que utilizan los estudiantes, docentes y personal que labora para la universidad.

Dicho portal esta desarrollado con el IDE de Microsoft específicamente el visual studio 2010, programado con asp.NET; además sus bases de datos están gestionadas con Microsoft SQL server 2000 y el diseño del portal se ha desarrollado con la herramienta Enterprise Architect. El portal está alojado en un servidor que opera con Windows Server 2008 R2.

B. Portafolio Asignatura.

Sistema implementado dentro del portal académico, el cual gestiona los elementos académicos que componen la praxis de la docencia dentro de la universidad en las carreras de pregrado, estos elementos son: Planificación de la asignatura, Planes de clases, Enfoque metodológico, Planificación de evaluaciones, Materiales didácticos, Trabajos de Estudiantes, Asistencia, Calificaciones y Análisis reflexivo.

C. Visual Studio 2010

Microsoft Visual Studio es un entorno de desarrollo integrado (IDE, por sus siglas en inglés) para sistemas operativos Windows. Soporta varios lenguajes de programación tales como Visual C++, Visual C#, Visual J#, y Visual Basic .NET, al igual que entornos de desarrollo web como ASP.NET. Aunque actualmente se han desarrollado las extensiones necesarias para muchos otros.

Visual Studio permite a los desarrolladores crear aplicaciones, sitios y aplicaciones web, así como servicios web en cualquier entorno que soporte la plataforma .NET (a partir de la versión .NET 2002). Así se pueden crear aplicaciones que se intercomuniquen entre estaciones de trabajo, páginas web y dispositivos móviles.

Acompañada por .NET Framework 4.0. La fecha del lanzamiento de la versión final fue el 12 de abril de 2010, hasta ahora, uno de los mayores logros de la versión 2010 de Visual Studio ha sido el de incluir las herramientas para desarrollo de aplicaciones para Windows 7, tales como herramientas para el desarrollo de las características de Windows 7 (System.Windows.Shell) y la Ribbon Preview para WPF.

Entre sus más destacables características, se encuentran la capacidad para utilizar múltiples monitores, así como la posibilidad de desacoplar las ventanas de su sitio original y acoplarlas en otros sitios de la interfaz de trabajo.
Además ofrece la posibilidad de crear aplicaciones para muchas plataformas de Microsoft, como Windows, Azure, Windows Phone 7 ó Sharepoint. Microsoft ha sido sensible a la nueva tendencia de las pantallas táctiles y con este Visual Studio 2010 también es posible desarrollar aplicativos para pantallas multitáctiles.
Entre las ediciones disponibles de Visual Studio 2010 que podemos adquirir se encuentran:

Visual Studio 2010 Ultimate: Conjunto completo de herramientas de gestión del ciclo de vida de una aplicación para los equipos que garantizan unos resultados de calidad, desde el diseño hasta la implementación. Ya sea creando nuevas soluciones ó mejorando las aplicaciones existentes, Visual Studio 2010 Ultimate le permite llevar sus ideas a la vida en un número creciente de plataformas y tecnologías - incluyendo la nube y la computación paralela.

Visual Studio 2010 Premium: Un conjunto de herramientas completo que simplifica el desarrollo de aplicaciones para personas ó equipos que entregan aplicaciones escalables de alta calidad. Que este escribiendo código de aplicaciones ó de bases de datos, creando bases de datos, ó quitando los errores, puede aumentar su productividad usando herramientas poderosas que funcionan de la manera que usted trabaja.

Visual Studio 2010 Professional: La herramienta esencial para las personas que realizan tareas de desarrollo básico. Visual Studio 2010 Professional simplifica la compilación, la depuración y el despliegue de las aplicaciones en una variedad de plataformas incluyendo SharePoint y la Nube. También viene con el soporte integrado

para el desarrollo con pruebas y con las herramientas de depuración que ayudan a garantizar unas soluciones de alta calidad.

Visual Studio Team Foundation Server 2010: Una plataforma de colaboración en el centro de la solución de gestión del ciclo de vida de una aplicación (ALM) de Microsoft. Team Foundation Server 2010 automatiza el proceso de entrega del software y le da las herramientas que necesita para gestionar eficazmente los proyectos de desarrollo de software a través del ciclo de vida de IT.

Visual Studio Test Professional 2010: Visual Studio Test Professional 2010 es un conjunto de herramientas integrado que entrega un flujo de trabajo completo planificar-probar-seguir para una colaboración en contexto entre los probadores y los desarrolladores, aumentando considerablemente la visibilidad de los probadores en la globalidad del proyecto.

Visual Studio Team Explorer Everywhere 2010: Permite a los equipos de desarrollo colaborar fácilmente entre las plataformas. Team Explorer Everywhere 2010 contiene las herramientas y los plug-ins necesarios para acceder a Visual Studio Team Foundation Server 2010 desde dentro de los entornos basados en Eclipse, de manera que todo el mundo puede trabajar juntos y lograr los objetivos del negocio[14].

D. Enterprise Architect.

Sparx Systems Enterprise Architect es una herramienta de modelado visual y diseño basado en el OMG UML. La plataforma soporta: el diseño y construcción de sistemas de software, modelado de procesos de negocio, y los dominios de base de la industria de modelado. Es utilizado por las empresas y organizaciones que no sólo en el modelo de la arquitectura de su sistema, pero para procesar la aplicación de estos modelos a través del desarrollo de la plena aplicación del ciclo de vida.

El Modelado de sistemas usando UML proporciona una base para el modelado de todos los aspectos de la arquitectura organizativa, junto con la capacidad de proporcionar una base para el diseño e implementación de nuevos sistemas ó modificar los sistemas existentes. Los aspectos que pueden ser objeto de este tipo de gama de modelos de trazar arquitecturas organizativas ó de sistemas negocio,

reingeniería de procesos, análisis de negocio y arquitecturas orientadas al servicio y modelado web, a través de la aplicación y diseño de base de datos y re-ingeniería y desarrollo de sistemas embebidos.

Junto con el modelado de sistemas, Enterprise Architect cubre los aspectos básicos del ciclo de vida de desarrollo de aplicaciones, desde la gestión de requisitos a través de las fases de diseño, construcción, pruebas y mantenimiento, con el apoyo de la trazabilidad, gestión de proyectos y de cambio de control de estos procesos, así como, facilidades para el modelo de desarrollo impulsado por código de la aplicación utilizando una plataforma de desarrollo integrado interno[13].

E. Microsoft SQL Server 2000

SQL Server 2000 es un sistema de gestión de bases de datos relacionales (SGDBR o RDBMS: Relational Database Management System) diseñado para trabajar con grandes cantidades de información y la capacidad de cumplir con los requerimientos de proceso de información para aplicaciones comerciales y sitios Web.

SQL Server 2000 ofrece el soporte de información para las tradicionales aplicaciones Cliente/Servidor, las cuales están conformadas por una interfaz a través de la cual los clientes acceden a los datos por medio de una LAN.

La hoy emergente plataforma NET exige un gran porcentaje de distribución de recursos, desconexión a los servidores de datos y un entorno descentralizado, para ello sus clientes deben ser livianos, tales como los navegadores de Internet los cuales accederán a los datos por medio de servicios como el Internet Information Services(IIS).

SQL Server 2000 está diseñado para trabajar con dos tipos de bases de datos:

- OLTP (OnLine Transaction Processing) Son bases de datos caracterizadas por mantener una gran cantidad de usuarios conectados concurrentemente realizando ingreso y/o modificación de datos. Por ejemplo: entrada de pedidos en línea, inventario, contabilidad ó facturación.
- OLAP (OnLine Analytical Processing) Son bases de datos que almacenan grandes cantidades de datos que sirven para la toma de decisiones, como por ejemplo las aplicaciones de análisis de ventas.

SQL Server puede ejecutarse sobre redes basadas en Windows Server así como sistema de base de datos de escritorio en máquinas Windows NT Workstation, Windows Millenium y Windows 98.

Los entornos Cliente/Servidor, están implementados de tal forma que la información se guarde de forma centralizada en un computador central (servidor), siendo el servidor responsable del mantenimiento de la relación entre los datos, asegurarse del correcto almacenamiento de los datos, establecer restricciones que controlen la integridad de datos, entre otros.

Del lado cliente, este corre típicamente en distintas computadoras las cuales acceden al servidor a través de una aplicación, para realizar la solicitud de datos los clientes emplean el Structured Query Language (SQL), este lenguaje tiene un conjunto de comandos que permiten especificar la información que se desea recuperar ó modificar[15].

F. ASP.NET

ASP.NET es una tecnología gratuita que permite a los programadores crear páginas web dinámicas, desde websites personales hasta aplicaciones web empresariales.
Actualmente, ASP.NET soporta tres modelos de programación: ASP.NET Web Forms, ASP.NET MVC y ASP.NET Web Pages. Aunque los tres modelos de programación se ejecutan sobre la misma base de ASP.NET, cada uno de ellos estructura la aplicación de maneras completamente distintas, promueve metodologías de desarrollo diferentes y se adapta a perfiles de desarrolladores distintos. Algunas características que son virtudes en unos modelos de programación, pueden ser consideradas debilidades en el otro. ¿Qué es más importante, desarrollar a un gran nivel de abstracción ó tener control total cada uno de los aspectos de la aplicación? Simplicidad vs. Control. Flexibilidad vs. Eficiencia. Estas son las compensaciones que hay que baremar a la hora de elegir. En esta serie de artículos repasaremos las diferencias entre los tres modelos de programación, y los escenarios favorables a cada uno de ellos.

Es importante recalcar que el hecho de elegir uno de los modelos de programación al comenzar un proyecto de ASP.NET no excluye necesariamente a los otros, sino que es posible tener aplicaciones “híbridas” y en muchos casos tendrá todo el sentido desarrollar ciertas partes de la aplicación con un modelo de programación y otras partes con otro modelo distinto.
ASP.NET Web Forms fue el primero de los tres modelos de programación en existir, y proporciona un gran nivel de abstracción con un modelo de programación familiar

basado en eventos y controles que favorece la productividad mediante la programación declarativa reduciendo la cantidad de código necesaria para implementar una determinada funcionalidad.

ASP.NET MVC se concibió como alternativa a Web Forms y proporciona un modelo de programación basado en el popular patrón de arquitectura MVC. Entre sus principales características destacan su completa integración con pruebas unitarias y su separación más clara entre la lógica de presentación, la lógica de negocio y la lógica de acceso a datos.

ASP.NET Web Pages es el más reciente de los tres modelos de programación, y fue creado como respuesta a una creciente demanda de desarrolladores web sin experiencia previa con ASP.NET, cuya iniciación en ASP.NET Web Forms ó MVC les suponía una inversión inicial de tiempo demasiado grande. Web Pages proporciona un modelo de programación más simple y rápido de aprender, sin renunciar a toda la funcionalidad y flexibilidad de ASP.NET[16].

G. Windows Server 2008

Sistema operativo de Microsoft diseñado para servidores, sucesor de Windows Server 2003 y se basa en el núcleo de Windows NT 6.1. Nuevas funcionalidades para el Active Directory, nuevas prestaciones de virtualización y administración de sistemas son dentro de algunas de las mejoras de esta versión junto con la inclusión del IIS 7.5 y el soporte para más de 256 procesadores. Hay siete ediciones diferentes de Windows server 2008: Foundation, Standard, Enterprise, Datacenter, Web Server, HPC Server y para Procesadores Itanium[17].

IV. METODOLOGIAS DE DESARROLLO AGIL

Con mucho el desarrollo de software es una actividad caótica, frecuentemente caracterizada por la frase "codifica y corrige". El software se escribe con un mínimo plan subyacente, y el diseño del sistemas es adoquina con muchas decisiones a corto plazo. Esto realmente funciona muy bien si el sistema es pequeño, pero con forme el sistema crece llega a ser cada vez más difícil agregar nuevos aspectos al mismo. Además los errores llegan a ser cada vez más frecuentes y más difíciles de corregir. La señal típica de tal sistema es una larga fase de pruebas después de que el sistema ha sido "completado". Tal fase larga de pruebas hace estragos con los planes de pruebas y depurado llegando a ser imposible de poner en el programa de trabajo.

El desarrollo ágil de software es un grupo de metodologías de desarrollo de software que se basan en principios similares. Las metodologías ágiles promueven generalmente un proceso de gestión de proyectos que fomenta el trabajo en equipo, la organización y responsabilidad propia, un conjunto de mejores prácticas de ingeniería que permiten la entrega rápida de software de alta calidad, y un enfoque de negocio que alinea el desarrollo con las necesidades del cliente y los objetivos de la compañía.

Como una reacción a estas metodologías, un nuevo grupo de metodologías ha surgido en los últimos años. Durante algún tiempo se conocían como las metodologías ligeras, pero el término aceptado ahora es metodologías ágiles. Para mucha gente el encanto de estas metodologías ágiles es su reacción a la burocracia de las metodologías monumentales. Estos nuevos métodos buscan un justo medio entre ningún proceso y demasiado proceso, proporcionando simplemente suficiente proceso para que el esfuerzo valga la pena.

Las metodologías ágiles son adaptativas más que predictivas. Las metodologías tradicionales potencian la planificación detallada de prácticamente todo el desarrollo software a largo plazo. Pero cuando se produce un cambio, toda esta planificación puede venirse abajo. Sin embargo, las metodologías ágiles proponen procesos que se adaptan y progresan con el cambio, llegando incluso hasta el punto de cambiar ellos mismos

Las metodologías ágiles están orientadas al personal más que orientadas al proceso. Intentan trabajar con la naturaleza del personal asignado al desarrollo, más que contra

ellos, de tal forma que permiten que la actividad de desarrollo software se convierta en una actividad grata e interesante.

Para asegurar el éxito durante el desarrollo de software no es suficiente contar con notaciones de modelado y herramientas, hace falta un elemento importante: la metodología de desarrollo, la cual nos provee de una dirección a seguir para la correcta aplicación de los demás elementos.

Generalmente el proceso de desarrollo llevaba asociado un marcado énfasis en el control del proceso mediante una rigurosa definición de roles, actividades y artefactos, incluyendo modelado y documentación detallada. Este esquema "tradicional" para abordar el desarrollo de software ha demostrado ser efectivo y necesario en proyectos de gran tamaño (respecto a tiempo y recursos), donde por lo general se exige un alto grado de ceremonia en el proceso. Sin embargo, este enfoque no resulta ser el más adecuado para muchos de los proyectos actuales donde el entorno del sistema es muy cambiante, y en donde se exige reducir drásticamente los tiempos de desarrollo pero manteniendo una alta calidad.

Ante las dificultades para utilizar metodologías tradicionales con estas restricciones de tiempo y flexibilidad, muchos equipos de desarrollo se resignan a prescindir de las buenas prácticas de la Ingeniería del Software, asumiendo el riesgo que ello conlleva. En este contexto, las metodologías ágiles emergen como una posible respuesta para llenar ese vacío metodológico.

Por estar especialmente orientadas para proyectos pequeños, las Metodologías Ágiles constituyen una solución a medida para ese entorno, aportando una elevada simplificación que a pesar de ello no renuncia a las prácticas esenciales para asegurar la calidad del producto.

“The Agile Alliance”, una organización, sin ánimo de lucro, dedicada a promover los conceptos relacionados con el desarrollo ágil de software y ayudar a las organizaciones para que adopten dichos conceptos. El punto de partida fue el Manifiesto Ágil, un documento que resume la filosofía “ágil”. En el manifiesto ágil se encuentran los principales valores del desarrollo ágil los cuales se mencionan a continuación:

- Al individuo y las interacciones del equipo de desarrollo sobre el proceso y las herramientas.
- Desarrollar software que funciona más que conseguir una buena documentación.
- La colaboración con el cliente más que la negociación de un contrato.
- Responder a los cambios más que seguir estrictamente un plan.

Los valores anteriores inspiran los doce principios del manifiesto. Estos principios son las características que diferencian un proceso ágil de uno tradicional. Los dos primeros son generales y resumen gran parte del espíritu ágil, luego existen una serie de principios que tienen que ver directamente con el proceso de desarrollo de software a seguir. Los doce principios del manifiesto son los siguientes:

I. La prioridad es satisfacer al cliente mediante tempranas y continuas entregas de software que le aporte un valor.
II. Dar la bienvenida a los cambios. Se capturan los cambios para que el cliente tenga una ventaja competitiva.
III. Entregar frecuentemente software que funcione desde un par de semanas a un par de meses, con el menor intervalo de tiempo posible entre entregas.
IV. La gente del negocio y los desarrolladores deben trabajar juntos a lo largo del proyecto.
V. Construir el proyecto en torno a individuos motivados.
VI. El diálogo cara a cara es el método más eficiente y efectivo para comunicar información dentro de un equipo de desarrollo.
VII. El software que funciona es la medida principal de progreso.
VIII. Los procesos ágiles promueven un desarrollo sostenible.
IX. La atención continua a la calidad técnica y al buen diseño mejora la agilidad.
X. La simplicidad es esencial.
XI. Las mejores arquitecturas, requisitos y diseños surgen de los equipos organizados por sí mismos.
XII. En intervalos regulares, el equipo reflexiona respecto a cómo llegar a ser más efectivo, y según esto ajusta su comportamiento.

4.1 Descripción de metodologías

El uso de un método ágil no es para todo el mundo. Hay un número de aspectos que hay que tener en mente si se decide seguir este camino. Por otro lado, estas metodologías se consideran ampliamente aplicables.

Para alguien nuevo en métodos ágiles, la pregunta es por dónde empezar. Como con cualquier nueva tecnología ó proceso, es necesario hacer nuestra propia evaluación de ello. Esto permite ver cómo encaja en un entorno.

El primer paso es encontrar proyectos adecuados para probar los métodos ágiles. Debido a que los métodos ágiles están fundamentalmente orientados a las personas, es esencial que se empiece con un equipo que quiera probar y trabajar de forma ágil.

Es valioso también tener clientes (aquellos que necesitan el software) que quieran trabajar de forma colaborativa. Si los clientes no colaboran, no se verán las ventajas completas de un proceso adaptativo. Los proyectos más grandes son indudablemente más complicados, así que es mejor aprender con proyectos de tamaño más manejable.

A continuación se explican brevemente las tres metodologías principalmente difundidas y sobre todo utilizadas por equipos desarrolladores de software y en las cuales se ha centrado nuestra atención para realizar el diseño del proyecto propuesto. Las tres metodologías agiles más difundidas son las siguientes:

- RUP (Rational Unified Process)
- XP (Xtreme Programming)
- MSF (Microsoft Solutions Framework)

4.1.1 Metodología RUP

RUP es una metodología que tiene como objetivo ordenar y estructurar el desarrollo de software, en la cual se tienen un conjunto de actividades necesarias para transformar los requisitos del usuario en un sistema Software. (Inicialmente fue llamada UP (Unified Process) y luego cambió su nombre a RUP por el respaldo de Rational Software de IBM. Ésta metodología fue lanzada en 1998 teniendo como sus creadores a Ivar Jacobson, Grady Booch y James Rumbaugh. El RUP nació del UML (Unified Modeling Language) y del UP.

El RUP es un proceso basado en los modelos en Cascada y por Componentes, el cual presenta las siguientes características: Es dirigido por los casos de uso, es centrado en la arquitectura, iterativo e incremental (Booch, Rumbaugh y Jacobson, 2000), lo cual es fundamental para el proceso de desarrollo de software. A continuación se explican las tres características de RUP:

a) **Casos de Uso:** Describe un servicio que el usuario requiere del sistema, incluye la secuencia completa de interacciones entre el usuario y el sistema.

b) **Centrado en la arquitectura:** Comprende las diferentes vistas del sistema en desarrollo, que corresponden a los modelos del sistema: Modelos de casos de uso, de análisis, de diseño, de despliegue e implementación. La arquitectura del software es importante para comprender el sistema como un todo y a la vez en sus distintas partes, sirve para organizar el desarrollo, fomentar la reutilización de componentes y hacer evolucionar el sistema, es decir, agregarle más funcionalidad.

En síntesis, la metodología RUP es un proceso de desarrollo de software que trabaja de la mano con el UML y es una de las metodologías estándar más usadas para el análisis, desarrollo y documentación de sistemas orientados a objetos, además de su gran respaldo por parte de IBM.

Por sus características se implementa con mayor frecuencia en proyecto de gran complejidad y magnitud, que dispongan de un equipo de trabajo con experiencia en desarrollo de proyectos, así como con un alto conocimiento en la metodología, sin dejar

de lado su aplicabilidad en proyectos de corto tiempo y poca complejidad, pues la metodología tiene la capacidad de poder adaptarse a los diferentes tipos de proyecto software.

El proceso de ciclo de vida de RUP se divide en cuatro fases bien conocidas llamadas Incepción, Elaboración, Construcción y Transición. Esas fases se dividen en iteraciones, cada una de las cuales produce una pieza de software demostrable. La duración de cada iteración puede extenderse desde dos semanas hasta seis meses. Las fases son:

1. **Incepción.** Significa “comienzo”, pero la palabra original (de origen latino y casi en desuso como sustantivo) es sugestiva y por ello la traducimos así. Se especifican los objetivos del ciclo de vida del proyecto y las necesidades de cada participante. Esto entraña establecer el alcance y las condiciones de límite y los criterios de aceptabilidad. Se identifican los casos de uso que orientarán la funcionalidad.

   Se diseñan las arquitecturas candidatas y se estima la agenda y el presupuesto de todo el proyecto, en particular para la siguiente fase de elaboración. Típicamente es una fase breve que puede durar unos pocos días ó unas pocas semanas.

2. **Elaboración.** Se analiza el dominio del problema y se define el plan del proyecto. RUP presupone que la fase de elaboración brinda una arquitectura suficientemente sólida junto con requerimientos y planes bastante estables. Se describen en detalle la infraestructura y el ambiente de desarrollo, así como el soporte de herramientas de automatización. Al cabo de esta fase, debe estar identificada la mayoría de los casos de uso y los actores, debe quedar descripta la arquitectura de software y se debe crear un prototipo de ella. Al final de la fase se realiza un análisis para determinar los riesgos y se evalúan los gastos hechos contra los originalmente planeados.

3. **Construcción.** Se desarrollan, integran y verifican todos los componentes y rasgos de la aplicación. RUP considera que esta fase es un proceso de manufactura, en el que se debe poner énfasis en la administración de los recursos y el control de costos, agenda y calidad. Los resultados de esta fase (las versiones alfa, beta y otras versiones de prueba) se crean tan rápido como sea posible. Se debe compilar también una versión de entrega. Es la fase más prolongada de todas.

4. **Transición.** Comienza cuando el producto está suficientemente maduro para ser entregado. Se corrigen los últimos errores y se agregan los rasgos pospuestos. La fase consiste en prueba beta, piloto, entrenamiento a usuarios y despacho del producto a mercadeo, distribución y ventas. Se produce también la documentación. Se llama transición porque se transfiere a las manos del usuario, pasando del entorno de desarrollo al de producción.

A través de las fases se desarrollan en paralelo nueve workflows ó disciplinas: Modelado de Negocios, Requerimientos, Análisis & Diseño, Implementación, Prueba, Gestión de Configuración & Cambio, Gestión del Proyecto y Entorno. Además de estos workflows, RUP define algunas prácticas comunes:

1. Desarrollo iterativo de software. Las iteraciones deben ser breves y proceder por incrementos pequeños. Esto permite identificar riesgos y problemas tempranamente y reaccionar frente a ellos en consecuencia.

2. Administración de requerimientos. Identifica requerimientos cambiantes y postula una estrategia disciplinada para administrarlos.

3. Uso de arquitecturas basadas en componentes. La reutilización de componentes permite asimismo ahorros sustanciales en tiempo, recursos y esfuerzo.

4. Modelado visual del software. Se deben construir modelos visuales, porque los sistemas complejos no podrían comprenderse de otra manera. Utilizando una herramienta como UML, la arquitectura y el diseño se pueden especificar sin ambigüedad y comunicar a todas las partes involucradas.

5. Prueba de calidad del software. RUP pone bastante énfasis en la calidad del producto entregado.

6. Control de cambios y trazabilidad. La madurez del software se puede medir por la frecuencia y tipos de cambios realizados.

Aunque RUP es extremadamente locuaz en muchos respectos, no proporciona lineamientos claros de implementación que puedan compararse, por ejemplo, a los métodos Crystal, en los que se detalla la documentación requerida y los roles según diversas escalas de proyecto. En RUP esas importantes decisiones de dejan a criterio del usuario. Se asegura que RUP puede implementarse "sacándolo de la caja", pero dado que el número de sus artefactos y herramientas es inmenso, siempre se dice que hay que recortarlo y adaptarlo a cada caso[4].

4.1.2 Metodología MSF

MSF es una guía de desarrollo de software flexible que permite aplicar de manera individual e independiente cada unos de sus componentes, es escalable pues está diseñada para poder expandirse según la magnitud del proyecto. La metodología MSF está basada en un conjunto de principios, modelos, disciplinas, conceptos, directrices y practicas aprobadas por Microsoft, que asegura resultados con menor riesgo y de mayor calidad, centrándose en el proceso y las personas.

Microsoft Solutions Framework se introdujo por primera vez en 1994 como un conjunto de las mejores prácticas en los desarrollo de Software de Microsoft y Microsoft Consulting Service. Esta metodología ha estado evolucionando y mejorando con la experiencia de grupos de trabajo reales los cuales contribuyeron a perfeccionar este Framework (Wilmot et al., 2004). De igual manera, MSF también retoma algunas de las características propias de metodologías tradicionales.

Éste Framework está basado en los modelos espiral y cascada, lo cual indica que toma elementos de los métodos tradicionales que aún son referentes importantes para procesos de software. Es adaptable, flexible y escalable, e independiente de tecnologías, lo cual significa que no se cierra a un sólo modelo de programación sino más bien queda abierto según la naturaleza del proyecto. Usa como referente el DSL (Domain-Specific Language) para realizar el modelado, así como RUP se apoya en UML para hacer el modelado (Microsoft).

Componentes MSF es un FrameWork que contiene tres componentes: Los principios fundamentales, los modelos y las disciplinas. Estos componentes pueden ser utilizados individuamente ó adoptados como un todo integrado según la naturaleza del proyecto (Microsoft, 2003), a continuación se explican los tres componentes:

Principios fundamentales

Los principios de MSF son 8 valores y normas que son comunes en todo el FrameWork, los cuales contribuyen a mejorar el trabajo en equipo y a centrarse en mantener el objetivo del proyecto siempre en marcha, estos principios son:

- **Fomentar la comunicación abierta:** El mantener un buen flujo de información entre los integrantes del equipo, pueden reducir las posibilidades de malentendidos y contribuir a la solución colectiva de problemas.
- **Trabajar hacia una visión compartida:** El equipo de trabajo debe comprender y trabajar hacia las metas y objetivos del proyecto.
- **Empoderar a los miembros del equipo:** Los miembros del equipo SCRUM deben tener funciones claras, de tal manera que puedan desenvolverse como un grupo creativo, eficaz y capaz de cumplir sus propios objetivos, al igual que resolver sus dificultades.
- **Establecer la rendición de cuentas claras y la responsabilidad compartida:** Cada equipo tiene funciones claras y son responsables de una parte de la solución, así como del éxito global del proyecto, trabajando desde la individualidad en procura de objetivos colectivos.
- **Centrarse en ofrecer valor empresarial:** El equipo de trabajo debe procurar ofrecer soluciones reales que satisfagan necesidades y beneficien al comprador del producto, pues entregar un producto bien elaborado y que cumpla los requerimientos para el cual fue diseñado es el objetivo a alcanzar
- **Mantenerse ágil, en espera de un cambio:** El cambio constante debe ser aceptado, es decir, el quipo debe estar preparado para afrontar estas situaciones y ofrecer soluciones eficaces. MSF acepta la combinación de caos y orden lo cual deja entrever las altas posibilidades de que los proyectos software se salgan de control.

- **Invertir en la calidad:** El equipo debe mantener la mentalidad centrada en la calidad, se debe invertir en las personas, en los procesos y las herramientas, pues ésta inversión puede retribuir en resultados de mayor calidad.
- **Aprender de todas las experiencias:** Tener una mentalidad abierta para aprender del éxito ó fracaso de proyectos propios y ajenos.

En conclusión, la metodología propuesta por MSF tiene como propósito lograr entregas con un margen amplio de éxito, basados en la calidad del producto software, teniendo presente las necesidades del cliente y el principio de flexibilidad, así como el cumplimiento con los compromisos adquiridos, la gestión de los costos y la minimización de los riesgos inherentes en todo proyecto de desarrollo de software.

4.1.3 Metodología XP

La programación extrema ó Extreme Programming, es una disciplina de desarrollo de software basada en los métodos ágiles, que evidencia principios tales como el desarrollo incremental, la participación activa del cliente, el interés en las personas y no en los procesos como elemento principal, y aceptar el cambio y la simplicidad . El trabajo fundamental se publicó por Kent Beck en 1999, y tomó el nombre de Programación Extrema por las prácticas reconocidas en el desarrollo de software y por la participación del cliente en niveles extremos. Éste método, al igual que RUP y MSF, también tiene principios los cuales son buenas prácticas a tener presente en el desarrollo del software.

Los principios XP comprenden diez buenas prácticas que involucran al equipo de trabajo, los procesos y el cliente; los cuales son:

- **El juego del planeamiento:** rápidamente determinar el alcance de la próxima liberación mediante la combinación de prioridades del negocio y estimaciones técnicas. A medida que la realidad va cambiando el plan, actualizar el mismo.
- **Pequeñas liberaciones:** poner un sistema simple en producción rápidamente, luego liberar nuevas versiones en ciclos muy cortos.
- **Metáfora (convención de nombres fácil de recordar):** guiar todo el desarrollo con una historia simple y compartida de cómo funciona todo el sistema.

- **Diseño simple:** el sistema deberá ser diseñado tan simple como sea posible en cada momento. La complejidad extra es removida apenas es descubierta.
- **Pruebas:** los programadores constantemente escriben pruebas unitarias, las cuales deben correr sin problemas para que el desarrollo continúe. Los clientes escriben pruebas demostrando que las funcionalidades están terminadas.
- **Refactorización:** los programadores reestructuran el sistema sin cambiar su comportamiento para remover duplicación, mejorar la comunicación, simplificar, ó añadir flexibilidad.
- **Programación en pares:** todo el código de producción es escrito por dos programadores en una máquina.
- **Propiedad colectiva del código:** cualquiera puede cambiar código en cualquier parte del sistema en cualquier momento.
- **Integración continua:** integrar y hacer construcciones del sistema varias veces por día, cada vez que una tarea se completa.
- **Semana de 40 horas:** trabajar no más de 40 horas semanales como regla. Nunca trabajar horas extras durante dos semanas consecutivas.
- **Cliente en el lugar de desarrollo (Cliente-in-situ):** incluir un cliente real en el equipo, disponible a tiempo completo para responder preguntas.
- **Estándares de codificación:** los programadores escriben todo el código de acuerdo con reglas que enfatizan la comunicación a través del mismo[2].

La figura 1, muestra el detalle del recorrido que debe desarrollarse en cada una de las prácticas.

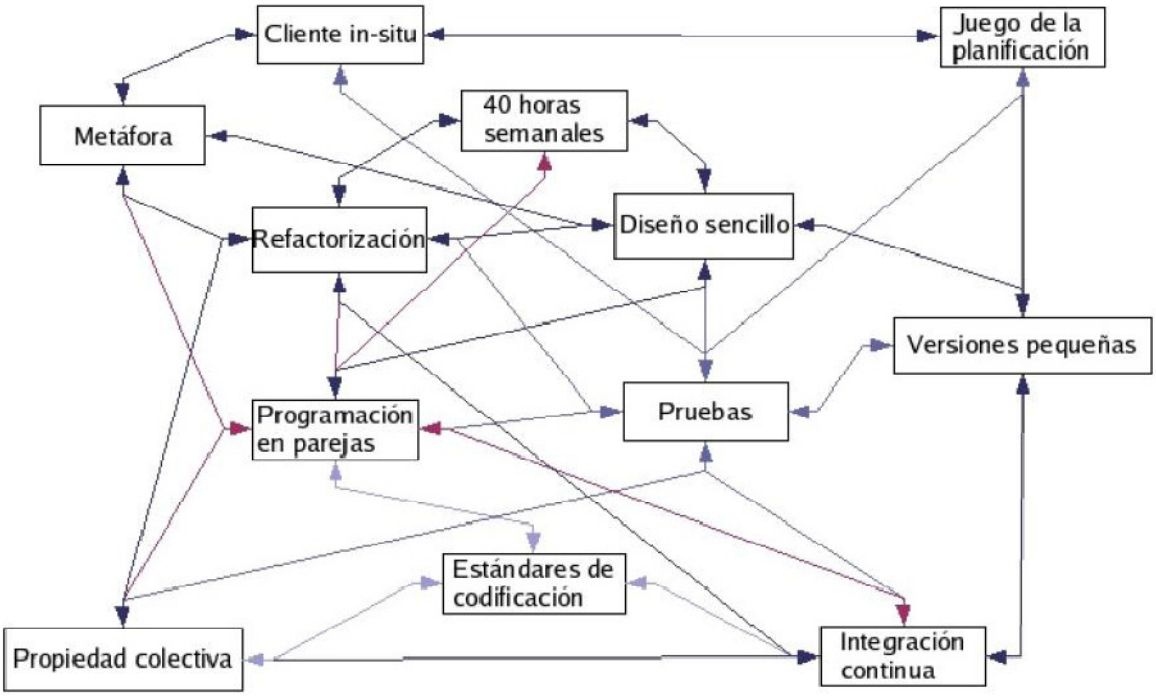


Figura 1: Principios de XP

4.1.3.1 Valores de XP

Los valores que posee XP, son los siguientes:

- **Comunicación**: es lo más importante dentro de un equipo de desarrollo de software, cuando aparecen problemas durante el desarrollo es muy probable que alguien del equipo conozca la solución.
- **Simplicidad**: hay que verla dentro del contexto del proyecto, mejorando la comunicación se ayuda a ganar en simplicidad eliminando requerimientos innecesarios.
- **Retroalimentación**: cuando se habla de desarrollo de software ó de los requerimientos de un sistema, los cambios son inevitables (creando la necesidad de retroalimentación). Es una parte crítica de la comunicación, y contribuye a la simplicidad.

- **Coraje**: en conjunto con los otros valores es poderoso, el coraje para descartar soluciones fallidas y buscar nuevas lleva a la simplicidad, el coraje para buscar respuestas concretas y reales crea retroalimentación.
- **Respeto**: los cuatro valores anteriores apuntan a este, que a su vez yace en la superficie de los anteriores. Si a los miembros de un equipo no les importa los otros y lo que estos hacen, XP no funciona.

4.1.3.2 Roles

En XP se identifican 7 roles:

- **Programador:** estima las historias, define las tareas de las historias y estimados e implementa las historias y las pruebas de unidad.
- **Cliente:** escribe las historias, especifica las pruebas funcionales, es quien determina las prioridades y puede no ser el usuario final.
- **Encargado de pruebas**: implementa y ejecuta las pruebas funcionales, y se asegura de dar a conocer cuando los resultados de las pruebas disminuyen.
- **Encargado de seguimiento**: monitorea el avance del programador y actúa si las cosas parecen irse de su curso normal. Entre las acciones a tomar están: reunirse con el cliente, preguntarle al jefe ó a otro programador para obtener ayuda.
- **Jefe**: lo mira todo, se asegura que el proyecto se mantenga en su curso y ayuda con cualquier cosa.
- **Consultor**: miembro externo al equipo con conocimiento en algún tema necesario al proyecto, guía al equipo a resolver un problema específico.
- **Gestor:** programa las reuniones, se asegura que el proceso es continuo, guarda los resultados de las reuniones para reportes en el futuro, es el vínculo entre los clientes y los programadores.

En síntesis, actualmente XP es una de las metodologías de mayor aceptación en la industria del software, su enfoque basado en los métodos ágiles, su énfasis en la gestión del recurso humano el cuál es uno de los puntos más críticos en todo proyecto, y sus principios de previsibilidad y adaptabilidad hacen de esta metodología una buena opción a seguir.

V. JUSTIFICACION METODOLOGÍCA

A pesar de todas las técnicas, métodos y metodologías existentes, aun no existe nada preciso para que un proyecto de desarrollo de software sea exitoso, sino que deben ser adaptados al contexto del proyecto. Una de las características de las metodologías ágiles es la adaptabilidad y la flexibilidad ante determinadas situaciones, de allí su gran uso y aplicación, entre ellas destacando a XP. Aunque cabe señalar la serie de inconvenientes y restricciones para su aplicación que no se deben pasar por alto si no queremos fracasar a la hora de la gestión de un proyecto de desarrollo de software.

A continuación se muestra un cuadro comparativo entre las metodologías:

Comparación de metodologías agiles			
Metodología	**RUP(Rational Unified Process)**	**XP(Xtreme Programming)**	**MSF(Microsoft Solutions Framework)**
Basado en casos de uso	Si	No	No
Compuesta por un ciclo de vida	Si	Si	Si
Permite un desarrollo rápido	No	Si	No
Mantiene objetivos específicos y entregables para cada ciclo	Si	No	Si
Permite varias alternativas tecnológicas para su desarrollo	Si	Si	Si
Mantiene la calidad del sistema en cada iteración	Si	No, El sistema se depura incrementalmente	Si
Relación con UML	Nativa	Integrable	Integrable

Tabla 1: Comparación de métodos agiles [9]

5.1 Razones para utilizar XP

Entre las razones para la elección de XP como metodología ágil tenemos:

- Simplicidad XP propone el principio de hacer la cosa más simple que pueda funcionar, en relación al proceso y la codificación. Es mejor hacer hoy algo simple, que hacerlo complicado y probablemente nunca usarlo mañana.
- Comunicación Algunos problemas en los proyectos tienen origen en que alguien no dijo algo importante en algún momento. XP hace casi imposible la falta de comunicación.
- Retroalimentación concreta y frecuente del cliente, del equipo y de los usuarios finales da una mayor oportunidad de dirigir el esfuerzo eficientemente.
- Reduce el costo del cambio en todas las etapas del ciclo de vida del sistema.
- Combina las que han demostrado ser las mejores prácticas para desarrollar software, y las lleva al extremo.

5.2 Características relevantes

Entre las características más relevantes que hacen que XP sea una muy buena opción son las siguientes:

- Equipo completo: Forman parte del equipo todas las personas que tienen algo que ver con el proyecto, incluido el cliente y el responsable del proyecto.
- Planificación: Se hacen las historias de usuario y se planifica en qué orden se van a hacer y las mini-versiones. La planificación se revisa continuamente.
- Test del cliente: El cliente, con la ayuda de los desarrolladores, propone sus propias pruebas para validar las mini-versiones.
- Versiones pequeñas: Las mini-versiones deben ser lo suficientemente pequeñas como para poder hacer una cada pocas semanas. Deben ser versiones que ofrezcan algo útil al usuario final y no trozos de código que no pueda ver funcionando.
- Diseño simple: Hacer siempre lo mínimo imprescindible de la forma más sencilla posible. Mantener siempre sencillo el código.

- Pareja de programadores: Los programadores trabajan por parejas (dos delante del mismo ordenador) y se intercambian las parejas con frecuencia (un cambio diario).
- Desarrollo guiado por las pruebas automáticas: Se deben realizar programas de prueba automática y deben ejecutarse con mucha frecuencia. Cuantas más pruebas se hagan, mejor.
- Integración continua: Deben tenerse siempre un ejecutable del proyecto que funcione y en cuanto se tenga una nueva pequeña funcionalidad, debe recompilarse y probarse. Es un error mantener una versión congelada dos meses mientras se hacen mejoras y luego integrarlas todas de golpe. Cuando falle algo, no se sabe qué es lo que falla de todo lo que hemos metido.
- El código es de todos: Cualquiera puede y debe tocar y conocer cualquier parte del código. Para eso se hacen las pruebas automáticas.
- Normas de codificación: Debe haber un estilo común de codificación (no importa cual), de forma que parezca que ha sido realizado por una única persona.
- Metáforas: Hay que buscar unas frases ó nombres que definan cómo funcionan las distintas partes del programa, de forma que sólo con los nombres se pueda uno hacer una idea de qué es lo que hace cada parte del programa. Un ejemplo claro es el "recolector de basura" de java. Ayuda a que todos los programadores (y el cliente) sepan de qué estamos hablando y que no haya mal entendidos.
- Ritmo sostenible: Se debe trabajar a un ritmo que se pueda mantener indefinidamente. Esto quiere decir que no debe haber días muertos en que no se sabe qué hacer y que no se deben hacer un exceso de horas otros días. Al tener claro semana a semana lo que debe hacerse, hay que trabajar duro en ello para conseguir el objetivo cercano de terminar una historia de usuario ó mini-versión.

Al haber revisado las 3 principales metodología de desarrollo y en vista de las necesidades que se quieren suplir con el diseño del software del portafolio asignatura incorporando tanto los enfoques por competencia y por objetivos utilizaremos una metodología de desarrollo XP.

El énfasis que pone en XP en las personas se manifiesta en las diversas prácticas que indican que se deben dar más responsabilidades a los programadores para que estimen

su trabajo, puedan entender el diseño de todo el código producido, y mantengan una metáfora mediante la cual se nombra las clases y métodos de forma consistente-

Concretamente, esta metodología empieza en pequeño y añade funcionalidad con retroalimentación continua. Propicia que el manejo del cambio se convierta en parte sustancial del proceso. Define que el costo del cambio no depende de la fase ó etapa de desarrollo en que se encuentre. Como metodología no introduce funcionalidades antes que sean necesarias. El cliente ó usuario se convierte en miembro del equipo.

VI. DISEÑO DEL PORTAFOLIO ASIGNATURA BAJO EL ENFOQUE POR COMPETENCIAS

Las mejoras que se diseñarán permitirán incluir en la planificación de la asignatura elementos bajo el enfoque por competencia, creando un canal de comunicación entre el docente y el alumno; de manera que el alumno interactúe con la herramienta y se retroalimente con los indicadores de logros, unidades de aprendizaje, materiales didácticos y el modelo de evaluación.

6.1 Modelo arquitectónico.

El modelo arquitectónico consiste en diseñar un marco de trabajo estructural básico, en donde se definen aspectos generales del software y el modo en que la estructura ofrece una integridad conceptual al sistema. De modo simple, se puede considerar que está compuesto por la estructura jerárquica de los componentes y la manera en que dichos componentes interactúan entre sí.

La importancia de diseñar un modelo arquitectónico es que facilita la comunicación entre los diferentes participantes en el desarrollo del software, que ayuda a resaltar las decisiones de diseño que pueden tener un gran impacto en todo el proceso de desarrollo; además, aporta una visión de cómo se estructura el sistema y cómo trabajan sus componentes.

A continuación se presenta el diseño del modelo arquitectónico, según información obtenida de las aplicaciones que se desarrollan en la institución.

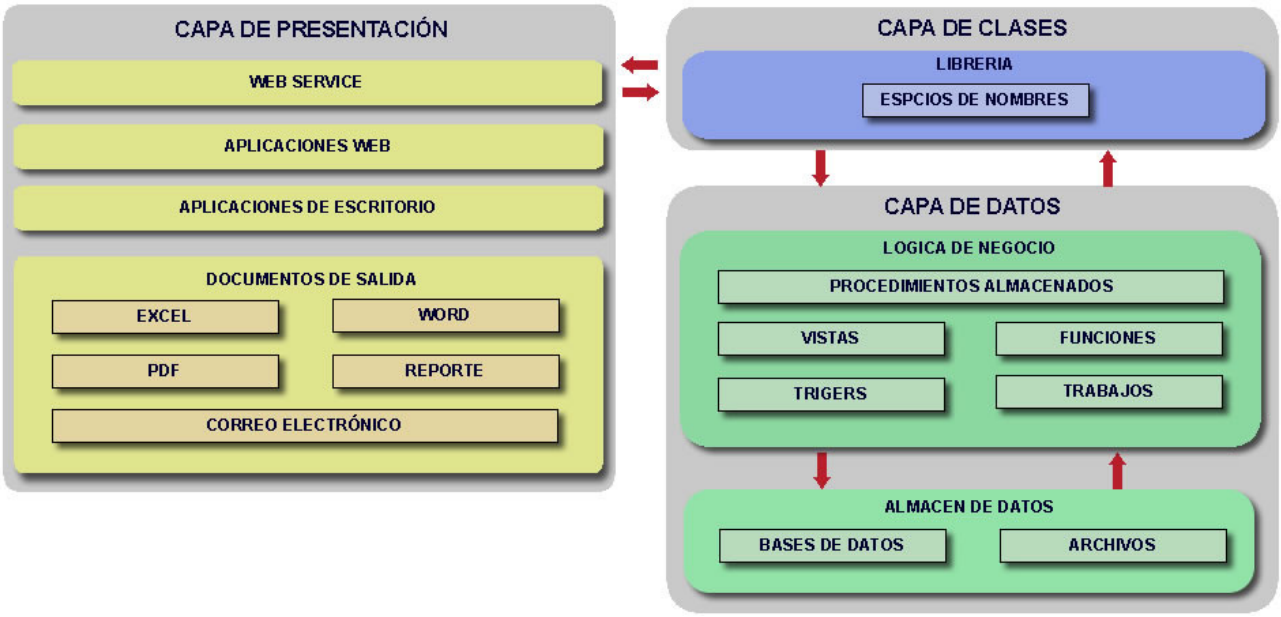


Figura 2: Modelo Arquitectónico

Terminología del Modelo Arquitectónico	
Término	**Descripción**
Capa	Hace referencia a la forma de cómo la solución es segmentada desde el punto de vista lógico.
Librería	Conjunto de estructuras de datos que conectan a un programa de software.
Espacios de Nombres	Es un conjunto de prefijos que sirven para identificar ó agrupar diferentes funcionalidades y evitar nombres ambiguos.
Lógica de negocios	Hace referencia a las tareas relacionadas en la forma de cómo los datos serán interpretados y procesados.

Tabla 2: Terminología del Modelo Arquitectónico

El Portafolio de Asignatura corresponde a una aplicación Web y como se puede apreciar en la figura 2 para acceder a los datos ó a los archivos, en la capa de presentación se hace uso de una librería de clases con un espacio de nombre definido, que se conecta a la capa de datos en donde se accede a la lógica de negocio de la aplicación.

6.2 Requerimientos Funcionales

Los requerimientos funcionales nos definen los servicios que debe proporcionar el sistema, de la manera en que éste debe reaccionar a las entradas de datos y de cómo se debe comportar en situaciones particulares. De forma general podemos decir que los requerimientos funcionales nos describen lo que el sistema debe hacer y en algunos casos, también pueden declarar explícitamente lo que el sistema no debe hacer.

Para identificar las necesidades de la solución, se programaron entrevistas a las personas involucradas en el proceso de elaboración del portafolio asignatura. Las personas seleccionadas fueron las siguientes:

- Humberto Flores: Vicerrector Académico
- Celina Rivera: Directora del departamento de investigación
- Amelia Sibrian: Coordinadora de programas de calidad académica.
- Tres docentes de la Facultad de Ingeniería, tres de la Facultad de Humanidades y dos de la Facultad de Estudios Tecnológicos, que hacen uso del portafolio de asignaturas.

Luego de todas las entrevistas se analizaron todas las grabaciones y se diseñó un modelo para representar los módulos que debe contemplar la herramienta.

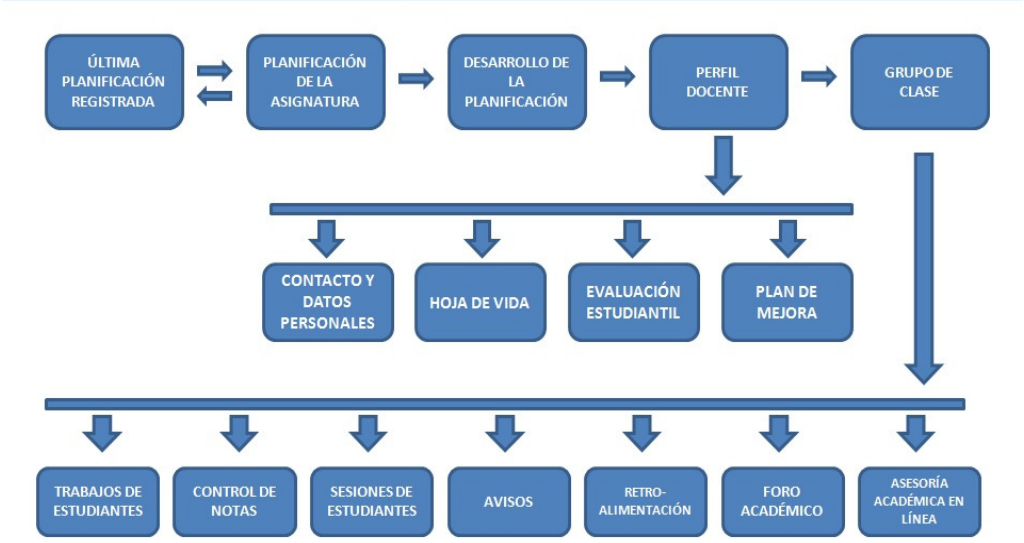


Figura 3: Modelo del Portafolio Asignatura

La herramienta que actualmente posee la Universidad, da soporte al enfoque por objetivos y se tiene planificado desarrollar los nuevos planes de estudio bajo el enfoque por competencia, de forma escalonada, para carreras específicas. El nuevo diseño del portafolio de asignatura, debe permitir llevar en paralelo los planes de estudio bajo el enfoque por competencia y las carreras que aún tienen un enfoque por objetivos.

6.3 Implementación de la metodología de desarrollo ágil

En base a la metodología de desarrollo ágil, **Extreme Programming**, que seleccionamos como apoyo para desarrollar la solución, se abordan las siguientes fases.

6.3.1 Fase I: Análisis del entorno.

En esta fase se desglosan las necesidades que proveerán el insumo para definir las funcionalidades, comportamientos y procesos que se incluirán en la solución.

Para resolver la dualidad de enfoques competencia-objetivos de los planes de estudio, se ha analizado la problemática y dado que la Universidad ya cuenta con una herramienta que da soporte al enfoque por objetivos, se diseñará un modulo independiente bajo el enfoque por competencia y se integrará al portal académico que posee la institución.

6.3.1.1 Módulos

Los módulos a los cuales les vamos a dar énfasis para aplicar la solución son:

Planificación de la Asignatura

La herramienta debe almacenar la última planificación registrada, con el objetivo de que se pueda copiar al siguiente módulo a impartir, una vez hechas las modificaciones, finalizar el proceso y esa información será el punto de partida para el desarrollo de la asignatura. Es decir, que todo lo que se planifique se copiará en los módulos correspondientes de trabajo. Por Ejemplo, temario de clases, actividades evaluativas y materiales didácticos de la asignatura.

La planificación de la asignatura debe enlazar los elementos que se muestran en la figura 4.

Figura 4: Elementos de la Planificación

La planificación de la asignatura no debe contener elementos parciales. Es decir, que la información debe estar relacionada. Por ejemplo, una evaluación debe pertenecer a una unidad de aprendizaje y ésta a un indicador de logro que abone a una competencia.

Unidad de Aprendizaje

La unidad de aprendizaje contiene el material didáctico que sustenta el contenido de la temática; cada unidad puede enlazar muchos materiales didácticos y se define la estrategia que se aplicará para el logro del aprendizaje.

Modelo de Evaluación

En este apartado se planifica todas las actividades de evaluación de la asignatura que se desarrollarán a lo largo del módulo; cada actividad asocia instrumentos que ayudan a definir los puntos a ser evaluados, como las listas de cotejo, rubricas ó guías de evaluación.

6.3.1.2 Historias de Usuario

A raíz de las entrevistas realizadas, se elaboraron las fichas propuestas por la metodología, para detallar la información de los usuarios e identificar las tareas.

A continuación se detallas las historias recopiladas, con sus respectivas tareas:

- **Historia 1: Competencia**
  - Tarea 1.1: Generar código de referencia
- **Historia 2: Indicador de Logro**
  - Tarea 2.1: Asociar indicador al código de referencia
- **Historia 3: Unidad de Aprendizaje**
  - Tarea 3.1: Registrar unidad de aprendizaje
  - Tarea 3.2: Crear mantenimiento de recursos
  - Tarea 3.3: Asociar Estrategia didáctica
- **Historia 4: Evaluación**
  - Tarea 4.1: Registrar Actividad de evaluación
  - Tarea 4.2: Crear mantenimiento de instrumentos de evaluación
  - Tarea 4.3: Crear vista de notas de un estudiante
  - Tarea 4.4: Crear vista de consolidado de notas del ciclo
  - Tarea 4.5: Crear Listas de notas de un curso

El detalle de las historias y tareas se muestran en el apéndice 1.

6.3.2 Fase II: Planificación

En esta fase se calendarizan las historias; tomando en cuenta una estimación de esfuerzo, para cada una de las tareas asociadas a la implementación de la historia; en base a tiempos estimados de programación.

6.3.2.1 Cronograma de actividades

Actividades	MES 1				MES 2			
	S1	S2	S3	S4	S1	S2	S3	S4
Análisis del Entorno								
Iteración 1: Competencia								
Iteración 2: Indicadores de Logro								
Iteración 3: Unidades de Aprendizaje								
Iteración 4: Modelo de Evaluación								
Modelado de Datos								
Diseño del prototipo								

Tabla 3: Cronograma de actividades

6.3.3 Fase III: Iteraciones

En esta fase se describen las iteraciones sobre las actividades de las tareas; cada una de las tareas es asignada a un responsable y desarrollada por una pareja de programadores.

A continuación se detallan cada uno de las iteraciones con sus respectivas tareas:

6.3.3.1 Iteración 1

Historia 1: Competencia

- ***Tarea 1.1*:** Generar código de referencia

1- Para identificar la competencia que será asociada, se utilizará el prefijo CMP y un correlativo de dos dígitos.
2- Registrar la descripción de la competencia.
3- Crear opción de edición y eliminación.

6.3.3.2 Iteración 2

Historia 2: Indicador de Logro

- **Tarea 2.1:** Asociar indicador al código de referencia

1- Para identificar el indicador de logro se utilizará el prefijo IL y un correlativo de dos dígitos.
2- Registrar la descripción del logro
3- Crear opción de edición y eliminación.

6.3.3.3 Iteración 3

Historia 3: Unidad de Aprendizaje

- **Tarea 3.1:** Registrar unidad de aprendizaje

1- Crear código de asociación con el prefijo UNI y un correlativo de dos dígitos
2- Guardar información

- **Tarea 3.2:** Crear mantenimiento de recursos

1- Encriptar nombre del archivo.
2- Guardar archivo en base de datos
3- Registrar descripción del material didáctico
4- Crear opción de eliminación.

- **Tarea 3.3:** Asociar Estrategia didáctica

1- Crear mantenimiento de estrategias didácticas.
2- Listar catálogo de estrategias

6.3.3.4 Iteración 4

Historia 4: Evaluación

- **Tarea 4.1:** Registrar Actividad de evaluación

1- Crear código de asociación con el prefijo EVA y un correlativo de dos dígitos
2- Guardar información.

- **Tarea 4.2:** Crear mantenimiento de instrumentos de evaluación

1- Crear modulo de búsqueda

2- Crear opciones de agregar, editar y eliminar

- **Tarea 4.3:** Crear vista de notas de un estudiante

1- Incluir gráficos que represente la nota ganada y la nota perdida.
2- Crear detalle de las notas de las actividades de una asignatura.

- **Tarea 4.4:** Crear vista de consolidado de notas del ciclo

1- Incluir gráficos que represente la nota ganada y la nota perdida.
2- Crear detalle de las notas globales de cada asignatura

- **Tarea 4.5:** Crear Listas de notas de un curso

1- Incluir gráficos que represente la nota ganada y la nota perdida.
2- Crear consulta de solvencia del estudiante

6.3.4 Fase IV: Diseño

En esta fase se describen los diferentes elementos utilizados para desarrollar la solución.

6.3.4.1 Modelado de datos

El modelado de datos es un conjunto de herramientas conceptuales que se utilizan para describir datos, sus relaciones, su significado y sus restricciones de consistencia. El modelado hace la pregunta “¿Qué?” en lugar de “¿Cómo?”, ésta última orientada al procesamiento de los datos; resulta en el descubrimiento y documentación de los recursos de datos del negocio. Es una tarea difícil, pero es una actividad cuya habilidad se adquiere con la experiencia.

6.3.4.2 Diagrama de clases

Para describir los tipos de datos y sus relaciones, utilizaremos un diagrama de clases de UML, por sus siglas en inglés, Unified Modeling Language. Es un lenguaje de modelado de software que ofrece un estándar para describir un modelo de forma gráfica, que a su vez documenta el sistema e incluye aspectos conceptuales del proceso de negocio.

En UML, una clase se representa con un rectángulo que posee tres divisiones, en donde:

- **Superior:** Contiene el nombre de la Clase
- **Intermedio:** Contiene los atributos que caracterizan a la Clase
- **Inferior:** Contiene las operaciones, que es la forma como interactúa el objeto con su entorno
- Existen tres tipos de clases:
- **Clases de Interfaz:** Se utilizan para modelar la interacción entre el sistema y sus actores. Representan pantallas, ventanas, paneles, etc.
- **Clases de Entidad:** Se utilizan para modelar información que posee una vida larga y que a menudo es persistente.
- **Clases de Control:** Representan coordinación, secuencia, transacciones y control de los objetos.

A continuación se muestra el diagrama de clases del proceso de planificación de la asignatura.

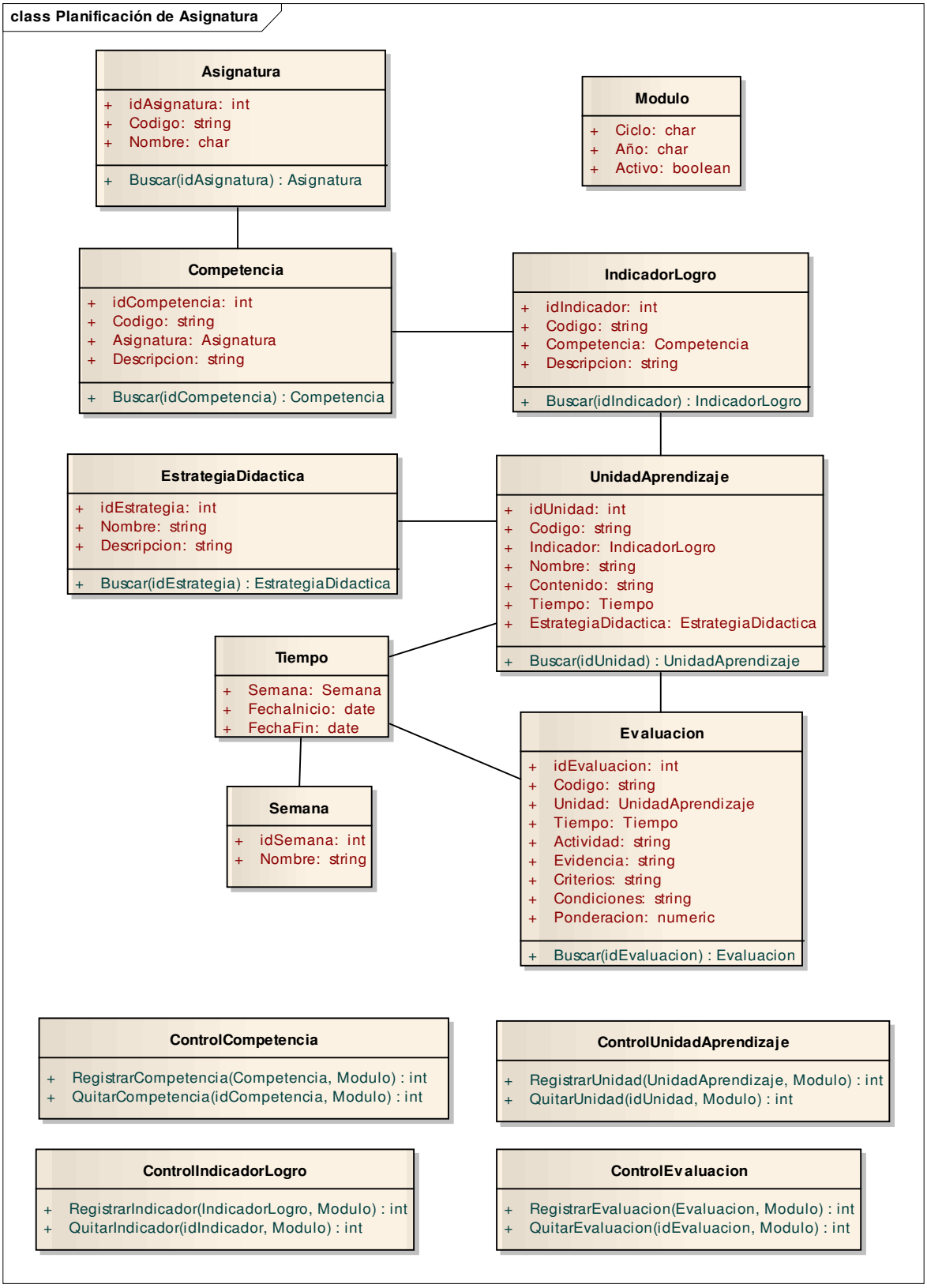


Figura 5: Diagrama Planificación de Asignatura

Como podemos observar en el diagrama de la figura 5, las clases de control permiten encapsular la funcionalidad, de manera que las clases entidades pueden heredarse en otros sub sistemas sin exponer sus funcionalidades, si así se requiere.

Para el control de calificaciones, creamos una clase de tipo pantalla para registrar la información; como se muestra en la figura 6

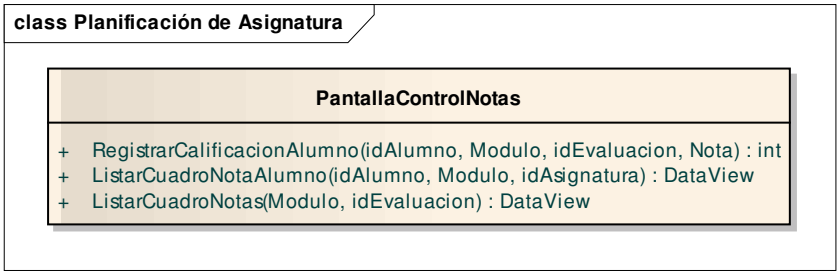


Figura 6: Pantalla Control Notas

Siguiendo el modelo arquitectónico de la figura 2, cada método representado en los diagramas de clases se implementará como un procedimiento almacenado de la base de datos, de tal manera que la lógica de negocio quede encapsulada en el procedimiento almacenado.

6.3.4.3 Diagrama entidad-relación

El diagrama entidad-relación (E-R) es un modelo que permite representar las entidades relevantes de un sistema de información así como sus interrelaciones y propiedades. La diferencia entre las clases de entidad consiste, en que las entidades representadas en un diagrama E-R hacen referencia a las tablas físicas que se crearán en la base de datos; una tabla entidad puede contener los atributos de una ó varias clases entidad. Los diagramas de clases se convierten en el insumo del diagrama E-R.
A continuación se presenta el diagrama E-R de la propuesta planteada.

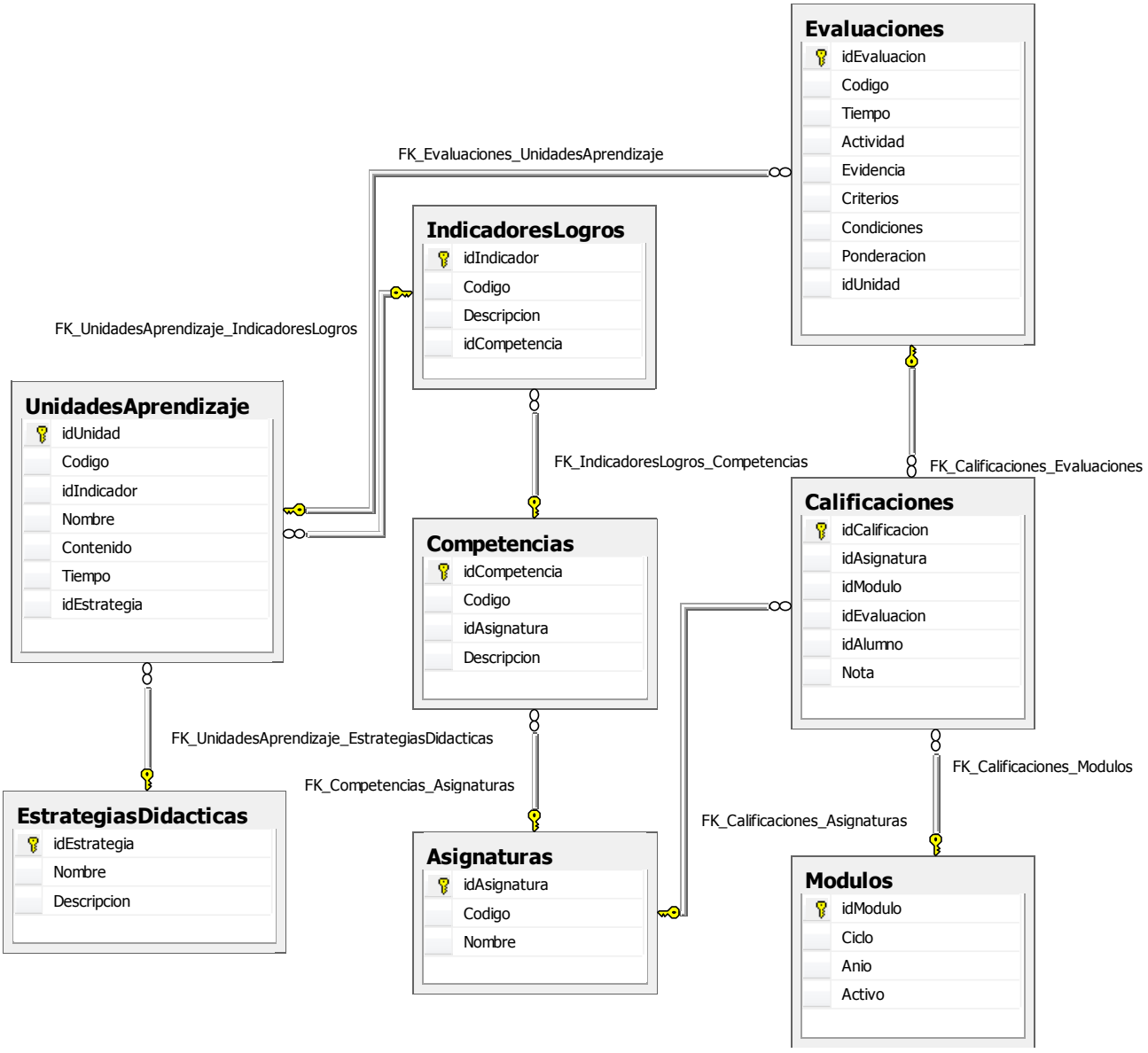


Figura 7: Diagrama E-R

6.3.4.4 Prototipo

Después de haber recopilado la información necesaria para la confección del prototipo del portafolio asignatura, se muestran a continuación las diferentes pantallas creadas para el mismo.

Para que los docentes puedan acceder al portafolio debemos de acceder a la pantalla principal.

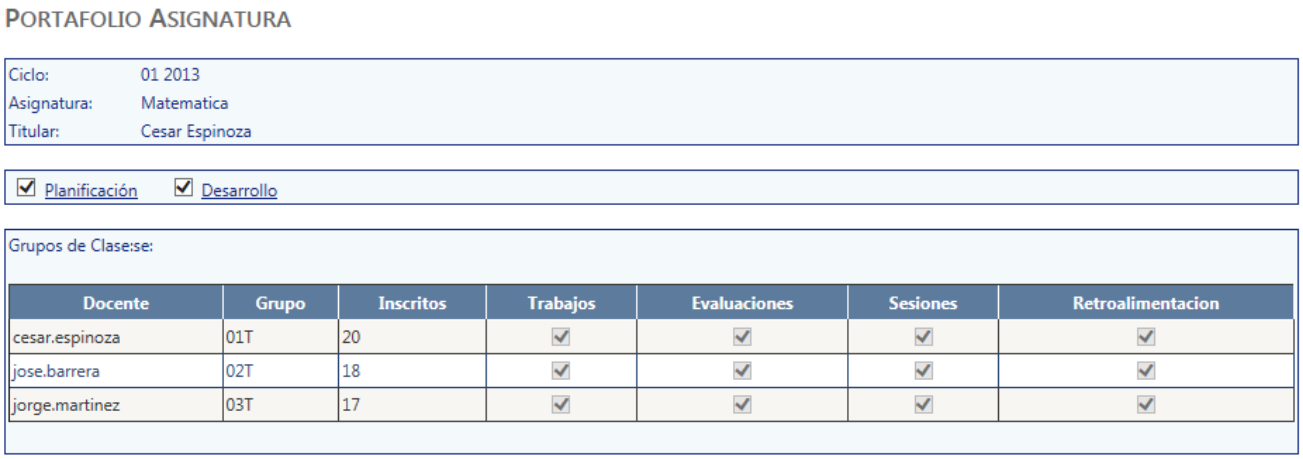
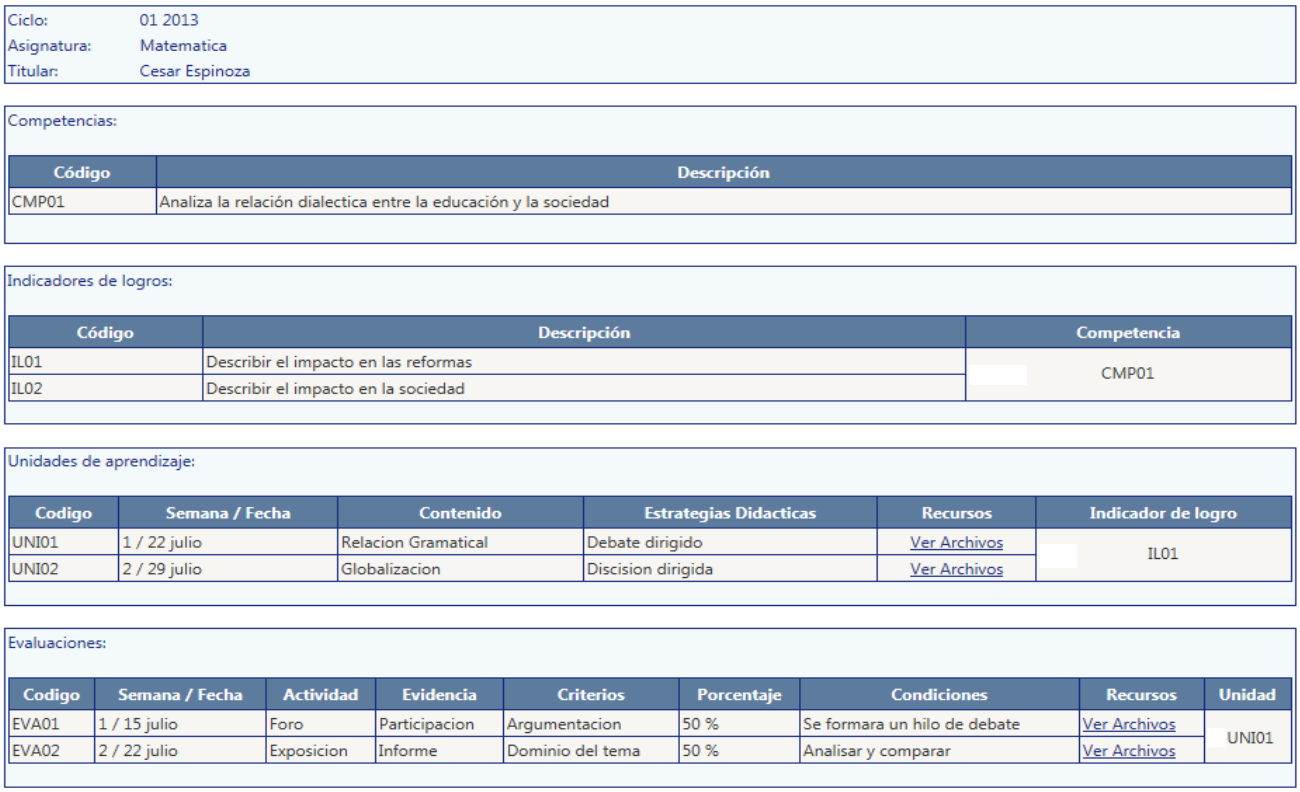


Figura 8: Pantalla principal del portafolio asignatura

En la figura 8 se muestra la pantalla principal en donde accederán los docentes; al dar clic sobre Planificación ó Desarrollo se mostrará el detalle de la planificación, como se muestra en la figura 9.



Ciclo: 01 2013
Asignatura: Matematica
Titular: Cesar Espinoza

Competencias:

Código	Descripción
CMP01	Analiza la relación dialectica entre la educación y la sociedad

Indicadores de logros:

Código	Descripción	Competencia
IL01	Describir el impacto en las reformas	CMP01
IL02	Describir el impacto en la sociedad	

Unidades de aprendizaje:

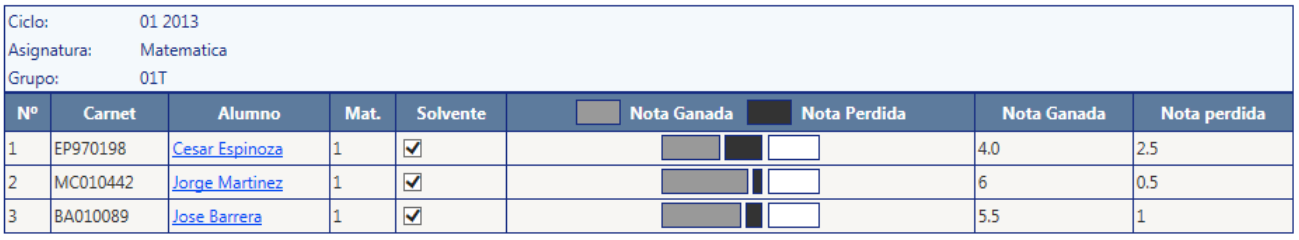
Codigo	Semana / Fecha	Contenido	Estrategias Didacticas	Recursos	Indicador de logro
UNI01	1 / 22 julio	Relacion Gramatical	Debate dirigido	Ver Archivos	IL01
UNI02	2 / 29 julio	Globalizacion	Discision dirigida	Ver Archivos	

Evaluaciones:

Codigo	Semana / Fecha	Actividad	Evidencia	Criterios	Porcentaje	Condiciones	Recursos	Unidad
EVA01	1 / 15 julio	Foro	Participacion	Argumentacion	50 %	Se formara un hilo de debate	Ver Archivos	UNI01
EVA02	2 / 22 julio	Exposicion	Informe	Dominio del tema	50 %	Analisar y comparar	Ver Archivos	

Figura 9: Detalle de la planificación de la asignatura

La planificación quedará solo de lectura y las modificaciones se harán en el desarrollo. A continuación veremos las pantallas para la gestión de notas.



Ciclo: 01 2013
Asignatura: Matematica
Grupo: 01T

Nº	Carnet	Alumno	Mat.	Solvente	Nota Ganada / Nota Perdida	Nota Ganada	Nota perdida
1	EP970198	Cesar Espinoza	1	☑		4.0	2.5
2	MC010442	Jorge Martinez	1	☑		6	0.5
3	BA010089	Jose Barrera	1	☑		5.5	1

Figura 10: Listado de estudiantes de un grupo de clases

Al dar clic sobre un estudiante, se presentará el consolidado de evaluaciones de ese estudiante, como se muestra en la figura 11.

Ciclo: 01 2013
Asignatura: Matematica
Grupo: 01T

Semana	Fechas	Actividad	Evidencia	Criterios	Porcentaje	Nota
Semana 1	del 29 de julio al 30 de agosto	Foro	Participación activa y propositiva	Argumentación	10 %	8
Semana 2	12 y 19 de septiembre	Exposición y debate	Realizar un informe	Dominio del tema	90%	6
				Nota Global	100%	6.2

Figura 11: Consolidado de evaluaciones de un estudiante.

Los estudiantes podrán ver su consolidado de notas de todo el ciclo, como se muestra en la figura 12.

Ciclo: 01 2013
Alumno: Jose Alexander Barrera [BA010087]
Carrera: Ingenieria en ciencias de la computación
CUM: 7.7

Código	Asignatura	Nota Ganada / Nota Perdida	Nota Ganada	Nota Perdida	Resultado
DID227	Digital		5	5	Reprobado
ECG229	Práctica Profesional		7.5	2.5	Aprobado
MAT501	Matematica I		8	2	Aprobado

Figura 12: Consolidado de notas del ciclo

Al final del ciclo, la nota ganada más la nota perdida será igual a la nota total, pero a lo largo del ciclo será un indicador para ver hasta qué punto se ha evaluado, como se muestra en la figura 10.

VII. CONCLUSIONES

1. Es necesario utilizar procedimientos de evaluación válidos, que evalúen lo que se quiere y pretende evaluar, en este caso competencias y no conocimientos.
2. Las metodologías de desarrollo Ágil proveen buenas prácticas que facilitan la comprensión de cómo hacer el desarrollo del software, entre el equipo de trabajo del proyecto.
3. Se logró determinar que XP, es la metodología ágil que más se adecua a las necesidades de este proyecto.

VIII. REFERENCIAS

[1] Metodologías Agiles en el Desarrollo de Software. Autores: José H. Canos, Patricio Letelier y Ma Carmen Penades. Disponible en línea: http://noqualityinside.com.ar/nqi/nqifiles/XP_Agil.pdf. Verificado, Junio 2013.

[2] Metodologías ágiles para el desarrollo de software: eXtreme Programming (XP), Autor: Patricio Letelier (2006). Disponible en línea: http://www.cyta.com.ar/ta0502/b_v5n2a1.htm. Verificado, Junio 2013.

[3] Balanceo de Metodologías Orientadas al Plan y Ágiles. Autor: Juan Gabardini, Lucas Campos (2004). Disponible en línea: http://www.eici.ucm.cl/Academicos/ygomez/descargas/Ing_Sw2/apuntes/Complementario%20al%20doc%20en%20ingles%20Using%20Risk.pdf. Verificado, Junio 2013.

[4] Procesos de desarrollo: RUP, XP y FDD. Autor: Alberto Molpeceres (2002). Disponible en línea: http://www.willydev.net/descargas/articulos/general/cualxpfddrup.pdf. Verificado, Junio 2013.

[5] Cuatro Enfoques metodológicos para el desarrollo de Software. Autor: Oiver Andres Perez (2011). Disponible en línea: http://biblioteca.uniminuto.edu/ojs/index.php/Inventum/article/view/9/9. Verificado, Junio 2013.

[6] Portafolio de Asignatura, Universidad Don Bosco. Disponible en línea: http://admacad.udb.edu.sv/PortalWeb/DocumentosPortal/Porta/Documentos/Doc11.pdf. Verificado, Junio 2013.

[7] Procesos Agiles para el desarrollo de Aplicaciones Web. Autores: Paloma Cáceres, Esperanza Marcos. Disponible en línea: http://dlsi.ua.es/webe01/articulos/s112.pdf. Verificado, Junio 2013.

[8] Tinoco Gómez, Oscar, Rosales López, Pedro Pablo y Salas Bacalla, Julio. Criterios de selección de metodologías de desarrollo de software. Ind. data, jul. 2010, vol.13, no.2, p.70-74. ISSN 1810-9993.

[9] Comparación y tendencias entre metodologías ágiles y formales. Metodología utilizada en el Centro de Informatización para la Gestión de Entidades. Autor: Universidad de las serie informativas (2011). Disponible en línea:

http://publicaciones.uci.cu/index.php/SC/article/view/484/469. Verificado, Julio 2013.

[10] Guía comparativa de metodologías ágiles. Autor: Pérez Pérez, María J.. Disponible en línea; http://cerro.cpd.uva.es/handle/10324/1495. Verificado, Junio 2013.

[11] Metodologías agiles RUP. Autor: Revolución (2009), Disponible en línea, http://ubingsoftware.blogspot.com/2009/06/rup-agil.html, Verificado, Julio 2013.

[12] Desarrollo e Implementación de una Interface de Usuario Interactiva y un Módulo de Entrenamiento para el Sistema de Simulación de Juego de Billar Arpool (Augmented Reality Pool) Utilizando Metodología de Desarrollo Ágil, Autores: Carlos Filiberto Alfaro Castro y Carmen Celia Morales Samayoa.(2011), , Universidad Don Bosco.

[13] Enterprise Architect (software), Disponible en línea: http://en.wikipedia.org/wiki/Enterprise_Architect_(software). Verificado, Julio 2013

[14] Así es Microsoft Visual Studio.Net, Madrid, España : MCGRAW HILL, 2001.

[15] SQL Server 2008, Francisco Charte Ojeda ,2009

[16] Desarrollo de Aplicaciones WEB con ASP.NET 2.0, Antonio Martín Sierra, 2007.

[17] Windows server 2008 : Guía del Administrador, William R. Stanek. 2008.

IX. APENDICE 1

Historia de Usuario	
Número: 1	Usuario: Vicerrector académico
Nombre Historia: Competencia	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alta
Puntos estimados: 2	Iteración asignada: 1
Descripción: Tiene como fin, entender el enfoque por competencia, como nace y como se relaciona con el plan de estudio de una carrera.	
Observaciones	

Tabla 1. Historia iteración 1

Tarea	
Número: 1.1	Número historia: 1
Nombre tarea: Generar código de referencia	
Tipo de tarea: Diseño	Puntos Estimados: 1
Fecha inicio:01/07/2013	Fecha fin:07/07/2013
Responsable: Desarrollador	
Descripción: Se lleva la secuencia de las competencias asociadas a una asignatura.	

Tabla 2. Tarea 1.1 Historia de iteración 1

Historia de Usuario	
Número: 2	Usuario: Coordinador
Nombre Historia: Indicador de logro	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alta
Puntos estimados: 1	Iteración asignada: 2
Descripción: Tiene como fin, entender como el indicador de logro se pondera con el plan de estudio.	
Observaciones	

Tabla 3. Historia iteración 2

Tarea	
Número: 1.1	Número historia: 2
Nombre tarea: Generar código de referencia	
Tipo de tarea: Diseño	Puntos Estimados: 1
Fecha inicio:08/07/2013	Fecha fin:14/07/2013
Responsable: Desarrollador	
Descripción: Se lleva la secuencia de las competencias asociadas a una asignatura.	

Tabla 4. Tarea 2.1 Historia de iteración 2

Historia de Usuario	
Número: 3	Usuario: Director de escuela
Nombre Historia: Unidad de Aprendizaje	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alta
Puntos estimados: 4	Iteración asignada: 3
Descripción: Describir todos los elementos con que se relaciona una unidad de aprendizaje	
Observaciones	

Tabla 5. Historia iteración 3

Tarea	
Número: 3.1	Número historia: 3
Nombre tarea: Registrar unidad de aprendizaje	
Tipo de tarea: Diseño	Puntos Estimados: 1
Fecha inicio:15/07/2013	Fecha fin:21/07/2013
Responsable: Desarrollador	
Descripción: Definir tipos de datos y sus características	

Tabla 6. Tarea 3.1 Historia de iteración 3

Tarea	
Número: 3.2	Número historia: 3
Nombre tarea: Crear Mantenimiento de recursos	
Tipo de tarea: Diseño	Puntos Estimados: 1
Fecha inicio:17/07/2013	Fecha fin:21/07/2013
Responsable: Equipo de desarrollo	
Descripción: Definir los tipos de archivos permitidos así como su peso en bytes	

Tabla 7. Tarea 3.2 Historia de iteración 3

Tarea	
Número: 3.3	Número historia: 3
Nombre tarea: Asociar Estrategia didáctica	
Tipo de tarea: Diseño	Puntos Estimados: 1
Fecha inicio:17/07/2013	Fecha fin:21/07/2013
Responsable: Desarrollador	
Descripción: Diseñar la búsqueda de una estrategia didáctica desde el mantenimiento de unidades de aprendizaje	

Tabla 8. Tarea 3.3 Historia de iteración 3

Historia de Usuario	
Número: 4	Usuario: Docente
Nombre Historia: Evaluación	
Prioridad en negocio: Alta	Riesgo en desarrollo: Alta
Puntos estimados: 5	Iteración asignada: 4
Descripción: Describir todos los elementos relacionados con las actividades de evaluación	
Observaciones	

Tabla 9. Historia de iteración 4

Tarea	
Número: 4.1	Número historia: 4
Nombre tarea: Registrar Actividad de Evaluación	
Tipo de tarea: Diseño	Puntos Estimados: 1
Fecha inicio:22/07/2013	Fecha fin:28/07/2013
Responsable: Desarrollador	
Descripción: Definir tipos de datos y sus características.	

Tabla 10. Tarea 4.1 Historia de iteración 4

Tarea	
Número: 4.2	Número historia: 4
Nombre tarea: Crear mantenimiento de instrumentos de evaluación	
Tipo de tarea: Diseño	Puntos Estimados: 1
Fecha inicio:22/07/2013	Fecha fin:28/07/2013
Responsable: Desarrollador	
Descripción: Diseñar búsqueda, vista, registro, edición y eliminación de la información que contiene el instrumento de evaluación	

Tabla 11. Tarea 4.2 Historia de iteración 4

Tarea	
Número: 4.3	Número historia: 4
Nombre tarea: Crear vista de notas de un estudiante	
Tipo de tarea: Diseño	Puntos Estimados: 1
Fecha inicio:26/07/2013	Fecha fin:28/07/2013
Responsable: Desarrollador	
Descripción: Diseñar la vista de datos y crear diseño de gráfico	

Tabla 12. Tarea 4.3 Historia de iteración 4

Tarea	
Número: 4.4	Número historia: 4
Nombre tarea: Crear vista de consolidado de notas de un ciclo	
Tipo de tarea: Diseño	Puntos Estimados: 1
Fecha inicio:27/07/2013	Fecha fin:28/07/2013
Responsable: Desarrollador	
Descripción: Diseñar la vista de datos y crear diseño de gráfico	

Tabla 13. Tarea 4.4 Historia de iteración 4

Tarea	
Número: 4.5	Número historia: 4
Nombre tarea: Crear lista de notas de un curso	
Tipo de tarea: Diseño	Puntos Estimados: 1
Fecha inicio:27/07/2013	Fecha fin:28/07/2013
Responsable: Desarrollador	
Descripción: Diseñar la vista de datos y crear diseño de gráfico	

Tabla 14. Tarea 4.5 Historia de iteración 4