SCRUM Y RUP:
COMPARATIVA Y PROPUESTA METODOLÓGICA

Scrum and RUP:
Comparative and methodological approach

Juan Sebastián Villanueva

Ingeniero de Sistemas, Universidad Distrital Francisco José de Caldas, Colombia

,jsvq85@gmail.com.

María Milena Siachoque

Ingeniero de Sistemas, Universidad Distrital Francisco José de Caldas, Colombia,

milena.siachoque@gmail.com

Tipo de artículo: revisión

Recibido: 2013-11-25

Aceptado: 2014-04-05

Resumen

Las metodologías de desarrollo agiles han ganado gran interés en la investigación y practica en el campo de la ingeniería de software. Particularmente, los desarrolladores sienten que Scrum es una metodología de desarrollo más compatible a sus prácticas actuales de trabajo. Por otra parte, perciben que el uso de Scrum entrega varias ventajas relativas por la dedicación del esfuerzo exclusivamente al desarrollo. Sin embargo, también se identificaron posibles barreras a la aceptación desde que los desarrolladores requieren la disciplina y rigidez propias de las metodologías tradicionales de desarrollo de software.

Palabras clave— Desarrollo de software, RUP, SCRUM, Arquitectura de software.

ABSTRACT

Agile development methodologies have gained great interest in research and practice at software engineer area. Particularly, developers felt scrum to be more compatible to their actual working practices. Moreover, they perceived the use of scrum to deliver numerous relative advantages concerning the dedication exclusively to the development effort. However, we also identified possible barriers to acceptance since developers require discipline and rigidity properly of traditional development methodologies.

Key Word — Software Developments, RUP, SCRUM, Software Architecture

1 .INTRODUCCIÓN

La ingeniería de software y su constante búsqueda de calidad que desde sus inicios con el modelo en cascada pasando por la aplicación del modelos evolutivos como el espiral, hasta la aplicación de procesos agiles en el desarrollo de software como Agile o Scrum, buscan mejorar el uso del conocimiento científico en el diseño y construcción de programas de computadora y la documentación asociada requerida para desarrollar, operar y mantenerlos [1].

Cada una de estas metodologías, las cuales tienen diferentes enfoques para la captura de requerimientos y el proceso de desarrollo del sistema software, algunas de ellas basadas en analizar y documentar rigurosamente las especificaciones del sistema, para luego realizar un desarrollo y posteriormente efectuar las pruebas.

Otros métodos proponen centrarse en la organización del equipo de trabajo, incluir al cliente activamente y en arrojar resultados satisfactorios más rápidamente. Sea cual fuere la metodología es conveniente saber que éstas se eligen e implementan de acuerdo a la naturaleza del proyecto, llegando incluso a combinarse entre sí para lograr mejores resultados.

La ingeniería de software desde su enfoque práctico describe seis buenas prácticas que son recomendables en el desarrollo de software en aras de la búsqueda de la calidad:

- Desarrollo iterativo
- gestión de requisitos
- Desarrollo basado en componentes
- Modelado visual, UML
- Verificación continua de la calidad
- Control de cambios de software [2]

El objetivo de éste trabajo es analizar dos metodologías propuestas (Scrum y RUP), caracterizarlas y proporcionar un guía que permita entender de una manera general el enfoque de cada una, así como sus ventajas y desventajas, para luego generar una propuesta hibrida con lo mejor de las dos, permitiendo a los interesados en el artículo tener un nuevo enfoque en el proceso de desarrollo de software.

El contenido del artículo es el siguiente: En la sección II se presenta la metodología Rational Unified Process (RUP) de IBM que a pesar de su rigurosidad y alta documentación, es un estándar altamente usado en el desarrollo de software. En la sección III se presenta la disciplina SCRUM basada en la rapidez y flexibilidad de métodos de desarrollos agiles ampliamente usados en la industria de productos comerciales. En la sección IV se presenta una comparación de ambas metodologías teniendo en cuenta sus ventajas y

desventajas. En la sección V se da a conocer la propuesta hibrida, tomando para la construcción de proyectos las mejores prácticas de las dos metodologías contrastadas. Finalmente en la sección VI se presentan las conclusiones de este trabajo.

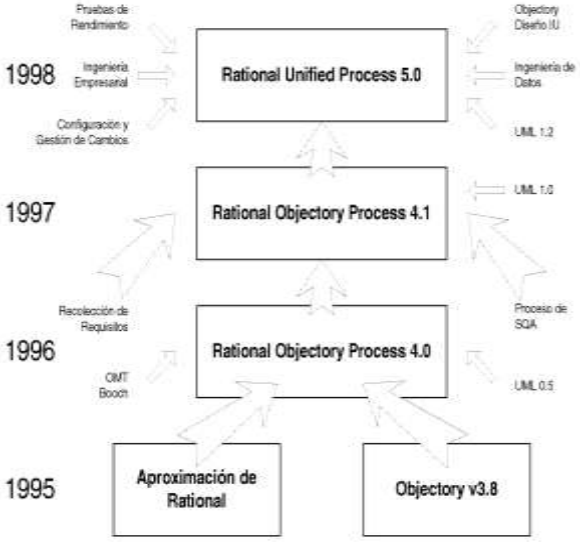
2. CONTENIDO

2.1 RUP (Rational Unified Process)

Es una metodología de desarrollo de software concebida en 1998 por Ivar Jacobson, Grady Booch y James Rumbaugh. RUP nació de la implementación del UML (Unified Modeling Language), como el lenguaje estándar para documentar y del UP (Unified Process) [3].

En la figura 1 se puede ilustrar la evolución del proceso desde sus inicios.

Figura 1 : Evolución de RUP



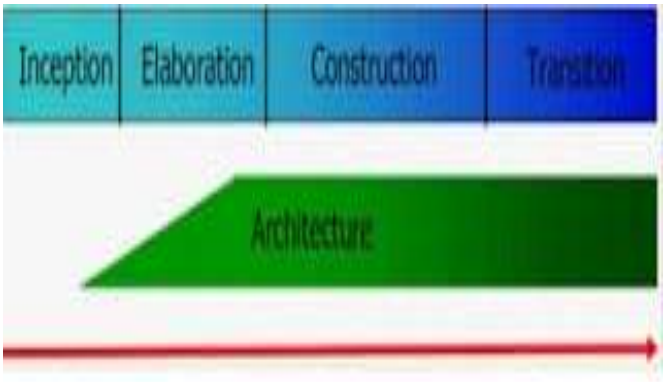
Fuente: Orígenes históricos del Proceso Unificado de Modelado, en línea, http://gmodulo.sourceforge.net/docs/html/manual/figures/rup02.png

RUP es un proceso dirigido por los casos de uso, centrado en la arquitectura, iterativo e incremental [4], 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 el servicio que el usuario u otro ente externo, usualmente denominado actor, requiere del sistema. Muestra la secuencia completa de interacciones entre el actor y el sistema.

b) Centrado en la arquitectura: Ilustra las diferentes vistas del sistema a desarrollar, que corresponden a los modelos dinámicos y estáticos. La arquitectura del software es importante para comprender el sistema como un todo y a la vez en sus distintas partes [5], sirve para organizar el desarrollo, fomentar la reutilización de componentes y hacerlo evolucionar [6], es decir, generar software mantenible, con la posibilidad de agregarle nueva funcionalidad generando aplicativos flexibles y escalables.

Figura 2: Evolución de la arquitectura



Fuente: Proyecto Socio tecnológico XI, en línea, http://segaridebol12.blogspot.com/

En la figura 2 se aprecia la forma en que los modelos de la arquitectura se completan a medida que se llega al final de la etapa de elaboración a través de una o más iteraciones. Al inicio de la etapa de elaboración se tiene una visión incompleta de la línea base de la arquitectura, evidenciándose una implementación parcial del sistema, lo cual mostraría solo algunas funciones y propiedades del software en construcción.

A esta parcialidad en la implementación se le conoce como arquitectura ejecutable. Al pasar del tiempo se evidencia el crecimiento de la arquitectura, lo cual muestra que el modelo se ha estado mejorando progresivamente, mostrando que durante la construcción los diferentes modelos se van desarrollando hasta completarse.

De esta forma se generan artefactos al final de cada etapa, proporcionando un prototipo evolutivo y funcional. De la misma manera la arquitectura como tal no cambia drásticamente pues gran parte de la arquitectura se definió durante la fase de elaboración, pero se pueden realizar pequeños cambios sin que afecte su trazabilidad.

c) Iterativo e Incremental: El aplicativo se divide en pequeños proyectos, donde cada una cumple una parte de las especificaciones, y el desarrollo de la misma es una iteración que va incrementando la funcionalidad del sistema de manera progresiva. Los requisitos y demás modelos no se desarrollan en una sola iteración sino progresivamente, ello con la finalidad de poder garantizar entregas funcionales e iterativas y de tal forma ir completando el sistema software paso a paso.

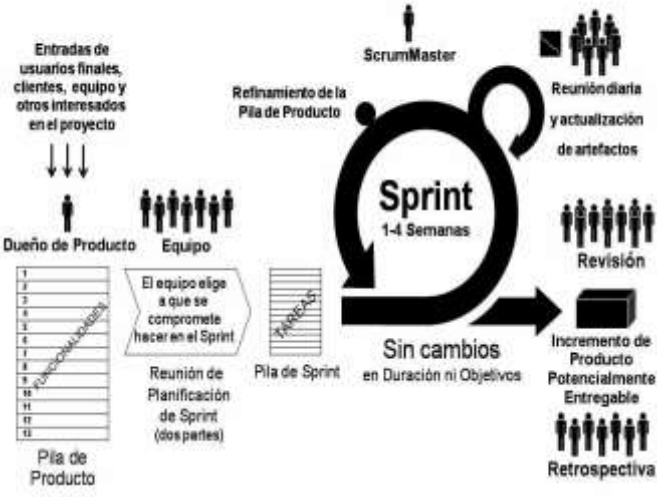
Otro beneficio del desarrollo iterativo e incremental es la priorización de los requerimientos por parte del cliente y del desarrollador. Al evaluar el nivel de criticidad y el riesgo de cada uno, se genera una lista con la ruta crítica del proyecto y la formulación de estrategias para su manejo.

2.2 SCRUM

El desarrollo ágil se centra en equipos multifuncionales con capacidad para decidir por ellos mismos, en vez de grandes jerarquías y divisiones por funcionalidad, se centra en iteraciones rápidas con el cliente dando su opinión continuamente, descentralizado, simple, adaptativo y flexible, con una alta predisposición y respuesta al cambio.

La familia de métodos de desarrollo ágiles evolucionó a partir de los conocidos ciclos de vida iterativos e incrementales. Nacieron de la creencia que un acercamiento más en contacto con la realidad humana (desarrollo de productos basados en el aprendizaje, innovación y cambio) daría mejores resultados. Los principios ágiles ponen el énfasis en construir software que funcione que se pueda usar rápidamente, en vez de pasarse mucho tiempo al principio escribiendo especificaciones. [7]

Figura 3 Scrum .Deemer, P., Benefield, G., Larman, C., & Vodde, B. (2009). Información básica de SCRUM



Fuente: https://bitbucket.org/rafa/drupal2/src/90dcc40947a3/PDF/scrumprimer_es.pdf

En Scrum hay 3 roles principales:

- El Dueño de producto (DP)
- El Equipo
- El Scrum Master (SM)

El dueño de producto es el responsable de maximizar el retorno de inversión (ROI) identificando las funcionalidades del producto, poniéndolas en una lista priorizada de funcionalidades, decidiendo cuales deberían ir al principio de la lista para el siguiente Sprint, y re priorizando y refinando continuamente la lista. Teniendo la responsabilidad de las pérdidas y ganancias del producto, asumiendo que es un producto comercial.

En el caso de una aplicación interna, el DP no es responsable del ROI en el mismo sentido de un producto comercial (que dará beneficio), pero es responsable de maximizar el ROI en el sentido de elegir - en cada Sprint – los elementos de más valor de negocio y menos coste. En algunas ocasiones el DP y el cliente son la misma persona; esto es muy común en aplicaciones internas. En otras, el cliente podría ser millones de personas con diferentes necesidades, en cuyo caso el rol de DP es parecido al rol de jefe de producto o jefe de marketing del producto que hay en muchas empresas.

Sin embargo el dueño de producto es diferente al tradicional jefe de producto porque interactúa activa y frecuentemente con el equipo, estableciendo personalmente las prioridades y revisando el resultado en cada iteración -de 1 a 4 semanas-, en vez de delegar las decisiones de desarrollo en el jefe de proyecto. Es importante destacar que en Scrum hay una persona y sólo una, que hace – y tiene la autoridad final – de dueño de producto.

El equipo construye el producto que va a usar el cliente, por ejemplo una aplicación o un sitio web. El equipo en Scrum es “multi-funcional” – tiene todas las competencias y habilidades necesarias para entregar un producto potencialmente distribuible en cada Sprint – y es “auto organizado” (auto-gestionado), con un alto grado de autonomía y responsabilidad.

En Scrum, los equipos se auto-organizan en vez de ser dirigidos por un jefe de equipo o jefe de proyecto el equipo decide a que se compromete, y como hacer lo mejor para cumplir con lo comprometido. El equipo en Scrum consta para un producto de software el equipo podría incluir analistas, desarrolladores, diseñadores de interface, y testers.

El equipo desarrolla el producto y da ideas al DP de cómo hacer un gran producto. En Scrum, el equipo debería estar dedicado al 100% al trabajo en el producto durante el Sprint; intentando evitar hacer varias tareas en diferentes productos o proyectos. A los equipos estables se les asocia con una productividad más alta, así que evita cambiar miembros del equipo. A los grupos de desarrollo de aplicaciones con mucha gente se les organiza en varios equipos Scrum, cada uno centrado en diferentes funcionalidades del producto, coordinando sus esfuerzos muy de cerca.

Dado que el equipo hace todo el trabajo (planificación, análisis, programación y pruebas) para una funcionalidad completa centrada en el cliente, a los equipos de Scrum también se les llama equipos por funcionalidades. El Scrum Master ayuda al grupo del producto a aprender y aplicar Scrum para conseguir valor de negocio.

El Scrum Master hace lo que sea necesario para ayudar a que el equipo tenga éxito, el Scrum Master no es el jefe del equipo o jefe de proyecto; el Scrum Master sirve al equipo, le protege de interferencias del exterior, enseña, guía al DP, al equipo en el uso fructífero de Scrum, él se asegura de que todo el equipo (incluyendo al DP y la gerencia) entienda siguiendo a las prácticas de Scrum, ayudando a llevar a la organización, a través de los cambios necesarios y frecuentemente difíciles, a conseguir el éxito con el desarrollo ágil.

Como la metodología Scrum, hace visibles muchos impedimentos y amenazas a la efectividad del DP y el equipo, es importante tener un Scrum Master comprometido y que trabaje enérgicamente para ayudar a resolver dichos asuntos, o si no el equipo y el DP tendrán dificultades para tener éxito.

Aunque no todos los métodos ágiles se basan en el desarrollo y entrega incremental, si comparten los principios del manifiesto ágil [8] para el desarrollo de software.

Comparativa entre RUP y SCRUM

Ambas metodologías tienen sus limitaciones y debilidades, así como las metodologías ágiles son las más adecuadas para proyectos pequeños y medianos, no son las más adecuadas para sistemas de gran escala que requieran de interacciones complejas con otros sistemas, debido a que estos sistemas requieren de un nivel de precisión bastante alto y tienen un gran riesgo de construcción.

No sería conveniente implementar una metodología ágil para el desarrollo de un sistema crítico en el cual es necesario el análisis detallado de todos los requerimientos para comprender su complejidad e implicaciones, debido a la complejidad y la extrema precisión que pueda tener la captura de requerimientos, en los cuáles las metodologías agiles como SCRUM ofrecen demasiada flexibilidad.

Karlstrm y Runeson [9] encontraron que los métodos agiles proveen herramientas ponderosas para la planeación a pequeña escala, control del trabajo diario, reporte de progreso y la mejora en los canales de comunicación del equipo.

Los procesos de desarrollo agiles son una gran opción cuando el objetivo es el incremento de la productividad [10], [11] ya que se enfoca en la importancia del manejo del equipo y de personas, o el mejoramiento de la capacidad de respuesta a peticiones de cambio hechas a través del ciclo de desarrollo de software [12].

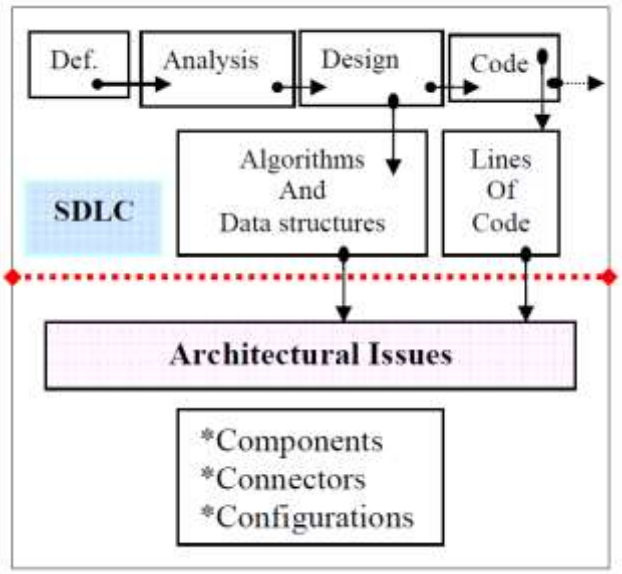
Sin embargo, algunos cambios demandados por procesos agiles son inaceptables para algunas compañías [13][14] y puede causar consternación en los ejecutivos de la organización [13]. Por políticas u otras necesidades, se encuentran situaciones en los cuales el cliente necesita una planeación temprana para efectos de contratación y medir el esfuerzo en tiempo, costos y personal. Poniéndolo en otras palabras, un proceso establecido y controlado es a veces necesario.

Por otro lado, la teoría hace ver RUP como un proceso rígido e inmodificable por la gran cantidad de documentación que es necesaria completar antes de la finalización de cada etapa, pero realmente es un proceso de software configurable, donde a cada stakeholder se le asignan sus actividades acorde con su rol dentro del proyecto, además del uso de una definición común para el proceso que es compartida por todo el equipo de desarrollo, y de esta forma asegurar una comunicación clara y sin ambigüedades entre los miembros del equipo.

Otro punto a resaltar del uso de RUP como metodología de desarrollo es su aplicación en la construcción de software basado en componentes, que ha tenido un gran avance desde la idea de construir nuevas aplicaciones teniendo en cuenta bloques prefabricados, propuesto inicialmente en los 60 [15].

Actualmente se busca la construcción de componentes que brinden servicios largamente usados dentro de la cadena vertebral de arquitecturas empresariales [15]. Como se ilustra en la figura 4, RUP por su proceso centrado en casos de uso y arquitectura, es ideal para este tipo de desarrollo.

Figura 4 - Aspectos arquitecturales



Fuente: IJCSNS International Journal of Computer Science and Network Security, Process Model for Software Architecture, VOL.7 No.4, April 2007A

Los partidarios de los procesos agiles continúan percibiendo la arquitectura de software como algo del pasado, por el hecho de la gran documentación que demanda su construcción. Ellos ven un bajo valor al diseño arquitectural, y que ésta emergerá gradualmente sprint a sprint, como el resultado iterativo e incremental de la solución a desarrollar. Por otro lado, en compañías donde se tienen practicas arquitecturales sólidas, las practicas agiles son a veces vistas como poco maduras y poco probadas, limitadas a desarrollos pequeños donde los riesgos son mínimos.

Otro punto de comparación entre las dos metodologías de software es que el tipo de documentación de RUP hace referencia a las comunicaciones formales con la finalidad de ser más predictivos, mientras que en SCRUM se enfoca en las comunicaciones informales continuas y a la adaptación al cambio, con la finalidad de ser más adaptivas.

Otra diferencia está relacionada con las iteraciones de desarrollo, mientras que en RUP tienden a ser pocas y largas que finalizan con la entrega de un artefacto, en SCRUM tienden a ser muchas pero frecuentes, llamadas sprints. En la metodología SCRUM se realizan reuniones diarias las cuales son llamadas "Daily Scrum" y es donde se sostiene una pequeña charla sobre el estado del proyecto. En particular muestran los impedimentos para progresar que se atraviesan y que la gerencia debe resolver.

Propuesta de metodología hibrida

La disciplina es la base de cualquier esfuerzo exitoso, una cualidad que en el desarrollo de software se ilustra en la aplicación de procesos y metodologías. Sin estas habilidades el éxito de un proyecto se sostiene únicamente en el talento personal de los integrantes del equipo, dejando a un lado la parte profesional que puede apoyar el desarrollo.

La disciplina y la constancia soportan el esfuerzo en situaciones difíciles, cuando algo nuevo o inesperado sale a la luz y una respuesta es requerida. La disciplina crea conocimientos y procesos estructurados, además de aumentar la experiencia. La agilidad es la contraparte de la disciplina. Donde la disciplina arraiga y fortalece, la agilidad libera e inventa. La agilidad aplica la experticia y el conocimiento estructurado en nuevos entornos, reacciona y adapta, toma ventaja de oportunidades inesperadas, y mejora la experiencia en un ciclo continuo. Toda organización en un mundo cambiante requiere ambas, la agilidad y la disciplina

Para mitigar los impactos de cambios abruptos de paradigma, soportar a las organizaciones que no piensan en cambiar sus prácticas tradicionales, generar una arquitectura clara desde el inicio del proyecto y darle una mayor importancia al equipo de trabajo, es necesario generar propuestas de procesos híbridos que incorporen principios de paradigmas tradicionales y agiles, una propuesta que incluya disciplina y agilidad de acuerdo a el tamaño del proyecto, el perfil y tamaño del equipo, criticidad, dinamismo y cultura del proyecto.

Se debe generar un modelo donde se haga explícita la importancia en el manejo del equipo y personas propia de metodologías agiles, complementándolo con la disciplina y rigidez (que no es tanta como se ilustro en el capítulo II) provenientes de las metodologías clásicas que están más inclinadas a los procesos y la documentación.

Para tal motivo, se propone una metodología de desarrollo con las siguientes características:

- Tener en cuenta el énfasis propuesto por RUP para la generación de la arquitectura base del proyecto.
- Usar UML como lenguaje para el diseño de dicha arquitectura.
- Realizar únicamente los diagramas necesarios para el entendimiento de la arquitectura.
- Involucrar activamente al cliente en el desarrollo del sistema, tal como lo propone SCRUM
- La especificación de casos de uso debe realizarse con formatos formales, las historias de usuario formuladas por SCRUM o los formatos propuestos por IEEE
- Realizar la planeación del proyecto teniendo en cuenta las 4 etapas propuestas por RUP (inicio, análisis, construcción e implementación) y finalizar cada una de estas con un artefacto

- Realizar la planeación de cada una de las etapas del punto anterior realizando un control de las actividades diarias por medio de backlogs formulados por SCRUM
- Durante el proyecto realizar las reuniones diarias o "Dairy Scrum" formuladas por SCRUM para discutir sobre el estado del proyecto y en particular mostrar los impedimentos para progresar y las acciones a tomar por parte de la gerencia para resolverlos.
- Generar los scripts de prueba teniendo en cuenta las especificaciones de casos de uso realizados.

3. CONCLUSIONES

Las metodologías ágiles permiten disminuir costos y brindar flexibilidad a los proyectos de software donde la incertidumbre está presente.

Los procesos de desarrollo agiles son una gran opción cuando el objetivo es el incremento de la productividad y el mejoramiento de la capacidad de respuesta a peticiones de cambio hechas a través del ciclo de desarrollo de software

Para que un grupo de desarrollo adopte una metodología ágil debe poseer experiencia trabajando con metodologías tradicionales, ya que la experiencia es la que predomina en los momentos cruciales del proyecto, además debe tener la capacidad de ser equipos auto-gestionados, altamente motivados y con gran innovación

Cualquier organización inmersa en la era de la información, debe reconocer el talento humano con el que se cuenta, y considerarlas su principal activo. Se debe mantener al personal con un alto nivel de satisfacción (a través de capacitación y motivación) para obtener su máximo rendimiento.

El proceso hibrido propuesto incorpora ventajas y beneficios de las metodologías agiles, reconociendo la importancia del manejo de requerimientos y arquitectura de acuerdo a paradigmas tradicionales.

REFERENCIAS

[1] Bohem, (1976) B.W. Bohem. Software Engineering. IEEE Trans. Computers, Diciembre 1976, pp. 1226-1241.

[2] Leterlier P, Introducción a RUP, Departamento de Sistemas informáticos y Computación (DSIC), Universidad Politécnica de Valencia (UPV), disponible en: http://ybathich.site90.com/documentos/Material/Metodologia/RUP/Introducci%F3n%20a%20RUP.doc, recuperado: 01 de Junio 2013.

[3] I. Sommersville (2010). Ingeniería del Software. Un enfoque práctico. 9ª edición. Editorial Addison-Wesley

[4] Booch, G.; Rumbaugh, J. & Jacobson, I. (2000), El proceso unificado de desarrollo de software, Pearson Educación, Madrid.

[5] Abrahamsson, P.; Salo, O.; Ronkainen, J. & Warsta, J. (2002), Agile software development methods: Review and analysis, Espoo 2002, VTT Publications 478, Oulu.

[6] Pressman, R. & Murrieta, J. (2006), Ingeniería del software un enfoque práctico. 6ª Edición. McGraw-Hill, pp. 67-73.

[7]Deemer,P.B.(2011).The scrum primer. .https://bitbucket.org/rafa/drupal2/src/90dcc40947a3/PDF/scrumprimer_es.pdf, recuperado; Febrero de 2014.

[8] Manifesto for Agile Software Development. http://agilemanifesto.org/RecuperadoMarzo 2014

[9] D. Karlstrom and P. Runeson (2006), "Integrating agile software development into stage-gate managed product development," in Empirical Software Engineering,

[10] A. Cockburn (2002), Agile Software Development: Software Through People, A. Wesley.

[11]SIlieva P.Ivanov, and E. Stefanova, (2004), "Analyses of an agile methodology implementation," in Euromicro Conference. Rennes, France: IEEE Computer Society Press.

[12] A. Cockburn and L.Williams, "An agile software development: it's about feedback and change," IEEE Computer, vol. 36, pp. 39–43.

[13]D. Leffingwell, (2011) Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise, 1st ed., ser. Agile Software Development Series. Addison-Wesley Professional, January 2011, no. ISBN-13: 978-0321635846.

[14] T. Dyb and T. Dingsoyr, (2009),"What do we know about agile software development?" vol. 26, no. 5, pp. 6–9, 2009.

[15] C. Atkinson and O. Hummel,(2012), "Iterative and Incremental Development of Component-Based Software Architectures".

ISSN: 2344-8288 | Vol. 1 | No 2 | Enero -Agosto 2014 | TIA