POTENCIALIDADES Y DEBILIDADES DEL PROCESO UNIFICADO DE DESARROLLO Y WATCH

Padilla Vedia Carmen Janeth
Departamento de Informática y Sistemas UAJMS
Tarija, Bolivia

Correo electrónico: padillac555@gmail.com

RESUMEN

Al realizar este artículo se trata de reflejar una panorámica acerca de los conceptos y características de la Ingeniería de Requerimientos (IR), buscando resaltar su relevancia dentro del ciclo de desarrollo de proyectos de software, conocer las diferentes alternativas o técnicas que existen para identificarlos, analizarlos, documentarlos ,así como mostrar la importancia que tienen herramientas automatizadas dentro de este proceso de administración de requerimientos.

PALABRAS CLAVE:

RUP, WATCH, stakeholders, Fases, Flujos, Actividades, Trabajadores, Procesos, Release.

PROBLEMA DE INVESTIGACIÓN

Todo proceso de desarrollo de software implica el trabajo coordinado de fases, etapas o como se denominen de acuerdo a la metodología empleada para su desarrollo. Lo cierto es que si tuviésemos que enfrentar un proyecto de desarrollo de software , nos enfrentamos a la interrogante ¿ que metodología o método usar para hacerle frente a esta actividad?, la cual conlleva tareas y responsabilidades dentro de una organización de desarrollo y que debe permitir obtener productos en tiempos considerables con costos adecuados , al igual que los resultados finales cumplan con los atributos de calidad mínimamente exigibles por el cliente. En esta oportunidad se presentara una evaluación comparativa de la metodología RUP y el método WATCH, identificando fortalezas y debilidades que presentan estas alternativas de para hacerle frente al proceso de desarrollo de software.

1. INTRODUCCIÓN

Desde hace tiempo se viene enfrentando el desarrollo de productos de software con metodologías tradicionales que permitían llevar adelante este proceso que en principio tenían características mayormente secuenciales, dada esta particularidad se fueron quedando obsoletas y fuimos pasando a otras tendencias como las orientadas a objetos a medida que los lenguajes de programación evolucionaban con estas particularidades.

Las metodologías tradicionales fueron explotadas oportunamente cuando las necesidades de los usuarios eran mínimas comparadas a las exigencias de los usuarios de hoy en dia, sin embargo estas metodologías cubrieron un marco de metodológica el proceso de desarrollo de software y permitir a los desarrolladores entregar productos con calidad en entornos que requerían menos recursos que los entornos actuales.

Hoy en dia los requerimientos son mayores por los tanto hay que sustentarse en modelos que permitan enfrentar el proceso de desarrollo de forma que podamos satisfacer estas necesidades que cada dia son más y mas exigentes por los usuarios. Hoy en dia los usuarios necesitan respuestas

tecnológicamente más complejas que nos hacen pensar en la (s) metodologías a emplear para el desarrollo serán aquellas que cubran de manera exitosa estas necesidades que son evidentes en nuestros entornos actuales.

El desarrollo de software para entornos o dominios empresariales es un proceso complejo que requiere el tratamiento apropiado de aspectos organizacionales, gerenciales y tecnológicos por tanto un de los aspectos importantes es la metodología a emplear para lograr este propósito.

1.1. Método WATCH

El método WATCH es un marco metodológico que describe los procesos técnicos, gerenciales y de soporte que deben emplear los equipos de trabajo que tendrán a su cargo el desarrollo de aplicaciones de software empresarial.

Un marco metodológico es un patrón que debe ser instanciado, es decir adaptado cada vez que se use. Cada equipo de trabajo deberá usar el método como un patrón o plantilla metodológica, a partir de la cual dicho equipo debe elaborar el proceso específico de desarrollo de la aplicación que se desea producir.

Se ubica dentro de los métodos disciplinados ya que se centra en los procesos, hace énfasis en los productos y la organización, involucra procesos bien definidos y documentados, requiere de alta formalidad en el proceso de desarrollo, son procesos repetibles, los resultados son predecibles.

Este método incluye, también, una descripción de los procesos de gerencia del proyecto que se aplicarán para garantizar que el proyecto se ejecute en el tiempo previsto, dentro del presupuesto acordado y según los estándares de calidad establecidos.

1.2. Proceso Unificado de Desarrollo de Software

Es un proceso que de manera ordenada defina las tareas y quién de los miembros del equipo de desarrollo las hará. Es una guía para usar UML.

RUP es el resultado de varios años de desarrollo y uso práctico en el que se han unificado técnicas de desarrollo, a través del UML, y trabajo de muchas metodologías utilizadas por los clientes. La versión que se ha estandarizado vio la luz en 1998 y se conoció en sus inicios como Proceso Unificado de Rational 5.0; de ahí las siglas con las que se identifica a este proceso de desarrollo.

Como RUP es un proceso, en su modelación define como sus principales elementos: Trabajadores, actividades, artefactos, flujo de actividades.

1.3. Proceso de Desarrollo de Software según RUP

Un Proceso de Desarrollo de Software es la definición del conjunto de actividades que guían los esfuerzos de las personas implicadas en el proyecto, a modo de plantilla que explica los pasos necesarios para terminar el proyecto2. En la fig. 1 se indica que este conjunto de actividades, en el proceso de desarrollo de software que proponen Jacobson, Rumbaugh y Booch, tiene la misión de transformar los requerimientos del usuario en un producto de software; de manera que los integrantes del equipo y todo aquel que pueda estar interesado en el producto final, tenga la misma visión y no ocurra cuando no se aplica un proceso de desarrollo (fig. 2).

Fig. 1 *Proceso de desarrollo de software*

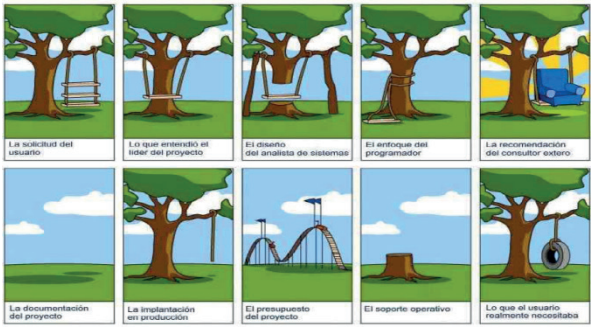

Fig. 2: *Visiones del producto final*

1.4. Proceso de desarrollo según WATCH

El método WATCH está fundamentado en las mejores prácticas de la Ingeniería de Software y la Gestión de Proyectos. Cubre todo el ciclo de vida de las aplicaciones; desde el modelado del dominio de la aplicación, pasando por la definición de los requisitos de los usuarios, hasta la puesta en operación de la aplicación.

Este método incluye, también, una descripción de los procesos de gerencia del proyecto que se aplicarán para garantizar que el proyecto se ejecute en el tiempo previsto, dentro del presupuesto acordado y según los estándares de calidad establecidos.

El método WATCH está compuesto por tres modelos fundamentales: (fig. 3 Modelos de WATCH)

- Un modelo de productos
- Un modelo de Actores
- Un modelo de Procesos

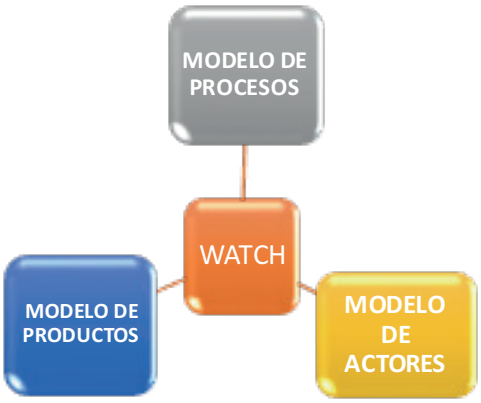

Fig. 3: *Modelos de WATCH*

2. CARACTERÍSTICAS DEL PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE

a) Las 4 P del proceso de Desarrollo

Se considera piedras angulares del proceso de desarrollo del software a: el proyecto, las personas y el producto; siendo las características del cliente, el entorno de desarrollo y las condiciones del negocio, elementos que influyen en el proceso.

El conocimiento y la experiencia que crea y sostiene la evolución del producto lo tienen las personas. Ellas también financian, se benefician, lo prueban y planifican. Sin un personal competente y experimentado es imposible crear productos competitivos que satisfagan las necesidades de los clientes. Además, el modo en que organiza y gestiona un producto afectan a las personas involucradas en su realización.

Un proyecto es un elemento organizativo a través del cual se gestiona el desarrollo del software.

Un proyecto de desarrollo obtiene un versión de un producto que contiene modelos, código fuente, documentación y un ejecutable. Este producto va evolucionando durante el proceso de desarrollo desde un proyecto inicial o innovador (prototipo

inicial) hasta convertirse en un release del proyecto.

b) Lenguaje Unificado de Modelado

UML es un lenguaje para visualizar, especificar, construir y documentar los artefactos de un sistema que involucra una gran cantidad de software, este lenguaje abarca actividades tales como:

- **Visualizar,** Lenguaje gráfico con una semántica bien definida que estandariza la modelación durante el proceso de desarrollo del software para que sea legible por todo el equipo de proyecto.
- **Especificar**, Se construyen modelos precisos, no ambiguos y completos.
- **Documentar**, Permite describir requerimientos, la arquitectura y modelar las pruebas a través de artefactos que permiten documentar el proceso.
- **Construir**, No es un lenguaje de programación, pero sus modelos pueden transformarse en código fuente, tablas o al almacenamiento de objetos (Generación directa del código).

c) Fases

Presenta 4 fases : Inicio, Elaboración, Construcción y Transición

- **Conceptualización (Concepción o Inicio):** Se describe el negocio y se delimita el proyecto describiendo sus alcances con la identificación de los casos de uso del sistema.
- **Elaboración:** Se define la arquitectura del sistema y se obtiene una aplicación ejecutable que responde a los casos de uso que la comprometen. A pesar de que se desarrolla a profundidad una parte del sistema, las decisiones sobre la arquitectura se hacen sobre la base de la comprensión del sistema completo y los requerimientos (funcionales y no funcionales) identificados de acuerdo al alcance definido.
- **Construcción:** Se obtiene un producto listo para su utilización que está documentado y tiene un manual de usuario . Se obtiene 1 o varios release del producto que han pasado las pruebas. Se ponen estos release a consideración de un subconjunto de usuarios.
- **Transición:** El release ya está listo para su instalación en las condiciones reales. Puede implicar reparación de errores.

Estas fases están relacionadas con 7 flujos fundamentales los cuales son:

- Modelamiento del negocio: Describe los procesos de negocio, identificando quiénes participan y las actividades que requieren automatización.
- Requerimientos: Define qué es lo que el sistema debe hacer, para lo cual se identifican las funcionalidades requeridas y las restricciones que se imponen.
- Análisis y diseño: Describe cómo el sistema será realizado a partir de la funcionalidad prevista y las restricciones impuestas (requerimientos), por loq ue indica con precisión lo que se debe programar.
- Implementación: Define cómo se organizan las clases y objetos en componentes, cuáles nodos se utilizarán y la ubicación en ellos de los componentes y la estructura de capas de la aplicación.
- Prueba (Testeo): Busca los defectos a los largo del ciclo de vida.

- Instalación: Produce release del producto y realiza actividades (empaque, instalación, asistencia a usuarios, etc.) para entregar el software a los usuarios finales.

- Administración del proyecto: Involucra actividades con las que se busca producir un producto que satisfaga las necesidades de los clientes.

- Administración de configuración y cambios: Describe cómo controlar los elementos producidod por todos los integrantes del equipo de proyecto en cuanto a: utilización/ actualización concurrente de elementos, control de versiones, etc.

- Ambiente: Contiene actividades que describen los procesos y herramientas que soportarán el equipo de trabajo del proyecto; así como el procedimiento para implementar el proceso en una organización.

d) El ciclo de vida de RUP se caracteriza por:

1. Dirigido por casos de uso: Los casos de uso reflejan lo que los usuarios futuros necesitan y desean, lo cual se capta cuando se modela el negocio y se representa a través de los requerimientos. A partir de aquí los casos de uso guían el proceso de desarrollo ya que los modelos que se obtienen, como resultado de los diferentes flujos de trabajo, representan la realización de los casos de uso (cómo se llevan a cabo).

2. Centrado en la arquitectura: La arquitectura muestra la visión común del sistema completo en la que el equipo de proyecto y los usuarios deben estar de acuerdo, por lo que describe los elementos del modelo que son más importantes para su construcción, los cimientos del sistema que son necesarios como base para comprenderlo, desarrollarlo y producirlo económicamente. RUP se desarrolla mediante iteraciones, comenzando por los CU relevantes desde el punto de vista de la arquitectura.

3. Iterativo e Incremental: Los flujos de trabajo se desarrollan en cascada, RUP propone que cada fase se desarrolle en iteraciones. Una iteración involucra actividades de todos los flujos de trabajo, aunque desarrolla fundamentalmente algunos más que otros.

Aunque cada iteración tiene que proponerse un incremento en el proceso de desarrollo, todas deben aportar al principal resultado de la fase en la que se desarrolla.

3. CARACTERÍSTICAS DEL MODELO WATCH

a)

1) Un modelo de productos que describe los productos intermedios y finales que se generan, mediante el uso del método, durante el desarrollo de una aplicación empresarial.

2) Un modelo de actores que identifica a los actores interesados (stakeholders) en el desarrollo de una aplicación y describe cómo deben estructurarse los equipos de desarrollo y cuáles deben ser los roles y responsabilidades de sus integrantes.

3) Un modelo de procesos que describe detalladamente los procesos técnicos, gerenciales y de soporte que los equipos de desarrollo deberán emplear para elaborar las aplicaciones.

b) El método WATCH emplea el paradigma de desarrollo de software basado en la reutilización

de componentes de software. En base a este paradigma, una aplicación empresarial tiene una arquitectura de software de tres o más capas, en la que cada una de las capas está compuesta de un conjunto de componentes de software interrelacionados.

c) Está sólidamente fundamentado. Posee una base conceptual y metodológica muy bien sustentada. El método descansa en conceptos bien establecidos que se derivan de la Ingeniería de Software y los Sistemas de Información Empresarial. En concreto, el método emplea una arquitectura de dominio de tres capas que define los elementos principales de las aplicaciones empresariales modernas.

Metodológicamente, el modelo ha sido elaborado tomando como referencia modelos de procesos bien conocidos o bien fundamentados, tales como el modelo RUPRational Unified Process (Krutchen, 2000) y versiones anteriores del método WATCH (Montilva y Barrios, 2004b).

d) Es estructurado y modular. Posee una clara estructura que facilita su comprensión y utilización. Esta estructura separa los tres elementos primordiales de un método: el producto que se quiere elaborar, los actores que lo elaboran y el proceso que siguen los actores para elaborar el producto. Estos tres elementos definen los tres componentes del método WATCH: modelo de productos, modelo de actores y modelo de procesos. Cada uno de ellos posee, a su vez, una estructura claramente visible y acorde al elemento que representa.

e) Es de propósito específico. El método está dirigido al desarrollo de aplicaciones de software en entornos empresariales; es decir, al desarrollo de aplicaciones que apoyan uno o más sistemas de negocios de una empresa. Esta orientación concreta y específica resuelve los problemas que tienen la mayoría de los métodos comerciales y académicos existentes, cuya generalidad va en detrimento de su aplicabilidad en software especializado. El método no es apropiado para desarrollar software del sistema (sistemas operativos, utilitarios, middleware, etc.), ni software de programación (compiladores, editores, entornos de programación, etc.).

Tampoco es útil en el desarrollo de software de entretenimiento (videojuegos, herramientas multimedia, etc.). En aplicaciones especializadas, tales como sistemas de información geográfica (GIS), sistemas de control, software educativo y software embebido, el usuario del método debe hacer las adaptaciones pertinentes para ajustar el método al dominio particular de este tipo de aplicaciones.

4. ASPECTOS COMPARATIVOS ENTRE EL PROCESO UNIFICADO DE DESARROLLO DE SOFTWARE (RUP) Y EL MARCO METODOLÓGICO WATCH

4.1 Potencialidades en ambos modelos

- WATCH considera el desarrollo de software iterativo, incremental y versionado, de forma similar lo hace RUP con la particularidad de estar centrado en la arquitectura , guiado por casos de uso, iterativo e incremental, rescatando una potencialidad de esta metodología como lo es "guiado por los casos de uso" lo que implica de forma puntual que el modelo de caso de uso está presente a lo largo de todo el proceso de desarrollo de software.

- Verificación continúa de la calidad de los productos. WATCH asegura la calidad de la aplicación, a través del uso de procesos bien

definidos de Aseguramiento de la Calidad y Verificación & Validación de software (V&V). Los procesos V&V son aplicados a todos los productos intermedios y finales que se elaboran a lo largo del desarrollo de cada aplicación. En tanto RUP con sus componentes asegura calidad en los productos la misma que se obtiene a lo largo del proceso de desarrollo, aplicando métricas de calidad, permitiendo la toma de decisiones adecuadas, tiempos reales en la planificación y pruebas exitosas que evitan la demora en la aceptación de los usuarios.

- WATCH considera una programación guiada por las pruebas. Para codificar los componentes de software, el método emplea el enfoque de programación guiada por las pruebas, la cual consiste en diseñar y preparar las pruebas de cada componente antes de iniciar su codificación. De esta manera, la codificación se hace con la intención de pasar la prueba, lo cual garantiza una mayor calidad del código producido. La codificación y la prueba unitaria del componente se hacen paralela y coordinadamente usando herramientas de pruebas automatizadas. RUP enfrenta esta actividad por casos de prueba que pueden ser diseñados en etapas tempranas del proceso, ya que su modelo lo permite realizar desde la fase inicial y a medida que avanza el proceso se van mejorando en cada iteración, todo esto gracias al modelo de casos de uso que es el hilo conductor del proceso y nos sirve como base para el diseño de las pruebas tanto iniciales como finales.

- WATCH considera una apropiada gestión de cambios. Los cambios en los requisitos y productos elaborados es una constante en el desarrollo de aplicaciones empresariales. Estos cambios pueden surgir en cualquier fase del desarrollo de una aplicación, por lo que es necesario controlarlos apropiadamente, a fin de evitar que el proyecto se postergue continua o indefinidamente. WATCH emplea procesos bien definidos de Gestión de Requisitos y Gestión de la Configuración de Software (SCM) que se encargan de controlar estos cambios. RUP por su parte es claro y preciso en cuanto a la gestión de cambios de refiere, ya que las iteraciones junto con los hitos de cada fase permiten tener un control efectivo de cambios y así permitir los incrementos efectivos a lo largo del proceso de desarrollo de software.

- Ambos modelos WATCH y RUP Emplean Buenas Prácticas y Procesos de Gestión de Proyectos. Integra los procesos de gestión con los procesos técnicos y de soporte.

  - Ambos modelos contemplan visibilidad al proyecto; pues, permite que el grupo de desarrollo y los usuarios del sistema puedan conocer en qué estado se encuentra el proyecto en cualquier momento.

  - Facilita al líder del proyecto las labores de planificación y control del proyecto.

  - Establecen un marco metodológico único que estandariza el proceso de desarrollo y unifica la documentación que se produce a lo largo del proyecto de desarrollo de una aplicación.

  - Están fundamentados en modelos de procesos de la Ingeniería de Software Basada en Componentes.

  - Emplean las mejores prácticas, técnicas y

notaciones utilizadas regularmente en la Industria del Software.

4.2 Debilidades en los modelos

4.2.1 WATCH

- Los aspectos iterativos e incremental no están claramente definidos de forma que permitan una aplicabilidad clara durante el proceso de desarrollo como lo hace RUP con las fases y 9 flujos a realizar en cada fase.
- No es un modelo muy aplicable por su poca popularidad y la presencia de una gran numero de detalles que muchas veces hacen perderse en el proceso de desarrollo.
- La aplicación de procesos, técnicas y prácticas gerenciales es un factor crítico de éxito en el desarrollo de software.
- Considera un estilo de cascada que confunde con lo iterativo, faltando puntualizar que procedimientos seguir, como y cuando de forma que sean verdaderas guías en el proceso de desarrollo.

4.2.2 RUP

- Es un método con un cierto grado de complejidad, implica seguir una visión bidimensional en las que se debe respetar fases y flujos , los mismos que requieren conocimientos bien definidos sobre procesos, lo cual conlleva tiempo que si no se llevan a cabo adecuadamente podría ocasionar retrasos considerables en los proyectos, lo cual implica incremento de costos en el proyecto.
- Es un modelo aplicable muy bien para grandes proyectos de software, sin embargo se puede acomodar a proyectos medianos y pequeños sin mayores problemas, pero aquí surge una interrogante ¿será que los costos de producción son rentables? ,en el caso de los proyectos pequeños puede suceder que los costos que se generen con la planilla del equipo de profesionales necesarios sea alta y no le convenga a la empresa que ofrece el software.
- No cuenta con un plan para la estimación de costos del proyecto, que permitirá la determinación del presupuesto del proyecto. No es claro con técnicas o herramientas adecuadas para la estimación de los costos del proyecto dejando un hueco en este aspecto, el director del proyecto no tiene un artefacto claro que le permita monitorear y controlar los costos del proyecto.

5. CONCLUSIONES

- Ambos modelos no cuentan con los procedimientos claros y precisos para la gestión de calidad, gestión de riesgos lo que provoca que los directores de proyecto como equipo de desarrollo sin la experiencia debida no puedan enfrentar fácilmente el proceso de desarrollo con alguno de estos modelos analizados.
- EL modelo de WATCH tiene muchas similitudes con RUP porque rescata elementos básicos de esta última, incluyendo rasgos particulares con un enfoque genérico algo diferente.
- Todo modelo propuesto para enfrentar el proceso de desarrollo es importante frente a producir directamente código sin el modelado respetivo, ocasionando productos sin calidad y difíciles de mantener. Sin embargo a veces quedan huecos o vacíos que los modelos no consideran y que provocan en los analistas y diseñadores situaciones de conflicto debido a

la insuficiente información puntual que debería tener cada proceso implicado en el modelo como también los artefactos claramente definidos.

REFERENCIAS BIBLIOGRÁFICAS

- Avionics Software Engineering. Requirements ,1998]
- Booch, Grady, Jccobson, Ivar y Rumbauch, James. El Proceso Unificado de Desarrollo de Software, España: Pearson Educación, 2007. 688 p
- Booch, G.: Rumbaugh, J. y Jacobson, I.; "El Lenguaje Unificado de Modelado". 2000. Addison-Wesley.
- Falção, H. y Fontes, J.;"¿En quién se pone el foco?. Identificando stakeholders para la formulación de la misión organizacional". Revista del CLAD Reforma y Democracia". No. 15 (Octubre 1999). Caracas.
- Hans van Vliet, "Software Engineering: Principles and Practice", segunda edición, John Wiley & Sons, 2001.
- Sommerville Ian, 2005, "Ingeniería del Software", Sétima edición, México DF, Editorial Pearson.
- Pressman, Roger S. 2006, "Ingeniería del Software: Un enfoque práctico", Sexta edición, México
- Pressman, Roger; Ingeniería de software. Un enfoque práctico. 2002. McGraw.Hill/ Interamericana de España.
- Sommerville I.Ingenieria del Software,setp. Edición .Madrid 2005
- Von Hallen, B.;"Building a Business.

Páginas Web

- IEEE Std 610.12-1990, "IEEE Standard Glossary of Software Engineering Terminology",
- rhttp://standards.ieee.org/reading/ieee/std_public/description/se/610.12-1990_desc.html
- Rules". http://www.Kpiusa.com/ReadingRoom ReadingRoom.htm