Enfoque UTE
E-ISSN: 1390-6542
enfoque@ute.edu.ec
Universidad Tecnológica Equinoccial
Ecuador

Fiallos Ordoñez, Ángel
Mejoramiento en la productividad de software por la adaptación de un marco de desarrollo ágil.
Enfoque UTE, vol. 6, núm. 2, abril-junio, 2015, pp. 117-134
Universidad Tecnológica Equinoccial

Disponible en: http://www.redalyc.org/articulo.oa?id=572260847009

Cómo citar el artículo
Número completo
Más información del artículo
Página de la revista en redalyc.org

Sistema de Información Científica
Red de Revistas Científicas de América Latina, el Caribe, España y Portugal
Proyecto académico sin fines de lucro, desarrollado bajo la iniciativa de acceso abierto

Enfoque UTE, V.6-N.2, Jun.2015, pp.117 - 134
http://ingenieria.ute.edu.ec/enfoqueute/
e-ISSN: 1390-6542 / p-ISSN: 1390-9663
Recibido (Received): 2015/02/20
Aceptado (Accepted): 2015/06/17
CC BY-NC-ND 3.0

Mejoramiento en la productividad de software por la adaptación de un marco de desarrollo ágil.

(Improving productivity software through the adaptation of an agile development framework)

Ángel Fiallos Ordoñez[1]

Resumen:
El presente trabajo de investigación plantea que el uso de las metodologías ágiles en conjunto con herramientas de software libre permite mejorar la productividad, reducir costos y optimizar recursos en la etapa de desarrollo informático, y contribuye a mejorar la satisfacción del usuario con la implementación de software de excelente calidad. A continuación se realiza un diagnóstico de las variables consideradas de importancia para el éxito en la ejecución o desarrollo de proyectos informáticos y que están asociadas al uso de las metodologías tradicionales y ágiles para el desarrollo de software.

Palabras clave: MDA; metodologías; ágiles; software; desarrollo

Abstract:
The current research suggests that using of agile methodologies in conjunction with open source software tools can improve productivity, reduce costs and optimize resources in the process of software development, and helps improve user satisfaction due to implementation of excellent quality software. The following analysis shows the most important variables for the successful implementation of IT development projects and their relation with the use of traditional and agile software development methodologies.

Keywords: MDA, methodologies; agile; software; development

1. Introducción

1.1. Situación Actual

En el mundo actual las organizaciones son cada vez más dependientes de la tecnología, la demanda de software es cada vez más elevada, y el proceso de desarrollo del software es más costoso y susceptible a cambios en las especificaciones en cualquiera de sus etapas. Los equipos de desarrollo deben ser cada vez más competitivos y eficientes en cuanto a recursos y presupuesto, por lo que deben adoptar marcos de trabajo y metodologías que permitan que el desarrollo de aplicaciones se realice en menor tiempo, de una manera organizada y con menos costos, facilitando el mantenimiento y las posteriores actualizaciones de dicho software.

En el proceso de construcción del software, las metodologías de desarrollo son un factor crítico, encontrándose vigentes para su aplicación, las denominadas metodologías tradicionales y ágiles. Standish Group en su reporte "Manifiesto del Caos", ha determinado estadísticamente que los

[1] Universidad de Especialidades Espíritu Santo, Guayas, Samborondón, Ecuador (afiallos@uees.edu.ec)

proyectos ejecutados con metodologías ágiles, tuvieron más probabilidad de culminar con éxito que aquellos proyectos en los que se utilizaron los métodos tradicionales (en cascada) y que los cambios en los objetivos de los proyectos en los que aplicaron las metodologías ágiles, disminuyeron con relación a los proyectos manejados con métodos tradicionales.

Este reporte establece como factor de éxito que el software debe ser construido en pequeños pasos iterativos, con equipos de trabajo pequeños y focalizados. En cambio los métodos tradicionales, asumen que es posible predecir y detallar a largo plazo las variables del proyecto, en lo que respecta la planificación, del alcance, costo, tiempo, recursos y calidad del producto final. Esta situación determina que el desarrollo de software sea poco adaptable al cambio.

Como apoyo a las metodologías ágiles de desarrollo de software, surge la especificación Model Driven Development (MDA) para dar facilidad en el entendimiento de sistemas complejos, mientras proporciona una abstracción del sistema. Esta visión abstracta del sistema es representada a través de estándares de modelado establecidos por el Object Management Group (OMG) como son: Unified Modeling Language (UML), Meta-Object Facility (MOF) y XML Metadata Interchange (XMI).

La especificación MDA no intenta reinventar otro proceso de desarrollo de software pero agrupa el conocimiento de los últimos veinte años en la ingeniería de software para varias arquitecturas. De hecho el enfoque de MDA se basa principalmente en el Model Driven Development (MDD) (Stahl, 2006) y lo que sugiere que un sistema debe ser modelado primero antes de que pueda ser transformado para adaptarse a la plataforma final en la que se ejecutará.

Este enfoque cambia el concepto de la metodología tradicional, centrándose en la definición y el diseño de los componentes, permitiendo que la codificación se pueda realizar de forma automática y en el lenguaje de programación a seleccionar, con el fin de reducir esfuerzo y dar mayor flexibilidad a las etapas del ciclo de vida de software.

1.2. Objetivo General.

Demostrar mediante el prototipo de una aplicación web y el análisis de métricas de ingeniería de software, que el uso de las metodologías agiles reduce el esfuerzo, costo, tiempo, líneas de código y aumenta la productividad en el proceso de desarrollo de software

1.3. Objetivos Específicos

Implementar un proceso desarrollo ágil en el marco de una arquitectura dirigida a modelos (MDA) para la construcción de un prototipo web, mostrando paso a paso todas las etapas necesarias hasta la implementación del producto.

Aplicar el modelo matemático COCOMO 2.O para medir el esfuerzo, tiempo y productividad en el desarrollo tradicional de software, tomando como referencia las funcionalidades implementadas en el prototipo web

Aplicar la métrica de punto de función para la medición del número de líneas de código en el desarrollo tradicional de software, tomando como referencia las funcionalidades implementadas en el prototipo web y el lenguaje de programación seleccionado.

Usar herramientas de código abierto para la configuración de un marco de trabajo para el desarrollo e implementación del prototipo web.

1.4. Hipótesis

El presente estudio de caso, plantea que el uso de metodologías de desarrollo ágil, en combinación con herramientas de software libre, aumentará la productividad, así también disminuirá los tiempos, los costos y mejorará la flexibilidad en las etapas de desarrollo de software.

A través de la construcción paso a paso de un prototipo funcional configurado en el entorno de una arquitectura dirigida por modelos, se realizará la demostración de la viabilidad técnica que ofrecen las herramientas de software libre para el efecto, así también se presentará la funcionalidad de la aplicación, estadísticas del sector software y un análisis cuantitativo de datos y variables relacionadas a los procesos de desarrollo de software, tales como tiempos, esfuerzos, calidad, costos y líneas de código manuales.

2. Metodología

2.1. Diseño de la investigación

La modalidad de investigación del presente trabajo considera la elaboración y desarrollo de una propuesta de modelo operativo viable para solucionar la problemática actual con respecto a las metodologías tradicionales.

Se contempla la medición de las variables indicadas en la Tabla 1 al culminar la etapa de implementación del prototipo web bajo el contexto de las metodologías ágiles. Para realizar la comparación respectiva de los valores con respecto al uso de metodología tradicional, se ha considerado tomar como referencia el cálculo estimado de las variables, mediante la aplicación de las siguientes métricas de ingeniería de software: punto de función y el modelo constructivo de costo – COCOMO.

La comparación de cada una de las variables medidas durante la construcción del prototipo web, con los valores estimados por las métricas de software, ha permitido evaluar y calcular los indicadores a ser presentados.

Tabla 1. Variables e Indicadores del estudio de caso

Variable	Concepto	Indicador variación
Esfuerzo	Horas de trabajo realizado por programador para una determinada actividad.	Horas de trabajo (estimadas e históricas) en desarrollo prototipo con metodologías tradicionales - Horas de trabajo empleadas en desarrollo prototipo con metodologías ágiles
Costo	Costos de mano de obra por programador en un tiempo determinado.	Costo por horas de trabajo estimado en desarrollo prototipo con metodologías tradicionales - Costo por horas empleadas en desarrollo prototipo con metodologías ágiles
Tiempo	Duración de las actividades de desarrollo de software	cantidad de tiempo estimado en desarrollo de prototipo con metodologías tradicionales - Cantidad de tiempo empleado en desarrollo de prototipo con metodologías ágiles
Productividad	Cantidad de trabajo realizado por programador en un tiempo determinado.	Trabajo en puntos objeto estimado en desarrollo de prototipo con metodologías tradicionales - Trabajo en puntos objeto realizado en el desarrollo de prototipo con metodologías ágiles
Líneas de código	Conteo de líneas de código generadas de forma automática y líneas codificadas manualmente	Total de líneas de código manuales estimadas con metodologías tradicionales - Líneas de código generadas automáticamente en el desarrollo de prototipo.

2.2. Población y Muestra

Teniendo como antecedente, el estudio del mercado de Hardware y Software del Ecuador para el año 2012 realizado por la Asociación Ecuatoriana de Software (AESOFT), existen 633 empresas que participan en el sector de “Programación, Informática, Consultoría de Informática y actividades conexas”. El 86% de las empresas está en la ciudad de Quito y Guayaquil, con el 49% y 37% respectivamente.

Del total indicado, las empresas que se dedican exclusivamente al desarrollo de Software y están registradas en el catálogo de empresas de Software de la AESOFT (AESOFT C. , 2013), suman

132 empresas a nivel nacional, número que será determinada como la población del presente estudio.

La muestra a tomar es de 31 empresas del sector Software, sobre la cual se realizó encuestas a propietarios, desarrolladores y líderes de proyecto que respondieron de forma positiva a su llenado.

3. Resultados

Para el desarrollo de la aplicación web, dentro del contexto de la arquitectura dirigida a modelos (MDA), se proponen las siguientes tecnologías de software libre:

- **AndroMDA.-** AndroMDA es un framework[2] de código extensible asociado a la especificación MDA. La herramienta toma cualquier modelo (usualmente modelos UML guardados en formato XMI producidos por alguna herramienta case), que en combinación con plugins de la propia herramienta (Plantillas y bibliotecas de traducción), produce componentes personalizados, es decir, se puede generar componentes en un gran número de lenguajes entre los que se encuentra: Java, .Net, HTML, PHP, etc. (AndroMDA.org, 2012).

- **Eclipse.-** Es una plataforma de software de código abierto independiente de otras plataformas. Esta plataforma, típicamente ha sido usada para desarrollar entornos integrados de desarrollo. Sin embargo, también se puede usar para otros tipos de aplicaciones cliente.

- **JBoss.-** es un servidor de aplicaciones Java EE de código abierto implementado en Java puro. Al estar basado en Java. JBoss puede ser utilizado en cualquier sistema operativo para el que esté disponible la máquina virtual de Java.

- **Herramienta de Modelado.-** Existen varias herramientas para el modelado UML, compatibles con el IDE Eclipse. Debe seleccionarse una herramienta con entorno gráfico de los diagramas UML, que cumpla con las especificaciones de OMG.

- **MySQL. -** La única limitación que establece AndroMDA respecto a la base de datos es que esta sea compatible con Hibernate. La base de datos se comunica con la capa superior mediante registros de la base de datos.

- **Hibernate.**- Es un Framework Java para el manejo de bases de datos, que permite el mapeo de objetos relacionales a objetos.

[2] Framework, se refiere un conjunto estandarizado de conceptos, prácticas y criterios para enfocar un tipo de problemática particular que sirve como referencia, para enfrentar y resolver nuevos problemas de índole similar.

El prototipo funcional se implementa bajo la plataforma J2EE, siguiendo una arquitectura de N niveles o capas distribuidas, basándose ampliamente en componentes de software modulares y que serán ejecutados en un servidor de aplicaciones. Bajo esta premisa, el marco ágil de trabajo propuesto tiene el marco de arquitectura de aplicaciones que se presenta en la Figura 1.

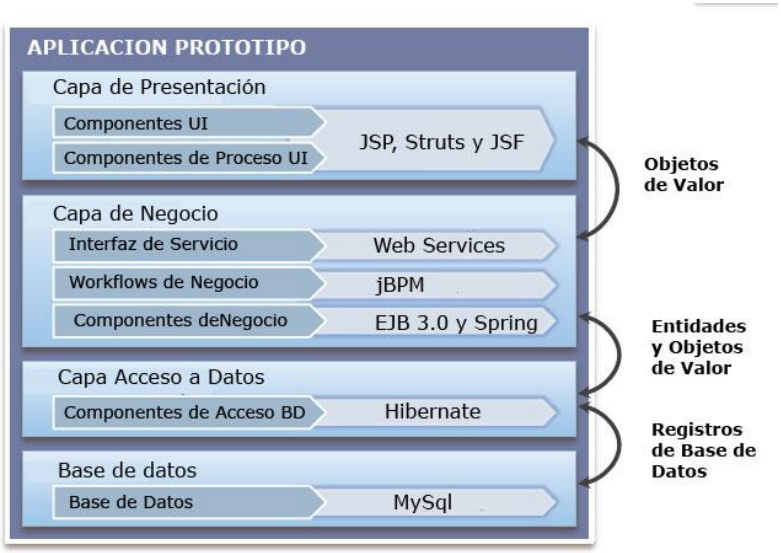


Figura. 1. Modelo de Arquitectura de aplicaciones J2EE**.**

3.1. Levantamiento de Información

El módulo web funcional a implementar consiste en una aplicación J2EE, para el registro de fichas de actividades de los empleados de un determinado negocio o empresa. Los empleados a medida que trabajan en las distintas tareas encomendadas, registran la fecha, hora de inicio y hora de finalización de labores.

Para el desarrollo del aplicativo, se emplearán los conceptos de la especificación MDA para las conversiones a código fuente de los modelos UML elaborados en la etapa de diseño, por medio de la ejecución del plugin de AndroMDA dentro del contexto de la arquitectura J2EE establecida para el efecto. Las reglas de negocio muy particulares deben codificarse manualmente con el mínimo esfuerzo posible.

El proceso de desarrollo en base al estándar MDA, inicia con la solicitud de nuevos requerimientos de usuario de una manera informal, cuya funcionalidad es modelada en diagramas UML (Diagrama de clases, casos de uso y diagramas de actividad), con el uso de herramientas como Magic Draw.

Una vez completados los modelos en programas XML, se ejecutan los componentes y plantillas de AndroMDA, que generan código fuente Java de acuerdo a los cartuchos con la tecnología Java seleccionado. Así también se generan los archivos de configuración y los scripts para la creación de tablas en la base de datos MySQL, configurada para el uso exclusivo del aplicativo.

El desarrollador puede tomar los programas Java autogenerados por el plugin de AndroMDA y sus plantillas para codificar de manera manual las reglas de negocio particulares requeridas por el usuario. Estos cambios deben ser el único esfuerzo de programación a realizarse por parte del desarrollador al aplicarse este marco de trabajo. La aplicación una vez culminado su desarrollo, puede ser desplegada en un servidor de aplicaciones que soporte la plataforma JEE, y pueda ser accedida desde un navegador web, a modo de intranet, según el esquema de la Figura 2.

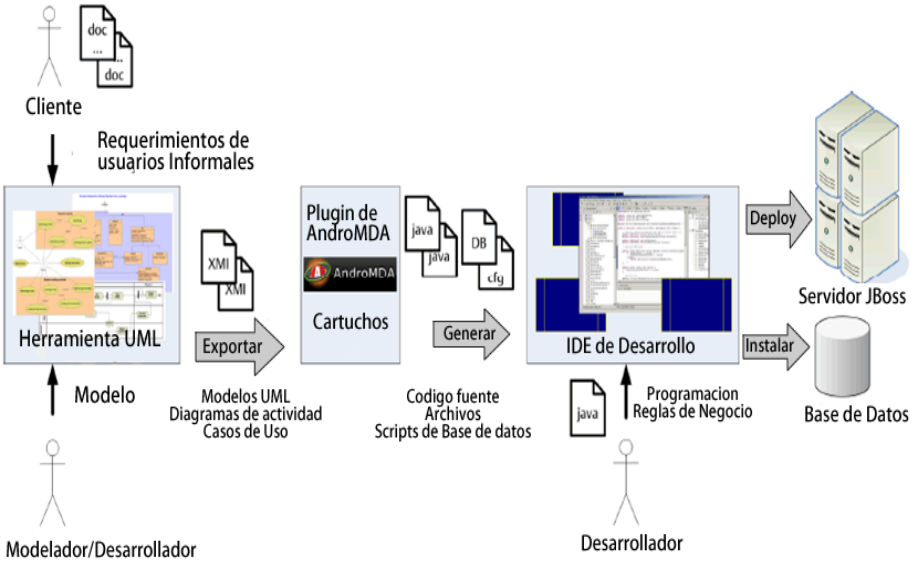


Figura 2. Proceso de desarrollo con estándar MDA.

3.2. Análisis

En resumen el prototipo funcional para el registro de fichas de actividades, debe contemplar las opciones disponibles al usuario presentadas en la Tabla 2.

Tabla 2: Detalle de opciones del aplicativo “Registro de Fichas de actividades”

Pantalla	Concepto
Pantalla de Logon	Pantalla que valida el usuario y contraseña de los usuarios registrados
Pantalla de Inicio	Pantalla de menú con las opciones del aplicativo
Mantenimiento de tareas	Pantalla para el registro, modificación, consulta y eliminación de tareas de actividades
Mantenimiento de usuarios	Pantalla para el registro, modificación, consulta y eliminación de usuarios del aplicativo
Búsqueda de Fichas	Consulta de fichas de actividades por varios criterios
Mantenimiento de Fichas y detalle de fichas	Pantalla para el registro, modificación, consulta y eliminación de fichas de actividades
Reporte de Fichas	Reporte de Fichas de actividades, a ser descargado en PDF, en formato XML o MS Excel.

Para el diseño UML del proyecto, se generó un modelo UML vacío con la herramienta de modelado MagicDraw en el archivo XML. Con la herramienta de modelado UML Magic Draw, se diseñaron las entidades, sus atributos, y el respectivo diagrama de clases con sus relaciones. En la Figura 3, se muestra la elaboración del diseño del diagrama de clases, las subclases y sus relaciones de asociación.

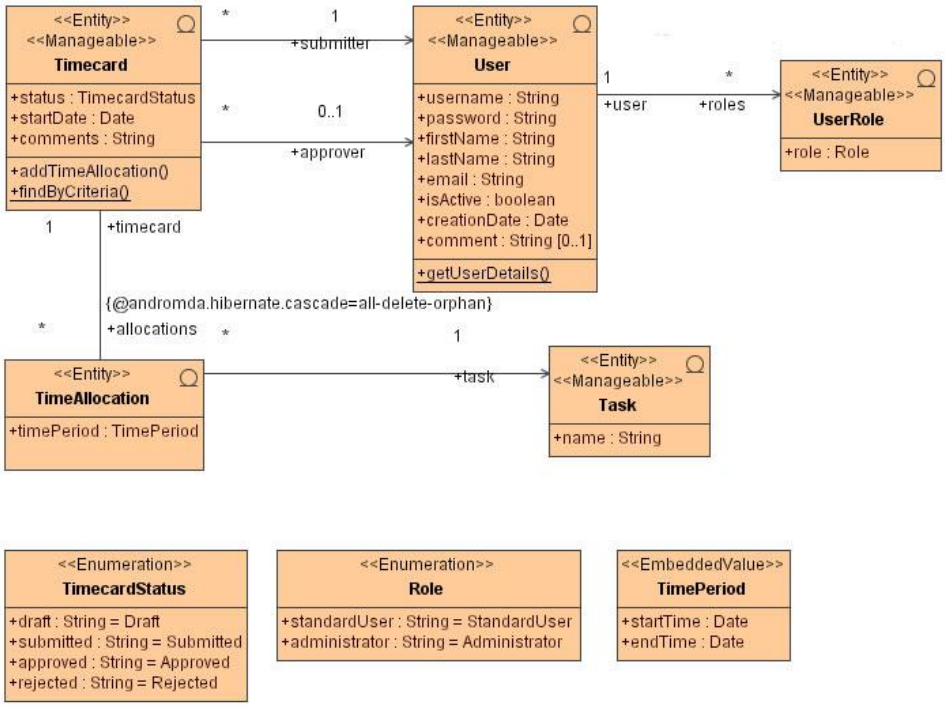


Figura 3. Diseño UML en Herramienta MagicDraw

3.3. Desarrollo

Una vez diseñado el diagrama de clases en la herramienta de diseño UML y los métodos de negocio asociados a cada una de las clases, se generó el archivo Maven: pom.xml. Este archivo contiene información sobre los detalles del proyecto y de configuración utilizados por Maven para construirlo. Dentro de este archivo se configuró la utilización del plugin de AndroMDA en la versión 3.4 y de los cartuchos con las tecnologías JEE a implementar.

En la configuración del archivo Maven: pom.xml, se habilitó el uso de los cartuchos AndroMDA para implementar la aplicación con los frameworks: Java Server Pages y Java Struts en la capa de presentación, Spring para la capa de negocio, Hibernate 3.5.6 para el mapeo de objetos relacionales y el acceso a datos y MySQL para la capa de datos.

Una vez lista la configuración, se valida y se ejecuta el archivo pom.xml por medio de la herramienta Maven, con el siguiente comando:

mvn org.andromda.maven.plugins:andromdapp-maven-plugin:3.4:generate

Al final, como se puede ver en la Figura 4, se espera el mensaje: "BUILD SUCCESSFUL" y la creación de un archivo EAR que empaqueta todas los módulos generados, para su despliegue (deploy) en el servidor JBOSS. En la figura siguiente se encuentra la interfaz de ejecución que culmina con la generación del archivo EAR desplegable de la aplicación.

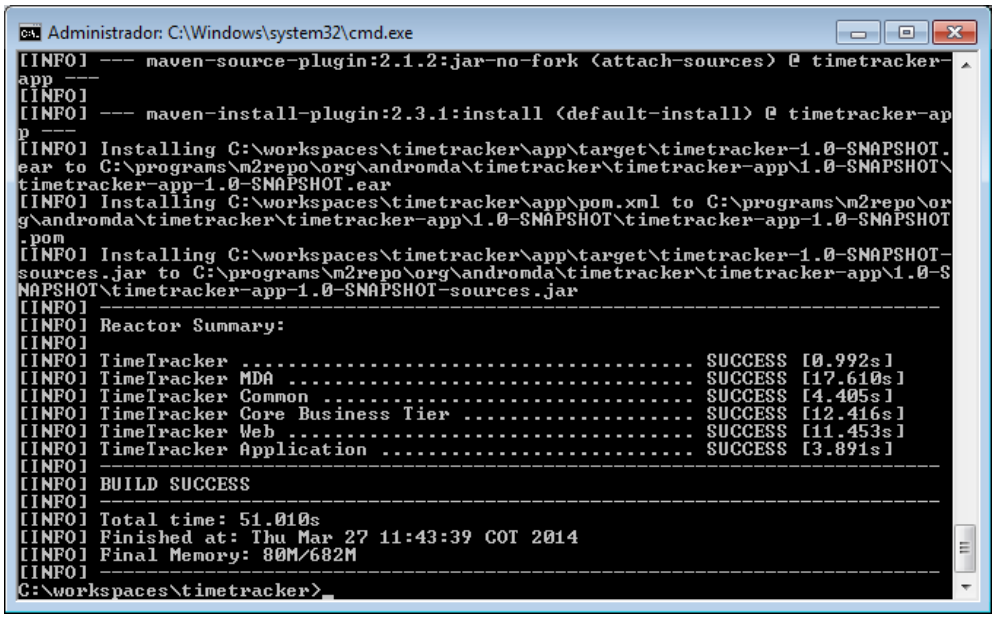


Figura 4. Ejecución del plugin AndroMDA para la generación de archivo EAR de la aplicación.

Cuando se ejecuta el plugin AndroMDA y existen objetos de tipo clases modelados con el tipo <<entidad>>, se realiza automáticamente el mapeo de objetos relacionales por medio del componente Hibernate, así como la generación de los programas de acceso a datos (DAO).

Las clases entidades que en el modelo se les asignó el atributo “manageable”, permitirán implementar los denominados archivos CRUD (Create, Read, Update y Delete), que permitirán usuario final la manipulación directa de los datos asociados a esa entidad. Es decir se generará automáticamente el código y las pantallas para insertar, editar y eliminar instancias de estas entidades.

A través de un plugin de Maven adicional, se genera y se ejecuta el script asociado con la creación de estas clases de persistencia en sus correspondientes tablas.

Aplicando el marco de trabajo descrito se consigue que los desarrolladores puedan centrarse en los problemas de alto nivel y reducir el tiempo de la programación de código repetitivo. En resumen, se registró que el tiempo de desarrollo fue de 144 horas con el esfuerzo de un desarrollador, incluyendo la curva de aprendizaje. Para la aplicación prototipo, se generó de manera automática los siguientes objetos:

- 5 tablas
- 65 archivos Java
- 5 archivos XML de mapeo relacional.
- 16 páginas JSP
- 4 archivos JAR

Al culminar el proceso de generación automática de código fuente, se determinaron las métricas presentadas en la Tabla 3:

Tabla 3. Métricas de la construcción del prototipo: "Registro de Fichas de actividades"

Métricas	Valores
Esfuerzo	0,9 personas mes
Tiempo	144 horas
Productividad	26,6 /0,9*1 = 29,55 Puntos Objeto / Tiempo * Esfuerzo
Líneas de código contabilizadas	7745 líneas de código

3.4. Pruebas

Enfoque ágil de desarrollo basado en pruebas

Siguiendo los lineamientos de las metodologías ágiles, el prototipo fue realizándose en varias iteraciones, implementando funcionalidades distintas en cada una de ellas. Cada interacción incluye un corte vertical completo de la aplicación a partir de las pantallas de la capa de presentación hasta la comunicación con la capa de base de datos.

También se adoptó el enfoque ágil del desarrollo basado en pruebas (TDD), el cual sugiere escribir la prueba antes de empezar a codificar. En el detalle de la prueba para cada uno de los casos de uso descritos en el modelo UML, se definen los requisitos y escenarios requeridos por el usuario. Una vez realizado el diseño, se ejecutan los plugins de AndroMDA para generar el código, se codifica las reglas de negocio, se despliega la aplicación y se prueba el caso de uso.

Proceso ágil de pruebas funcionales.

A continuación se detallan las actividades realizadas por cada caso de negocio desarrollado.

- Generar las entidades, interfaces de servicios y casos de uso a partir del modelo UML

- Escribir una prueba y testear el caso de uso de negocio. El primer intento de ejecutar esta prueba, obviamente fallará porque el servicio no se ha implementado todavía.
- Implementar la lógica de la capa de negocio o servicio de cada caso de uso a través de la generación automática de código y de la codificación manual.
- Repetir la prueba hasta asegurarse de que fue superada.
- Registrar conformidad del usuario de la prueba exitosa y continuar con las demás.

Una vez realizados los ajustes a las pruebas con errores, se debe ejecutar nuevamente los cartuchos androMDA para procesar los archivos del modelo PSM y generar automáticamente la nueva versión del código fuente. El mismo que deberá estar disponible para la siguiente prueba funcional.

3.5. Implementación

Servidor de aplicaciones.

Para el despliegue de la aplicación, se debe preparar y configurar las instancias del servidor de aplicaciones JBoss 5.1. Este servidor está basado en Java, y puede ser utilizado en cualquier sistema operativo para el que esté disponible la máquina virtual de Java. Una vez levantados los servicios se debe acceder al panel administrativo para configurar el respectivo datasources de la aplicación.

Puesta en ambiente de producción.

Una vez generado el archivo EAR, en el que se encuentra empaquetada todas las librerías y programas JAVA de la aplicación, se procede a copiar la misma en la carpeta \jboss-5.1\server\default. El servidor JBoss, despliega la aplicación como se ve en la Figura 5, y si la misma no tiene errores, estará disponible para el acceso.

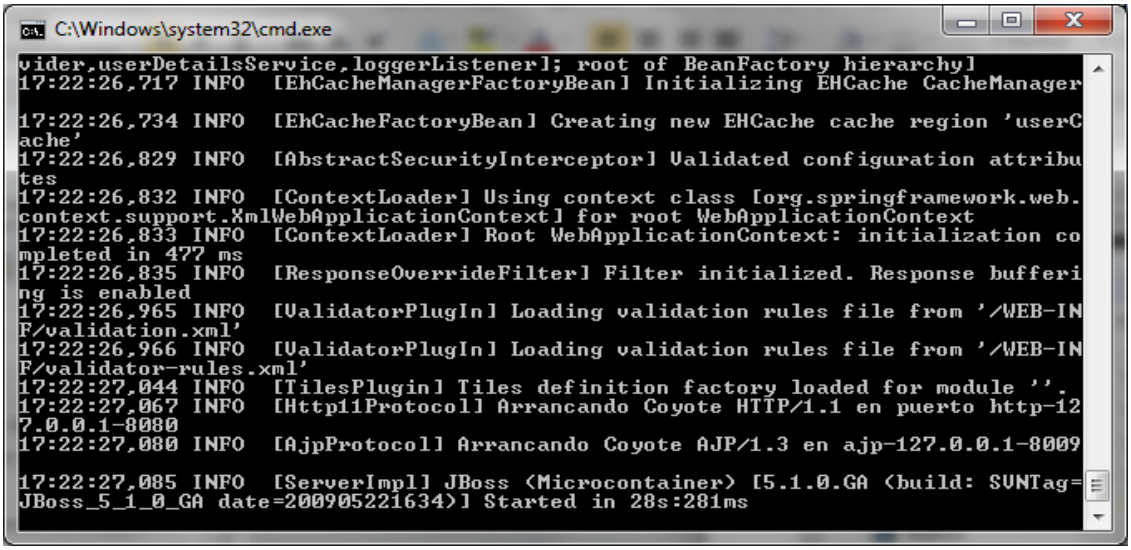


Figura No. 5: Despliegue del archivo EAR en el servidor JBoss 5.1.1.

Se puede acceder al prototipo a través del navegador de la preferencia, accediendo a la ruta y puerto: http://localhost:8080/, una vez se despliegue la aplicación empaquetada con éxito, en el contexto del servidor JBoss. En caso de no levantarse el servicio por errores al momento de ejecutar el proceso de deploy, se debe revisar los archivos logs del servidor JBoss.

Un resumen visual del proceso de desarrollo, implementación y ejecución del prototipo funcional, se encuentra publicado en la siguiente dirección:
https://www.youtube.com/watch?v=kY_z2kRxPPs

4. Discusión

4.1. Estimación de variables con métricas de ingeniería de software.

Una vez analizado el entorno de las empresas desarrolladoras de Software, los resultados determinaron que durante la ejecución de desarrollos de software, consideran a las variables: calidad, tiempo, esfuerzo y costo, como muy importantes para el éxito en los proyectos.

Con el fin de evaluar el comportamiento de las variables indicadas, se procede a realizar en primera instancia la estimación de las variables cuantitativas: esfuerzo, tiempo, costo y tamaño (en líneas de código) necesarios para el desarrollo de los puntos objeto de los casos de uso planteados para el prototipo funcional: “Registro de Fichas de actividades”, con el supuesto de que se ejecute bajo el método tradicional de cascada pura.

Para la estimación del esfuerzo, tiempo, costos y tamaño, se emplearan las métricas de ingeniería de software: Cocomo 2 y Puntos de Función. Con los resultados teóricos del esfuerzo calculado, se realizará la comparación respectiva con respecto a los valores que fueron consecuencia de la implementación del prototipo funcional.

4.2. Evaluación con modelo Cocomo 2.

Para la estimación cuantitativa de las variables: esfuerzo, tiempo y costo, bajo los métodos tradicionales, se empleó el modelo Cocomo 2.0, aplicado a la funcionalidad del prototipo presentado. Los datos se presentan en la Tabla 4.

Para el cálculo de esfuerzo, tiempo y costo de desarrollo, se asume para realiza la estimación que se cuenta con desarrolladores con experiencia media y que no cuentan con herramientas CASE.

Aplicando la relación No. Puntos Objeto / Tiempo * Esfuerzo, se determina que un desarrollador pudiera codificar 8,52 puntos objetos por mes, en la aplicación Java

Tabla 4. Cálculo de Puntos Objeto del Prototipo

Componentes del Sistema	**Grado**	**Tipo**	**Peso**
Pantalla de Ingreso	Simple	Pantalla	2
Aprobación de Tarjetas de Registro de Horario	Medio	Pantalla	5
Mantenimiento de actividades	Simple	Pantalla	2
Mantenimiento de usuarios	Medio	Pantalla	5
Consulta especifica de Tarjetas de Registro	Simple	Pantalla	2
Mantenimiento de tarjetas de registro	Difícil	Pantalla	8
Reporte de Tarjetas de Registro	Medio	Reporte	4

NOP = (Puntos Objetos) x (100 - %Reutilización) /100	
% Reutilización =	5
NOP (Número de puntos objeto)=	26,60

Maduración y capacidad 4 (Sin Herramientas Case)
Experiencia de desarrolladores 13 (Experiencia media)
Promedio 8,5
EPM = NOP/Promedio
EPM = 26,6/8,5
EPM = 3,12 Esfuerzo de Personas por mes.
Costo = 3,12 personas * $1.600,00 = $5007,00
Tiempo = 1 mes

4.3. Evaluación del tamaño de software con el Método Punto de Función

En la Tabla 5 se establece la tabla de valores, estimados por la complejidad de la tecnología J2EE.

4.4. Estimación de líneas de código

Para estimar las líneas de código, se deben multiplicar los puntos de función en la Tabla 6, por la tabla de factores por lenguajes que se muestra en la Figura 6. Para el presente estudio se utilizó como referencia la tabla QSM para la multiplicación respectiva por el número de puntos de función calculado (QSM, 2014). El resultado es 8731 líneas de código (SLOC)

Tabla 5. Cálculo de peso promedio por categoría y ponderación de puntos de función.

	Archivos e interfaces internas	**Interfaces Externas**	**Entradas**	**Salidas**	**Interacciones /Consultas**
Valor calculado	7	7	4	5	4
Complejidad:	Simple	Promedio	Promedio	Promedio	Promedio

Componente	**Archivos e interfaces internas**	**Interfaces Externas**	**Entradas**	**Salidas**	**Interacciones/ Consultas**
Pantalla de Ingreso	0	0	1	0	1
Aprobación Tarjetas de Registro de Horario	0	0	1	1	2
Mantenimiento de actividades	0	0	3	1	1
Mantenimiento de usuarios	0	0	3	1	3
Consulta especifica de Tarjetas de Registro	0	0	0	1	3
Mantenimiento de tarjetas de registro	0	0	3	1	3
Reporte de Tarjetas de Registro	0	0	0	1	3

Tabla 6. Calculo de puntos de función por categoría de componente.

Tipo	**Cantidad**	**Promedio**	**Puntos de Función**
Archivos e interfaces internas	0	0	0
Interfaces Externas	0	5,4	0
Entradas	11	4	44
Salidas	6	5	30
Interacciones/ Consultas	16	4	64
Total Puntos de Función			138

Se calcula el CAF (Factor de Ajuste de Complejidad)
CAF = 0.65 + 0.01 * N
CAF = 0.65 + 0.01 * 46
CAF = 1,11
AFP = FP * CAF
AFP = 138 * 1,11
AFP = 153,18

Lenguaje	Lineas de codigo / Puntos de Funcion			
--	Pro	Med	Mín	Máx
ASP	56	50	32	106
Assembler	209	203	91	320
C	148	107	22	704
C++	59	53	20	178
C#	58	59	51	66
FoxPro	36	35	34	38
J2EE (Java)	*57*	*50*	*50*	*67*
Java	55	53	9	214
JavaScript	54	55	45	63
JSP	59	-	-	-
.NET	60	60	60	60
Perl	57	57	45	60
PL/SQL	47	39	16	78

Figura 6: Factores QSM por lenguaje de Programación.

4.5. Evaluación de Resultados

Se presenta a continuación en la Tabla 7, el resumen referente al análisis cuantitativo de las variables calculadas e históricas, comparadas con la información registrada en las etapas de construcción del prototipo de software y el resultado de los indicadores establecidos:

Tabla No. 7: Resumen de análisis de variables cuantitativas y resultado de indicadores

Variable	Metodología Tradicional	Especificación MDA (ágil)	Indicador variación	Observación
Esfuerzo	3,12 Desarrolladores - mes	0,90 desarrolladores – mes	Reducción del 71,15% en etapa de desarrollo y del 28,46%, del esfuerzo total del proyecto	Reducción del esfuerzo en etapa de desarrollo, al realizarse el mismo trabajo estimado de tres recursos con el método tradicional con un solo recurso.
Costo	$4.992,00	$1.440,00	Reducción del 71,15% en costo de salarios en etapa de desarrollo.	Reducción de costos por ser una variable directamente proporcional al esfuerzo. No existe costo por concepto de licencias de software.
Tiempo	. 160 horas	144 horas	Reducción del 10% .	El empleo del TDD no disminuye de manera significativa el tiempo de desarrollo, pero reduce el riesgo de cambios
Productividad	8,52 Puntos Objetos – Mes por Programador	29,55 Puntos Objetos – Mes por Programador	Mejora en productividad del 72,52%	Mayor trabajo realizado con menos recursos por la generación automática de código.
Líneas de código	8731 líneas	7745 líneas	Reducción del 11,29%	No considera líneas de código del modelo UML

5. Conclusiones y Recomendaciones

Se ha demostrado en el análisis de resultados que los datos experimentales de las variables: esfuerzo, costo, tiempo y líneas de código registrados en la construcción del prototipo funcional aplicado en un marco ágil de trabajo propuesto en combinación con herramientas de software libre, tuvieron una reducción del 71,75%, 71,75%, 10%, y 11,29% respectivamente con relación a los datos calculados en base a métricas de estimación de software, en el contexto de las metodologías tradicionales. La productividad en la etapa de desarrollo mejoró en un 72,52%.

El prototipo web muestra cada una de las etapas de desarrollo de la especificación MDA, en el contexto de un marco de desarrollo ágil y una arquitectura de alto rendimiento de capas distribuidas, cuyo despliegue fue realizado en un servidor de aplicaciones habilitado para el efecto y con herramientas de código abierto utilizables en múltiples dispositivos, plataformas y sistemas operativos.

Entre las limitaciones de la presente investigación, se puede citar a la curva de aprendizaje del proceso de investigación, con respecto a la configuración de la especificación MDA y su integración con los modelos UML generados en la etapa de diseño.

Se pudo identificar que el uso de software libre se encuentra muy afianzado, sobre todo en proyectos de gobierno. Las empresas consideran que presenta ventajas con respecto al software propietario, aunque la gran mayoría no trabaja en exclusividad con un determinado tipo de software, sino que depende del proyecto.

Para finalizar, en el resumen de las encuestas realizadas se puede concluir que en nuestro país, las metodologías ágiles tales como SCRUM, Kanban, Programación extrema, tienen mayor aceptación en el sector de desarrollo de software que las metodologías tradicionales, aunque no de manera exclusiva sino conforme a la necesidad de los proyectos en curso.

Las empresas consideran que el principal problema durante el proceso de desarrollo de Software es el cambio en las especificaciones funcionales, seguido de la planificación muy optimista y que el uso de las metodologías agiles les permitirá tener una mejor comunicación con los usuarios y una mayor flexibilidad en los cambios requeridos. Esta información abre las posibilidades de aceptación a nivel empresarial del marco de trabajo propuesto.

Bibliografía

AESOFT. (2012). *Estudio de Mercado de Hardware y Software en Ecuador*. Obtenido de http://www.revistalideres.ec/tecnologia/Estudio-mercado-software-hardware-Ecuador_LIDFIL20120620_0001.pdf

Ambler, S. (2012). *Agile Model Driven Development (AMDD): The Key to Scaling Agile Software Development.* Obtenido de http://www.agilemodeling.com/essays/amdd.htm

AndroMDA.org. (2012). *What is AndroMDA?* Obtenido de http://www.andromda.org/whatisit.html

Beck, K. (2001). *Manifesto for agile software development.* Obtenido de http://agilemanifesto.org

Brown, A. (Febrero de 2004). *An introduction to Model Driven Architecture.* Obtenido de http://www.ibm.com/developerworks/rational/library/3100.html

Flatter, D. (2008). Obtenido de Impact of Model-Driven Standards.OMG: http://www.omg.org/mda/mda_files/mda_1.7_cleanformat.pdf

Group, S. (2012). *Chaos Manifesto.*

IFPUG. (2001). *Function Points IFPUG 4.3, Quick Reference.* Obtenido de http://www.totalmetrics.com/__data/assets/pdf_file/0012/2046/R087_Function-Points-IFPUG-4.3-Quick-Reference.pdf

Kleppe, A. (2003). *MDA Explained: The Model Driven Architecture.* Addison-Wesley, ISBN-13: 978-0321194428, Pag 321.

Mellor, S. (2002). *Executable UML, A Foundation for Model-Driven Architecture.* Addison Wesley, ISBN 0-201-74804-5, Pag 45-47

OMG, I. (2011). Meta Object Facility (MOF) 2.0 Query/View/Transformation Specification - Object Management Group.

QSM. (2014). *QSM Function Points Languages Table*. Obtenido de http://www.qsm.com/resources/function-point-languages-table

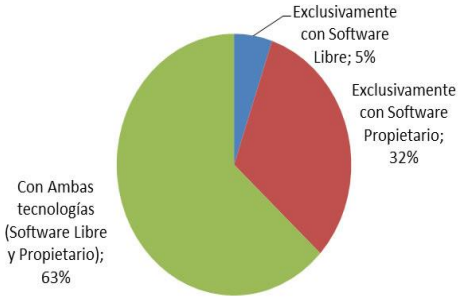
Stahl, T. y. (2006). *Model-Driven Software Development: Technology, Engineering, Management.* Wiley, ISBN 978-0470025703, Pag 101-102.

Anexo: Estadísticas sobre el uso de metodologías ágiles y software libre.

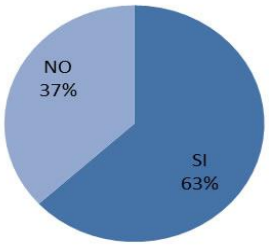
Fueron realizadas encuestas a líderes de proyecto, analistas y programadores de empresas dedicadas a consultorías y desarrollo de Software, en el número estimado en la muestra. Entre las empresas que se pueden destacar están las siguientes: Gennassis S.A, CADCOMP S.A, NDeveloper, Corlasosa, AuroraSI, Cobiscorp, IXION Technology, Latinoamericana de Software, Devsu Software, MAINT Cia Ltda, MIWebWorks, Imagetech, Consulting & Management SAC, SIC (Servicios Informáticos), ECS - Evolution Consulting Services, KCingle CUPIA del Ecuador, entre otras. La tabulación de las encuestas, determino los siguientes resultados.

Estadísticas del uso Metodologías Ágiles y Software Libre

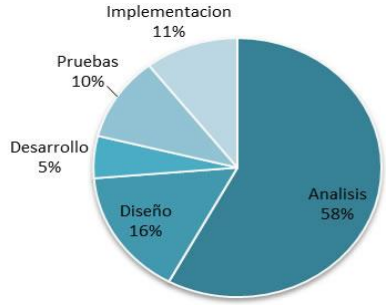
Tipo de Software utilizado en empresas consultoras de software.



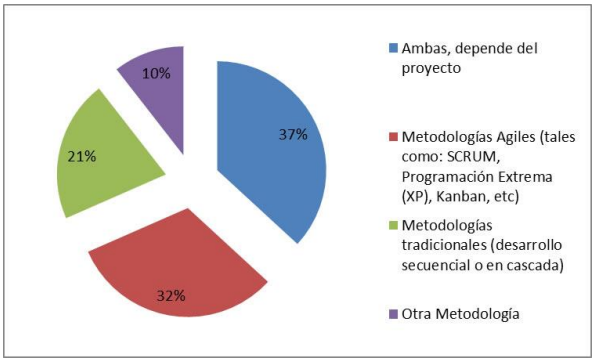
Opinión de empresas consultoras de software sobre ventajas del Software libre



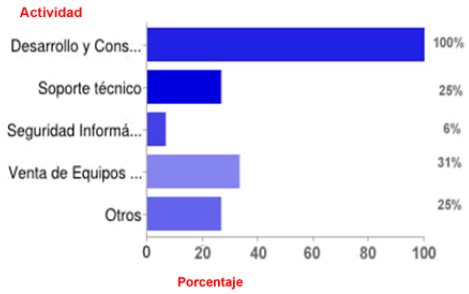
Etapas críticas en proyectos de desarrollo de software



Metodología utilizada en proyectos de software.



Actividades de empresas desarrolladoras de Software.



Principales inconvenientes durante proyectos de desarrollo de software.