INGENIERIA DE SOFTWARE

METODOLOGIAS AGILES RUP

MATERIA : ***Ing. Software***

ESTUDIANTE : ***Benedicto Canaza M.***

DOCENTE : ***Lic. Ervin Flores***

GESTION : ***2009***

El Alto – Bolivia

METODOLOGIAS AGILES RUP

I. INTRODUCCIÓN

Se entiende como **Desarrollo Ágil de Software** a un paradigma de Desarrollo de Software basado en procesos ágiles. Los procesos ágiles de desarrollo de software, conocidos anteriormente como *metodologías livianas*, intentan evitar los tortuosos y burocráticos caminos de las metodologías tradicionales enfocándose en la gente y los resultados.

Es un marco de trabajo conceptual de la ingeniería de software que promueve iteraciones en el desarrollo a lo largo de todo el ciclo de vida del proyecto. Existen muchos métodos de desarrollo ágil; la mayoría minimiza riesgos desarrollando software en cortos lapsos de tiempo. El software desarrollado en una unidad de tiempo es llamado una iteración, la cual debe durar de una a cuatro semanas. **Cada iteración del ciclo de vida incluye: planificación, análisis de requerimientos, diseño, codificación, revisión y documentación.** Una iteración no debe agregar demasiada funcionalidad para justificar el lanzamiento del producto al mercado, pero la meta es tener un demo (sin errores) al final de cada iteración. Al final de cada iteración el equipo vuelve a evaluar las prioridades del proyecto.

Los métodos Agiles enfatizan las comunicaciones cara a cara en vez de la documentación. La mayoría de los equipos Agiles están localizados en una simple oficina abierta, a veces llamadas "plataformas de lanzamiento" (bullpen en inglés). La oficina debe incluir revisores, diseñadores de iteración, escritores de documentación y ayuda y directores de proyecto.

Los métodos ágiles también enfatizan que el software funcional es la primera medida del progreso. Combinado con la preferencia por las comunicaciones cara a cara, generalmente los métodos ágiles son criticados y tratados como "indisciplinados" por la falta de documentación técnica.

II. HISTORIA

La definición moderna de desarrollo ágil de software evolucionó a mediados de los años 1990 como parte de una reacción contra los métodos de “peso pesado”, muy estructurado y estricto, extraídos del modelo de desarrollo en **cascada.** El proceso originado del uso del modelo en cascada era visto como burocrático, lento, degradante e inconsistente con las formas de desarrollo de software que realmente realizaban un trabajo eficiente. Los métodos de desarrollos ágiles e iterativos pueden ser vistos como un

retroceso a las prácticas de desarrollo observadas en los primeros años del desarrollo de software (aunque en ese tiempo no había metodologías formales). Inicialmente, los métodos ágiles fueron llamados métodos de "peso liviano". En el año 2001, miembros prominentes de la comunidad se reunieron en Sonwbird, Utah, y adoptaron el nombre de "Metodologías ágiles". Poco después, algunas de estas personas formaron la "alianza ágil", una organización sin fines de lucro que promueve el desarrollo ágil de aplicaciones. Muchos métodos similares al ágil fueron creados antes del 2000. Entre los más notables se encuentran: Scrum(1986), Crystal Clear (cristal transparente), programación extrema o XP (1996), desarrollo de software adaptativo, feature driven development, Método de desarrollo de sistemas dinámicos(1995). Kent Beck creó el método de Programación Extrema (usualmente conocida como XP) en 1996 como una forma de rescatar el proyecto del Sistema exhaustivo de compensaciones de Chrysler (C3). Mientras Chrysler cancelaba ese proyecto, el método fue refinado por Ron Jeffries.

III. METODOLOGIA RUP

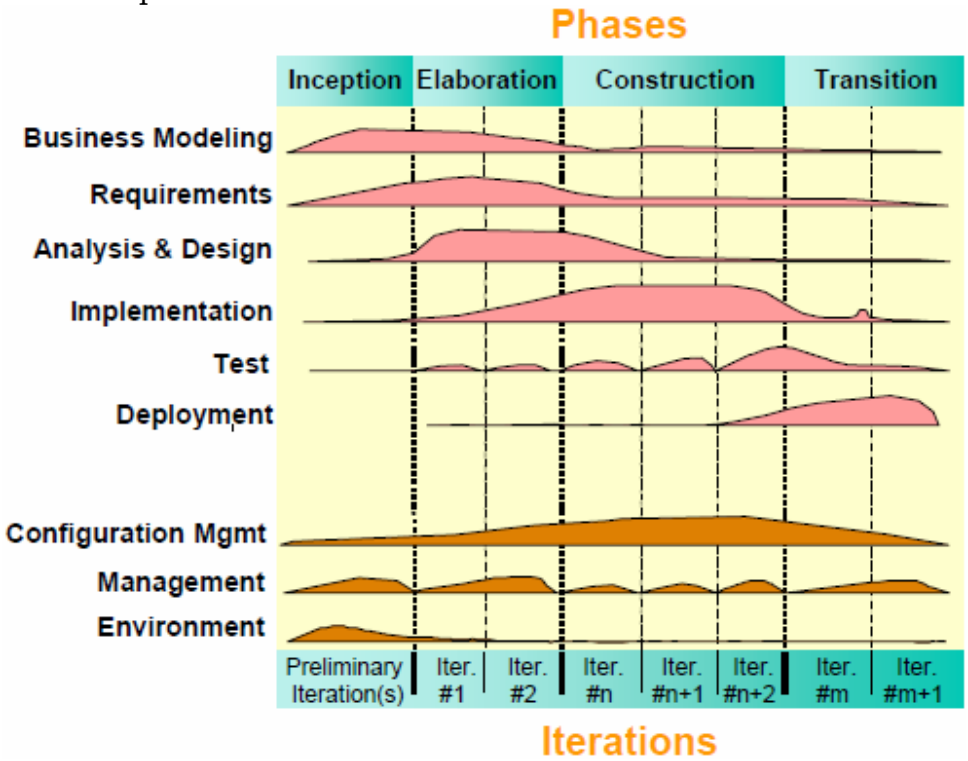
Siempre que empezamos discutiendo métodos en la arena OO, inevitablemente salimos con el papel del Rational Unified Process. El Proceso Unificado fue desarrollado por Philippe Kruchten, Ivar Jacobson y otros de la Rational como el proceso complementario al UML. El RUP es un armazón de proceso y como tal puede acomodar una gran variedad de procesos. De hecho ésta es mi crítica principal al RUP - como puede ser cualquier cosa acaba siendo nada. Yo prefiero un proceso que dice qué hacer en lugar de dar opciones infinitas.

Como resultado de esta mentalidad de armazón de procesos, el RUP puede usarse en un estilo muy tradicional de cascada o de una manera ágil. Como resultado usted puede usar el RUP como un proceso ágil, o como un proceso pesado - todo depende de cómo lo adapte a su ambiente.

Craig Larman es un fuerte defensor de usar el RUP de una manera ágil. Su excelente libro introductorio sobre desarrollo OO contiene un proceso que está muy basado en su pensamiento ligero del RUP. Su visión es que mucho del reciente empujón hacia los métodos ágiles no es nada más que aceptar desarrollo OO de la corriente principal que ha sido capturada como RUP. Una de las cosas que hace Craig es pasarse los primeros dos o tres días de una iteración mensual con todo el equipo usando el UML para perfilar el diseño del trabajo a hacerse durante la iteración. Esto no es un cianotipo del que no se pueda desviarse, sino un boceto que da una perspectiva sobre cómo pueden hacerse las cosas en la iteración.

Otra tachuela en el RUP ágil es el proceso dX de Robert Martin. El proceso dx es una versión totalmente dócil del RUP que simplemente es idéntico a la XP (voltear dX al revés para ver la broma). El dX está diseñado para gente que tiene que usar el RUP pero quiere usar XP. Como tal es a la vez XP y RUP y por tanto un buen ejemplo del uso ágil del RUP.

Para mí, una de las cosas clave que necesita el RUP es que los líderes del RUP en la industria enfaticen su acercamiento al desarrollo de software. Más de una vez he oído a la gente que usa el RUP que están usando un proceso de desarrollo estilo cascada. Gracias a mis contactos en la industria, sé que Philippe Kruchten y su equipo son firmes creyentes en el desarrollo iterativo. Clarificando estos principios y animando las versiones ágiles del RUP tales como los trabajos de Craig y de Robert tendrá un efecto importante.



Rational Unified Process (RUP)

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 o 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. Secorrigen 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 o 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. El proceso de implementación mismo es complejo, dividiéndose en seis fases cíclicas.

Existe una versión recortada de RUP, dX de Robert Martin, en la cual se han tomado en consideración experiencias de diversos MAs, reduciendo los artefactos de RUP a sus mínimos esenciales y (en un gesto heroico) usando tarjetas de fichado en lugar de UML. Es como si fuera RUP imitando los principios de XP; algunos piensan que dX es XP de cabo a rabo, sólo que con algunos nombres cambiados [ASR+02]. RUP se ha combinado con Evo, Scrum, MSF y cualquier metodología imaginable. Dado que RUP es suficientemente conocido y su estructura es más amplia y compleja que el de cualquier otro método ágil, su tratamiento en este texto concluye en este punto.

IV. Conclusiones

- No existe una metodología universal para hacer frente con éxito a cualquier proyecto de desarrollo de software.
- Las metodologías tradicionales históricamente han intentado abordar la mayor cantidad de situaciones del contexto del proyecto, exigiendo un esfuerzo considerable para ser adaptadas, sobre todo en proyectos pequeños y de requisitos cambiantes.

- Las metodologías ágiles ofrecen una solución casi a medida para una gran cantidad de proyectos.
- Las metodologías ágiles se caracterizan por su sencillez, tanto en su aprendizaje como en su aplicación; sin embargo, gozan tanto de ventajas como de inconvenientes.
- Las metodologías ágiles permiten a los pequeños grupos de desarrollo concentrarse en la tarea de construir software fomentando prácticas de fácil adopción y en un entorno ordenado que permiten que los proyectos finalicen exitosamente.
- Se pueden distinguir dos rangos distintos de conjuntos de metodologías ágiles. Por un lado están las metodologías más declarativas y programáticas como XP, Scrum, LD, entre otras; y por otro lado se encuentran las metodologías finamente elaboradas como DSDM, Cristal, etc.
- XP es una de las metodologías ágiles más extendidas y populares, además es considerada como una metodología posmoderna cuyas grandes capacidades se generan a través de procesos emergentes.
- Scrum implementa en sus prácticas de desarrollo una estrategia de caos controlado, permitiendo maximizar la realimentación sobre el desarrollo pudiendo corregir problemas y mitigar riesgos de forma temprana.
- A pesar de las continuas criticas que las metodologías ágiles sufren, son usadas por muchas grandes empresas y se han utilizado en grandes sistemas, lo que hace prever que estas metodologías han llegado para quedarse.