Unidad 2 / Escenario 4

Lectura fundamental

Proceso Unificado de Desarrollo de Software

Contenido

Palabras clave: Software, Desarrollo de Software, Ingeniería del Software, Metodologías de Desarrollo de Software, RUP.

1. Proceso Unificado de Desarrollo de Software

El Proceso Unificado de Rational RUP por sus siglas en inglés (Rational Unified Process), es una metodología de desarrollo de software que integra todos los aspectos que se deben tener en cuenta durante el ciclo de vida de del software y busca ser aplicado en proyectos, tanto grandes como pequeños de software (Jacobson, I., Booch,G., y Rumbaugh, J.,2000).

1.1. Características principales de RUP

Las principales características de RUP son las siguientes:

- **Dirigido por casos de uso:** la razón de ser de un sistema software es servir a usuarios, ya sean humanos u otros sistemas. Un caso de uso es una facilidad que el software debe proveer a sus usuarios. Los casos de uso reemplazan la antigua especificación funcional tradicional y constituyen la guía fundamental establecida para las actividades a realizar durante todo el proceso de desarrollo incluyendo el diseño, la implementación y las pruebas del sistema.
- **Centrado en arquitectura:** la arquitectura involucra los elementos más significativos del sistema y está influenciada, entre otros, por plataformas software, sistemas operativos, manejadores de bases de datos, protocolos, consideraciones de desarrollo como sistemas heredados y requerimientos no funcionales. Es como una radiografía del sistema que estamos desarrollando, lo suficientemente completa como para que todos los implicados en el desarrollo tengan una idea clara de qué es lo que están construyendo.
- **Iterativo e Incremental:** Iterativo e Incremental: para hacer más manejable un proyecto se recomienda dividirlo en ciclos. Para cada ciclo se establecen fases de referencia, cada una de las cuales debe ser considerada como un mini proyecto cuyo núcleo fundamental está constituido por una o más iteraciones de las actividades principales básicas de cualquier proceso de desarrollo. En concreto RUP divide el proceso en cuatro fases, dentro de las cuales se realizan varias iteraciones en número variable según el proyecto y en las que se hace un mayor o menor hincapié en las distintas actividades (Martínez, 2000).

1.2. Mejores prácticas de la Ingeniería del Software implementadas en RUP

El principal objetivo de RUP es asegurar la producción de software de calidad dentro plazos y presupuestos predecibles para lo cual implementa las siguientes buenas o mejores prácticas en ingeniería de software (Larman, 2003, p.18):

- **Desarrollo iterativo:** como se vio anteriormente, una de las principales características de RUP es que implementa un flujo de procesos iterativo e incremental lo cual se considera una buena práctica de ingeniería de software ya que los proyectos de software son cada vez más complejos. El desarrollo iterativo permite ir comprendiendo los requerimientos a la vez que se va haciendo crecer el sistema. Además, este flujo permite abordar las tareas que generan mayor riesgo primero, con esto se logra reducir el riesgo del proyecto y tener un subsistema ejecutable lo más temprano posible.
- **Administración de requerimientos:** esta metodología describe como levantar requerimientos, organizarlos, documentaros, rastrearlos, documentar las decisiones de diseño y lo más importante, como comunicar los requerimientos del negocio.
- **Arquitectura basada en componentes:** de esta manera se divide el sistema en componentes con interfaces bien definidas que luego serán ensamblados para obtener el sistema a medida que se van desarrollando o madurando sus partes.
- **Modelado visual de software:** RUP utiliza como única herramienta de modelado el lenguaje unificado de modelado (UML, por sus siglas en inglés, Unified Modeling Language) con el cual se hace un modelamiento visual de la estructura y el comportamiento de la arquitectura y sus componentes. Además, se usan los bloques de construcción que facilitan la especificación de los detallas permitiendo mostrar u ocultar el nivel de detalle de la arquitectura, facilita la comunicación en el equipo de desarrollo y permite analizar la consistencia en los componentes y entre lo diseñado y o implementado.
- **Verificación de la calidad del software:** RUP hace énfasis en que no sólo la funcionalidad es importante, también es importante verificar cualidades no funcionales como el rendimiento y la confiabilidad del software. RUP ayuda a planificar, diseñar, implementar, ejecutar y evaluar pruebas que verifiquen estas cualidades.
- **Control de Cambios:** los cambios son inevitables, pero es necesario evaluar si estos son necesarios y rastrear su impacto. RUP indica cómo controlar, rastrear y monitorear los cambios dentro del proceso iterativo de desarrollo.

Para implementar estas buenas prácticas, RUP se estructura en cuatro componentes principales:

1. Roles: que da respuesta a la pregunta ¿Quién? Un rol define el comportamiento y responsabilidades de uno más individuos. Estas responsabilidades se definen en términos de las actividades que debe desarrollar y los productos que requiere generar. Una persona puede ejecutar más de un rol en un momento dado, y un rol puede ser ejecutados por más de una persona.
2. Actividades: que da respuesta a la pregunta ¿Cómo? Una actividad es una unidad de trabajo específica que se asigna a un rol, generalmente se define en términos de crear o modificar un producto o artefacto.
3. Productos: que da respuesta a la pregunta ¿Qué? Son elementos de información producidos, modificados o usados por el proceso. Son los resultados tangibles del proceso, ejemplo de estos son: modelos, documentos, unidades de código, entre otros.
4. Flujos de trabajo: que da respuesta a la pregunta ¿Cuándo? Un flujo de trabajo es una secuencia de actividades que produce un resultado valioso. En RUP se distinguen dos tipos de flujo de trabajo: los de proceso y los de apoyo.

Si recuerdan, en el escenario 1 de la unidad 1, se mencionó que estas preguntas son fundamentales en la definición del proceso y RUP define claramente como darles respuesta. Las relaciones entre estos elementos se ilustran en la Figura 1. Relación entre: roles, actividades, productos y flujos de trabajo.

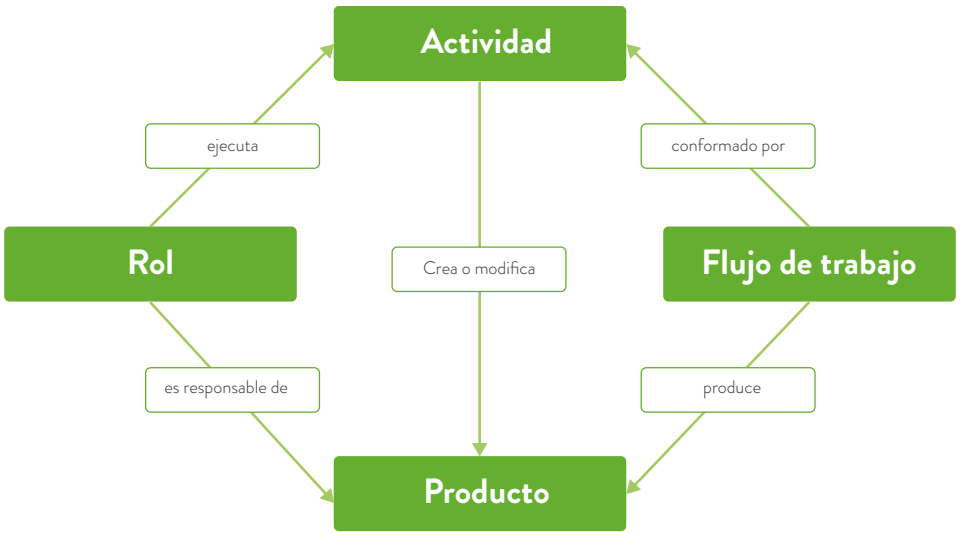

Figura 1. Relación entre roles, actividades, productos y flujos de trabajo

Fuente: elaboración propia

1.3. Ciclos y fases

En RUP se divide el proceso de desarrollo en ciclos, teniendo un producto al final de cada ciclo. Cada ciclo se divide en cuatro fases secuenciales: inicio, elaboración, construcción y transición, cada una de estas fases concluyen con un producto intermedio. A continuación, veremos las características de cada una de las fases.

1.3.1. Fase de Inicio

Objetivos de esta fase, de acuerdo con Jacobson, Booch y Rumbaugh (2000), son:

- Establecer el ámbito del proyecto y sus límites.
- Encontrar los casos de uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
- Estimar el coste en recursos y tiempo de todo el proyecto.
- Estimar los riesgos y las fuentes de incertidumbre.

Para esto, se desarrollan las siguientes actividades:

- Identificar los actores con los que se interactúa y definir esta interacción a un alto nivel de abstracción.
- Se identifican todas las entidades externas con las que se trata (actores) y se define la interacción a un alto nivel de abstracción.
- Se identifican todos los casos de uso y describen algunos en detalle, los restantes se describirán en la siguiente fase.
- Describir la oportunidad de negocio incluyendo:
  - criterios de éxito
  - identificación de riesgos
  - estimación de recursos necesarios
  - Plan de fases incluyendo los hitos

Los productos de esta fase son los siguientes:

- Visión del negocio: Describe los objetivos y restricciones a alto nivel.
- Modelo de casos de uso.
- Especificación adicional: requisitos no funcionales.
- Glosario: Terminología clave del dominio.
- Lista de riesgos y planes de contingencia.
- Establecer el ámbito del proyecto y sus límites.
- Encontrar los casos de uso críticos del sistema, los escenarios básicos que definen la funcionalidad.
- Estimar el coste en recursos y tiempo de todo el proyecto.
- Estimar los riesgos, las fuentes de incertidumbre.

No todos los productos son obligatorios ni deben terminarse en un 100% al final de esta fase. Para dar por terminada esta fase se debe verificar que las partes interesadas han acordado el alcance de la iteración y la estimación de recursos requeridos, para lo cual es indispensable haber identificado, documentado y validado los requerimientos a través de casos de uso.

1.3.2. Fase de Elaboración

Continuando con lo definido por Jacobson, Booch y Rumbaugh (2000), vemos que las principales características de esta fase son las siguientes:

Objetivos de la fase:

- Definir, validar y cimentar la arquitectura.
- Completar la definición de la visión del proyecto y la iteración.
- Crear un plan fiable para la fase de construcción. Este plan puede evolucionar en sucesivas iteraciones. Este plan debe incluir los costes si procede.

- Demostrar que la arquitectura propuesta soportará la visión con un coste razonable y en un tiempo razonable.

Los productos que se deben generar en esta fase son los siguientes:

- Modelo de casos de uso (80% completo) con descripciones detalladas.
- Otros requerimientos no funcionales o no asociados a casos de uso.
- Descripción de la Arquitectura del Software.
- Un prototipo ejecutable de la arquitectura.
- Lista revisada de riesgos y del caso de negocio.
- Plan de desarrollo para el resto del proyecto.
- Un manual de usuario preliminar.

Es importante considerar que a partir de esta etapa los requerimientos y los planes de desarrollo deben ser estables, para dar por terminada esta fase se debe contar una arquitectura que contemple los casos de uso críticos y los riesgos identificados y se demuestre que estos han sido abordados y todas las personas involucradas están de acuerdo con la arquitectura propuesta.

1.3.3. Fase de Construcción

El objetivo de esta fase es alcanzar la capacidad operacional del producto. Todos los componentes, características y requisitos, que no hayan sido hechos hasta ahora, deben ser implementados, integrados y probados, hasta lograr una versión del producto que se pueda poner en manos de los usuarios. En esta fase es fundamental cuidar la calidad del producto mientras se optimizan los recursos incluido el recurso humano y el tiempo. Los productos que se deben generar en esta fase son los siguientes:

- El producto de software integrado y corriendo en la plataforma adecuada.
- Manuales de usuario.

- Una descripción del release actual. El release, es una versión que se distribuye a los clientes y que se distingue de las anteriores o de otras versiones porque tiene nuevas funcionalidades implementadas o porque son versiones que se ejecutan en plataformas de hardware diferentes. También se habla de Software Release Management en español: gestión de entregas de Software al proceso de publicar las actualizaciones del software.

1.3.4. Fase de Transición

El objetivo de esta fase es entregar el software desarrollado a la comunidad de usuarios, lo cual incluye:

- Pruebas Beta para validar el producto con las expectativas del cliente.
- Ejecución paralela con sistemas antiguos.
- Conversión de información de sistemas legado si fuera necesario.
- Entrenamiento de usuarios para garantizar el uso y apropiación del software.

Para saber que esta fase ha terminado con éxito, es necesario garantizar que los usuarios ya pueden usar el software de manera autosuficiente y se sienten satisfechos con los logros del producto en coherencia con el alcance definido en la fase de inicio.

1.4. Flujos de trabajo

En RUP se definen nueve flujos de trabajo distintos que se desarrollan durante la ejecución de las fases mencionadas anteriormente y se agrupan como se muestra en la Figura 2. Clasificación de los flujos de trabajo.

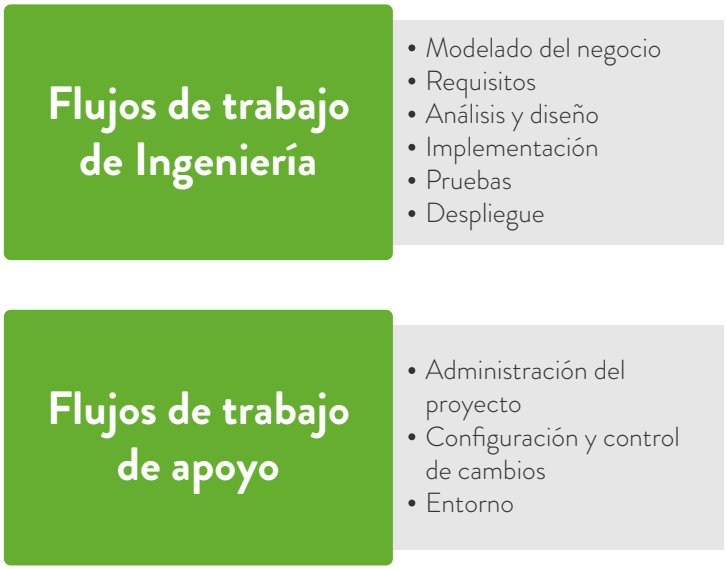

Figura 2. Clasificación de los flujos de trabajo según Jacobson, I., Booch,G., y Rumbaugh, J. 2000

Fuente: elaboración propia

Si bien es cierto que los nombres de los flujos de trabajo de ingeniería son similares a las actividades que vimos en los modelos de procesos, estos flujos se ejecutarán durante el desarrollo de cada una de las fases. A continuación, se verá de qué se hace cargo cada uno de estos flujos de trabajo.

1.4.1. Modelado de Negocio

Este flujo de trabajo busca que garantizar el mejor entendimiento de la organización o contexto en donde se va a implantar el software de manera que se asegure que será útil y que se adapte de la mejor manera posible. Además, se busca que ese entendimiento de un marco común a los desarrolladores, los clientes y los usuarios finales.

1.4.2. Requisitos

Este flujo de trabajo se hace cargo de que los desarrolladores y los clientes acuerden qué debe hacer el sistema para esto se ejecutan las siguientes actividades:

- Levantar (también conocido como relevar) requerimientos
- Documentar funcionalidades y restricciones
- Documentar decisiones
- Identificar actores
- Identificar casos de uso

Se deben identificar tanto los requisitos funcionales, es decir, ¿Qué hace el sistema? Los cuales se especifican mediante casos de uso, y los requisitos no funcionales, es decir, ¿Cómo lo hace? ¿Qué características tiene el software? Como disponibilidad, rendimiento, seguridad entre otros. Para esto se incluyen especificaciones complementarias.

1.4.3. Análisis y diseño

El objetivo de este flujo de trabajo es traducir los requisitos a una especificación que describa cómo implementar el sistema. De acuerdo con Jacobson, Booch y Rumbaugh (2000), esta especificación debe garantizar que el sistema pueda implementarse sin ambigüedades. Este diseño debe basarse en la noción de arquitectura y el modelo debe constar de los siguientes elementos como mínimo:

- Clases estructuradas en paquetes
- Diseños de subsistemas con interfaces definidas (componentes)
- Forma de colaboración entre las clases

1.4.4. Implementación

En este flujo de trabajo se implementan todas las unidades de código y artefactos requeridos para tener un sistema ejecutable y se hacen las pruebas unitarias, es decir, que cada responsable de implementar una unidad debe probarla. Adicionalmente, cada componente se debe ntegrar al sistema.

1.4.5. Pruebas

Este flujo de trabajo es el encargado de evaluar la calidad del producto que estamos desarrollando, para lo cual se debe verificar lo siguiente:

- La interacción entre los objetos
- La integración apropiada de componentes
- Que se satisfacen los requerimientos
- Identificar los defectos y corregirlos antes de la instalación

1.4.6. Despliegue

El propósito de este flujo de trabajo es lograr que el software llegue a los usuarios finales, para lo cual incluye las siguientes actividades:

- Producir un “release”
- Empaquetar el software
- Distribuir el software
- Instalar el software
- Preparar a los usuarios

La mayor parte de este flujo de trabajo se desarrolla durante la fase de transición.

1.4.7. Administración del proyecto

El objetivo de la administración de un proyecto es conseguir equilibrar el completar los objetivos, administrar el riesgo y superar las restricciones para desarrollar un producto que sea acorde a los requisitos de los usuarios. RUP incluye las siguientes herramientas para la administración del proyecto:

- Un framework o marco de trabajo estandarizado para manejo de proyectos de software
- Guías para planificación, provisión de personal, ejecución y monitoreo de planes
- Un framework o marco de trabajo estandarizado para manejar riesgos

1.4.8. Configuración y control de cambios

El propósito de este flujo de trabajo es garantizar la integridad de todos los productos que se crean en el proceso y mantener la rastreabilidad de su proceso de creación para evitar actualizaciones simultáneas o múltiples versiones de los productos. Para lograr esto RUP ofrece guías para:

- Desarrollos en paralelo
- Automatizar la construcción
- Administrar defectos

1.4.9. Entorno o Ambiente

El objetivo de este flujo de trabajo es tener a punto las herramientas que se vayan a necesitar en cada momento, así como definir la instancia concreta de proceso unificado que se va a seguir. Para lo cual se desarrollan las siguientes actividades:

- Selección y adquisición de herramientas
- Establecer y configurar las herramientas para que se ajusten a la organización
- Configuración del proceso
- Mejora del proceso
- Servicios técnicos

En el transcurso de este escenario vimos la organización y los elementos de alto nivel de RUP, sin embargo, durante las siguientes unidades podremos profundizar las herramientas específicas que ofrece para RUP para desarrollar las actividades de cada flujo de trabajo. La Figura 3. Fases y flujos de trabajo fundamentales de RUP muestra la relación de en qué momento se ejecutan estos flujos de trabajo de acuerdo con las fases descritas anteriormente, en cada iteración. Como se puede observar, los flujos de trabajo se desarrollan de manera paralela, haciéndose énfasis en unos más que en otros en cada fase. Por ejemplo, el flujo de trabajo requisitos se puede desarrollar en paralelo todos los demás flujos, sin embargo, se un mayor énfasis en este flujo de trabajo en la fase de elaboración, con menor intensidad en la fase de inicio, con poca intensidad en la fase de construcción y está ausente en la fase de transición.

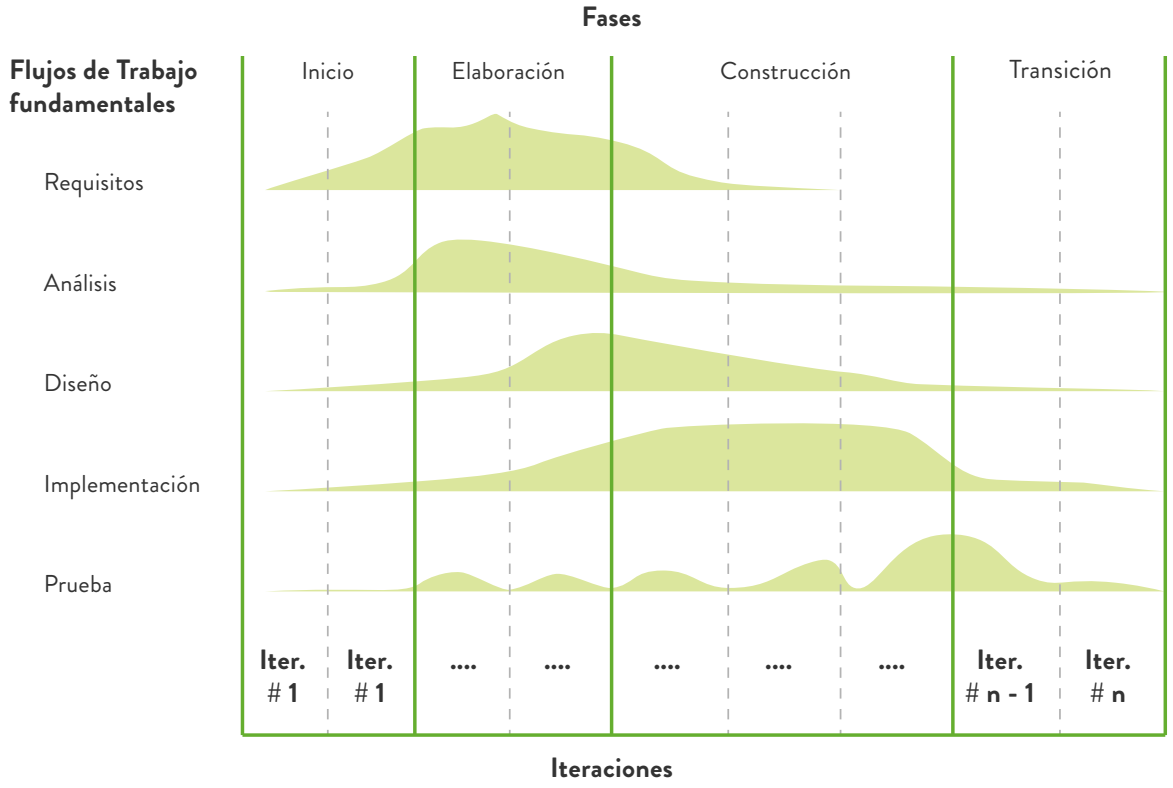

Figura 3. Fases y flujos de trabajo fundamentales de RUP según Jacobson, I., Booch,G., y Rumbaugh, J. 2000

Fuente: elaboración propia, modificado de Jacobson (2000)

En este documento se ha dado una mirada general a lo qué es RUP, mencionando sus tres características fundamentales, es decir: ser una metodología dirigida por casos de uso, centrada en arquitectura e ser iterativo e incremental. Además de su estructura bidimensional dividiendo el proceso en fases y éstas en flujos de trabajo, a través de los cuáles se proponen herramientas para implementar seis buenas prácticas de la ingeniería del software: desarrollo iterativo, administración de requerimientos, arquitectura basada en componentes, modelado visual de software, verificación de la calidad el software y control de cambios.

- Como mejorar: RUP es una metodología muy completa que intenta abarcar todos los aspectos del desarrollo de software. Sin embargo, de acuerdo con el tamaño y complejidad del proyecto, se puede elegir el grado de formalidad a la hora de utilizar los distintos artefactos que propone. En ***http://www.upedu.org/*** se pueden encontrar consejos sobre qué artefactos utilizar y cómo hacerlo, en un entorno educativo.

Referencias

Jacobson, I., Booch,G., y Rumbaugh, J. (2000). El proceso unificado de desarrollo de software. Madrid, España: Adisson Wesley.

Jacobson, I., Booch,G., y Rumbaugh, J. (2000). Figura 1.5. Los cinco flujos de trabajo – requisitos, análisis, diseño, implementación y prueba- tienen lugar sobre las cuatro fases: inicio, elaboración, construcción y transición. [Figura] Recuperado de: El proceso unificado de desarrollo de software. Madrid, España: Adisson Wesley

Larman, C. (2003). UML y patrones. Una introducción al análisis y diseño orientado a objetos y al proceso unificado. Madrid, España: Pearson Educación. S.A.

Martínez, A., Martínez, Raúl. (2000). Guía a Rational Unified Process.

INFORMACIÓN TÉCNICA

FACULTAD DE
INGENIERÍA, DISEÑO
E INNOVACIÓN

Módulo: Ingeniería del Software I

Unidad 2: Metodologías de Desarrollo de Software

Escenario 4: Proceso Unificado de Desarrollo de Software

Autor: Diana Angélica Cruz Ortega

Asesor Pedagógico: Luisa Esperanza Rincón Jiménez

Diseñador Gráfico: Cristian Navarro

Asistente: Ginna Quiroga