UNIVERSIDAD TECNOLOGICA DEL CENTRO DE VERACRUZ

“El Proceso Unificado de Desarrollo de Software”.

Elaborado por: Estuardo Sánchez Gómez.
Matricula: 6675.
Grado y grupo: TICSI 6° A DZ.
Materia: Ingeniería de software I.
Profesor: Honorato Aguilar Galicia.
Lectura técnica. Resumen del capítulo 1 y capítulo 2.

28/05/2015

Índice.

Capítulo 1. El proceso unificado: dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental.

La tendencia actual del software lleva al desarrollo de sistemas más grandes y más complejos. Y el crecimiento del software viene dado a la necesidad de realizar software más sofisticado, adaptándose a nuestras necesidades, y va de la mano con el aumento de las diferentes tecnologías. Esto lleva a quererlo en el menor tiempo posible, pero esto no siempre se puede hacer. El problema se reduce a que los desarrolladores necesitan administrar las diferentes acciones para un gran proyecto, usando procesos que lo faciliten.

1.1. El proceso unificado en pocas palabras.

El proceso unificado es un proceso de desarrollo de software. Un proceso de desarrollo de software es el conjunto de actividades necesarias para transformar los requisitos de un usuario en un sistema software.

El proceso unificado utiliza el lenguaje unificado de modelado para preparar todos los esquemas de un sistema software.

El proceso unificado se resume en tres frases clave dirigido por casos de uso, centrado en la arquitectura, e iterativo e incremental.

1.2. El proceso Unificado está dirigid por casos de uso.

Un sistema software ve la luz para dar servicio a sus usuarios.

El termino usuarios representa alguien o algo que interactúa con el sistema que estamos desarrollando.

Un caso de uso es un fragmento de funcionalidad del sistema que proporciona al usuario un resultado importante, todos los casos de uso constituyen el modelo de casos de uso, el cual describe la funcionalidad total del sistema.

Los casos de uso guían la arquitectura del sistema y la arquitectura del sistema influye en la selección de los casos de uso.

1.3. El proceso unificado está centrado en la arquitectura.

La arquitectura es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles de lado.

El arquitecto debe realizar las siguientes tareas:

- Crea un esquema en borrador de la arquitectura, comenzando por la parte de la arquitectura que no es específica de los casos de uso.
- Trabaja con un subconjunto de los casos de uso especificados, con aquellos que representan las funciones clave del sistema en desarrollo.
- Se lleva a la maduración de más casos de uso.

Este proceso continúa hasta que se considere que la arquitectura es estable.

1.4. El proceso unificado es iterativo e incremental.

Es práctico dividir el trabajo en partes más pequeñas o miniproyectos (es una iteración que resulta en un incremento).

La iteración trata un grupo de casos de uso que juntos amplían la utilidad del producto desarrollado hasta ahora.

La iteración trata los riesgos más importantes.

Beneficios del proceso iterativo controlado:

- Reduce el coste del riesgo a los costes de un solo incremento.
- Reduce el riesgo de no sacar al mercado el producto en el calendario previsto.
- Acelera el ritmo del esfuerzo de desarrollo en su totalidad debido a que los desarrolladores trabajan de manera más eficiente para obtener resultados claros.
- Reconoce una realidad que a menudo se ignora, las necesidades del usuario y sus correspondientes requisitos no pueden definirse completamente al principio.

1.5. La vida del proceso unificado.

El proceso unificado se repite a lo largo de una serie de ciclos que constituyen la vida de un sistema. Cada ciclo concluye con una versión del producto.

Cada ciclo consta de cuatro fases: inicio, elaboración, construcción y transición.

1.5.1. El producto.

El producto terminado incluye los requisitos, caso de uso, especificaciones no funcionales y casos de prueba.

Los desarrolladores necesitan todas las representaciones del producto software:

- Un modelo de casos de uso, con todos los casos de uso y su relación con los usuarios.
- Un modelo de análisis, con dos propósitos: refinar los casos de uso con más detalles y establecer la asignación inicial de funcionalidad del sistema a un conjunto de objetos que proporcionan el comportamiento.
- Un modelo de diseño que define la estructura estática del sistema en la forma de subsistemas, clases e interfaces y los casos de uso reflejados como colaboradores entre los subsistemas, clases e interfaces.
- Un modelo de implementación, que incluye componentes, que representan el código de fuente y la correspondencia de las clases con los componentes.
- Un modelo de despliegue que define los nodos físico (ordenadores) y la correspondencia de los componentes con esos nodos.
- Un modelo de prueba, que describe los casos de prueba que verifican los casos de uso.
- Y, por supuesto, una representación de la arquitectura.

1.5.2. Fases dentro de un ciclo.

- Durante la fase de inicio, se desarrolla una descripción del producto final a partir de una buena idea y se presenta el análisis de negocios para el producto, esencialmente, esta fase responde a las siguientes preguntas:

    - ¿Cuáles son las principales funciones del sistema para sus usuarios más importantes?
    - ¿Cómo podría ser la arquitectura del sistema?
    - ¿Cuál es el plan de proyectos y cuánto costará desarrollar el producto?
- En la fase de elaboración, se especifican en detalle la mayoría de los casos de uso del producto y se diseña la arquitectura del sistema. La relación entre la arquitectura del sistema y del propio sistema es primordial. Una manera simple de expresarlo es decir que la arquitectura es análoga al esqueleto cubierto por la piel pero con muy poco músculo (el software) entre los huesos y la piel –solo lo necesario para permitir que el esqueleto haga movimientos básicos. Al final de la fase de elaboración, el director de proyecto está en disposición de planificar las actividades y estimar los recursos necesarios para terminar el proyecto.
- En la fase de construcción se crea el producto se añaden los músculos (software terminado) al esqueleto (la arquitectura). En esta fase, la línea base de la arquitectura crece hasta convertirse en el sistema completo. La descripción evoluciona hasta convertirse en un producto preparado para ser entregado a la comunidad de usuarios.
- La fase de transición cubre el periodo durante el cual el producto se convierte en versión beta. En la versión beta un número reducido de usuarios con experiencia prueba el producto e informa de defectos y deficiencias. Los desarrolladores corrigen los problemas e incorporan algunas de las mejoras sugeridas en una versión general dirigida a la totalidad de la comunidad de usuarios.

1.6. Un proceso integrado.

El proceso unificado está basado en componentes y ha establecido un marco de trabajo que integra todas esas diferentes facetas. El marco de trabajo sirve también como paraguas bajo el cual los fabricantes de herramientas y los desarrolladores pueden construir herramientas que soporten la automatización del proceso entero.

Capítulo 2. Las cuatro “P” en el desarrollo de software: Personas, Proyecto, Producto y Proceso.

El resultado final de un proyecto software es un producto que toma forma durante su desarrollo gracias a la intervención de muchos tipos distintos de personas.

- Personas: Los principales autores de un proyecto software son los arquitectos, desarrolladores, ingenieros de prueba, y el personal de gestión que les da soporte, además de los usuarios, clientes, y otros interesados.
- Proyecto: Elemento organizado a través del cual se gestiona el desarrollo de software.
- Producto: Artefactos que se crean durante la vida del proyecto, como los modelos, código fuente, ejecutables y documentación.
- Proceso: Un proceso de ingeniería de software es una definición del conjunto completo de actividades necesarias para transformar los requisitos de usuarios en un producto.
- Herramientas: Software que se utiliza para automatizar las actividades definidas en el proceso.

2.1. Las personas son decisivas.

Las personas financias el producto, lo planifican, lo desarrollan, lo gestionan, lo prueban, lo utilizan y se benefician de él. El proceso que guía este desarrollo debe funcionar bien para las personas que lo utilizan.

2.1.1. Los procesos de desarrollo afectan a las personas.

Conceptos como la viabilidad, la gestión del riesgo, la organización de los equipos, la planificación del proyecto y la facilidad de comprensión del proyecto tienen un papel importante:

- Viabilidad del proyecto: Una aproximación iterativa en el desarrollo permite juzgar pronto la viabilidad del proyecto. Los proyectos que no son viables pueden detenerse en una fase temprana.
- Gestión del riesgo: La exploración de los riesgos significativos en las primeras fases atenúa este problema.
- Estructura de los equipos. Una buena arquitectura, o interfaces bien definidas entre subsistemas y componentes hace posible una división del esfuerzo de este tipo.
- Planificación del proyecto: Las técnicas que se utilizan en las fases de inicio y de elaboración permiten a los desarrolladores tener una buena noción de cuál debería ser el resultado del proyecto.
- Facilidad de comprensión del proyecto: La descripción de la arquitectura proporciona una visión general para cualquiera que se encuentre implicado en el proyecto.
- Sensación de cumplimiento: La retroalimentación frecuente y las conclusiones obtenidas aceleran el ritmo de trabajo.

2.1.2. Los papeles cambiarán.

Para comprender y dar soporte a esos procesos de negocios más complejos y para implementarlos en software, los desarrolladores deberán trabajar con muchos otros desarrolladores. Para trabajar eficazmente en equipos cada vez más grandes, se necesita un proceso que sirva como guía. Esta guía tendrá como resultado desarrolladores “que trabajan de manera más inteligente”, es decir, que limitan sus esfuerzos individuales a aquello que proporcione un valor al cliente.

En los años venideros, la mayoría de los informáticos van a empezar a trabajar más estrechamente con sus objetivos, y serán capaces de desarrollar software más complejo gracias a un proceso automatizado y a los componentes reutilizables.

2.1.3. Convirtiendo “recursos” en “trabajadores”.

Cada trabajador es responsable de un conjunto completo de actividades, como las actividades necesarias para el diseño de un subsistema. Necesitan comprender cuáles son sus roles en relación con los de otro trabajadores. Las herramientas no sólo deben ayudar a los trabajadores a llevar a cabo sus propias actividades, también deben aislarles de la información que no les sea relevante.

2.2. Los proyectos construyen el producto.

- Una secuencia de cambio: Los proyectos de desarrollo de sistemas obtienen productos como resultados, pero el camino hasta obtenerlos es una serie de cambios.
- Una serie de iteraciones: Dentro de cada fase de un ciclo, los trabajadores llevan a cabo las actividades de la fase a través de una serie de iteraciones.
- Un patrón organizado: Indica los tipos de trabajadores que el proyecto necesita y los artefactos por los cuales hay que trabajar.

2.3. El proceso es más que código.

El producto que se obtiene es un sistema software, hace referencia al sistema entero, y no sólo al código que se entrega.

2.3.1. ¿Qué es un sistema software?

Un sistema es todos los artefactos que se necesitan para representarlo en una forma comprensible por máquinas u hombres, para las maquinas, los trabajadores y los interesados.

2.3.2. Artefactos.

Es un término general para cualquier tipo de información creada, producida, cambiada o utilizada por los trabajadores en el desarrollo del sistema.

2.3.3. Un sistema posee una colección de modelos.

La construcción de un sistema es por tanto un proceso de construcción de modelos, utilizando distintos modelos para describir todas las perspectivas diferentes del sistema. El proceso unificado proporciona un conjunto de modelos cuidadosamente seleccionado con el cual comenzar.

2.3.4. ¿Qué es un modelo?

Un modelo es una abstracción del sistema, especificando el sistema modelado desde un cierto punto de vista y en un determinado nivel de abstracción. Un punto de vista es, una vista de especificación o una vista de diseño del sistema.

Los modelos son abstracciones del sistema que construyen los arquitectos y desarrolladores.

2.3.5. Cada modelo es una vista autocontenida del sistema.

Un modelo es una abstracción semánticamente cerrada del sistema. Es una vista autocontenida en el sentido de que un usuario de un modelo no necesita para interpretarlo más información.

La mayoría de los modelos de ingeniería se definen mediante un subconjunto de UML cuidadosamente seleccionado.

2.3.6. Dentro de un modelo.

Un modelo siempre identifica el sistema que está modelando. Este elemento del sistema es por tanto el contenedor de los demás elementos. En el modelo de diseño de cada subsistema puede ser contenedor de construcciones similares.

2.3.7. Relaciones entre modelos.

Un sistema contiene todas las relaciones y restricciones entre elementos incluidos en diferentes modelos.

Existen trazas entre las colaboraciones del modelo de diseño y las colaboraciones en el modelo de análisis, y entre los componentes en el modelo de implementación y los subsistemas en el modelo de diseño.

2.4. El proceso dirige los proyectos.

Proceso es un término demasiado utilizado. Proceso unificado, se refiere a los procesos "de negocio" claves en una empresa de desarrollo de software.

2.4.1. El proceso: una plantilla.

Proceso hace referencia a un contexto que sirve como plantilla que puede reutilizarse para crear instancias de ellas.

Un proceso es una definición de un conjunto de actividades, no su ejecución.

2.4.2. Las actividades relacionadas conforman flujos de trabajo.

Describimos el proceso entero en partes llamadas flujos de trabajo, en términos UML, un flujo de trabajo es un estereotipo de colaboración, en el cual los trabajadores y los artefactos son los participantes.

2.4.3. Procesos especializados.

Los factores principales que influyen en cómo se diferenciará el proceso son:

- Factores organizativos: La estructura organizativa, la cultura de la empresa, la organización y la gestión del proyecto, las aptitudes y habilidades disponibles, experiencias previas y sistemas software existentes.
- Factores del dominio: El domino de la aplicación, procesos de negocio que se deben soportar, la comunidad de usuarios y las ofertas disponibles de la competencia.
- Factores del ciclo de vida: El tiempo de salida al mercado, el tiempo de vida esperando del software, la tecnología y la experiencia de las personas en el desarrollo de software, y las versiones planificadas y futuras.
- Factores técnicos: Lenguaje de programación, herramientas de desarrollo, base de datos, marcos de trabajo y arquitecturas "estándar" subyacentes, comunicaciones y distribución.

2.4.4. Méritos del proceso.

- Todo el mundo en el equipo de desarrollo puede comprender lo que tiene que hacer para desarrollar el producto.
- Los desarrolladores pueden comprender mejor lo que los otros están haciendo.
- Los supervisores y directores, incluso los que no entienden el código, pueden comprender lo que los desarrolladores están haciendo gracias a los esquemas de la arquitectura.
- Los desarrolladores, los supervisores y los directores pueden cambiar de proyecto o de división sin tener que aprender un nuevo proceso.
- La información puede estandarizarse dentro de la empresa, y puede obtenerse de compañeros o de cursos breves.
- El devenir del desarrollo del software es repetible, puede planificarse y estimarse en coste con suficiente exactitud como para cumplir las expectativas.

2.5. Las herramientas son esenciales en el proceso.

Soportan los procesos de desarrollo de software modernos. El proceso y las herramientas vienen en el mismo paquete.

2.5.1. Las herramientas influyen en el proceso.

Las herramientas son buenas para automatizar procesos repetitivos y para guiarnos a lo largo de un camino de desarrollo concreto.

2.5.2. El proceso dirige las herramientas.

Las herramientas que implementan un proceso automatizado deben ser fáciles de usar, la herramienta debe ser sencilla de comprender y manejar para los trabajadores, debe proporcionar un incremento de productividad sustancial.

2.5.3. El equilibrio entre el proceso y las herramientas.

Debe haber un equilibrio entre proceso y herramientas. El desarrollo del proceso y su soporte por herramientas debe tener lugar de manera simultánea y en cada versión del proceso debe haber también una versión de las herramientas.

2.5.4. El modelado visual soporta UML.

UML define reglas sintácticas que especifican cómo combinar elementos del lenguaje, la herramienta debe ser capaz de garantizar que se cumplen esas reglas. La herramienta de modelado debe incorporar más que conocimientos de UML.

2.5.5. Las herramientas dan soporte al ciclo de vida completo.

- Gestión de requisitos: Se utilizan para almacenar, examinar, revisar, hacer el seguimiento y navegar por lo diferentes requisitos de un proyecto software.
- Modelado visual: Se utiliza para automatizar el uso de UML, para modelar y ensamblar una aplicación visualmente.
- Herramientas de programación: Proporciona una gama de herramientas, incluyendo editores, compiladores, depuradores, detectores de errores y analizadores de rendimiento.
- Aseguramiento de calidad: Se utiliza para probar aplicaciones y componentes, para registrar y ejecutar casos de prueba que dirigen la prueba de un IGU y de las interfaces de un componente.

Bibliografía.

Jacobson I., Booch G., Rumbaugh J. (2000). *El proceso unificado de desarrollo de Software*. Madrid. Addison Wesley.