El Proceso Unificado de Desarrollo de Software
Ivar Jacobson
Grady Booch
James Rumbaugh
Addison Wesley
Resumen Capítulo 1. El proceso unificado: dirigido por casos de uso, centrado en la arquitectura, iterativo e incremental

La tendencia actual en el software lleva a la construcción de sistemas más grandes y más complejos. Esto es al hecho de que los computadores son más pequeños potentes cada año, y los usuarios, esperan más de ellos. Queremos un software que esté mejor adaptado a nuestras necesidades. Pero esto, a su vez, simplemente hace el software más complejo.También lo queremos más rápido, el tiempo de salida al mercado es otro conductor importante.

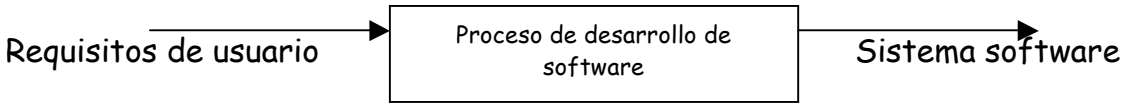
La mayoría de la gente desarrolla software mediante los mismos métodos que llevan utilizándose desde hace 25 años. A menos que renovemos nuestros métodos, no podremos cumplir con el objetivo de desarrollar el software complejo que necesita actualmente.

El problema del software se reduce a la dificultad que afrontan los desarrolladores para coordinar las múltiples cadenas de trabajo de un gran proyecto de software. Se necesita un proceso que integre las múltiples facetas del desarrollo. Se necesita un método común, un proceso que:

- Proporcione una guía para ordenar las actividades de un equipo
- Dirija las tareas de cada desarrollador por separado y del equipo como un todo
- Especifique los artefactos que deben desarrollarse
- Ofrezca criterios para el control y la medición de los productos y actividades del proyecto.

El proceso unificado en pocas palabras

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 es un marco de trabajo genérico que puede especializarse para una gran variedad de sistemas de software.



El proceso unificado está basado en *componentes*, lo cual quiere decir que el sistema software en construcción está formado por componentes software interconectados a través de interfaces bien definidas.

El proceso unificado está dirigido por *casos de uso*

El término *usuario* no sólo hace referencia a usuarios humanos sino a otros sistemas. En este sentido, el término *usuario* representa alguien o algo (como otro sistema fuera del sistema en consideración) 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. Los casos de uso representan *los requisitos funcionales*. Puede decirse que una *especificación funcional* contesta a la pregunta ¿Qué debe hacer el sistema? La estrategia de los casos de uso puede describirse añadiendo tres palabras al final de esta pregunta ¿… para cada usuario? Nos fuerzan a pensar en términos de importancia para el usuario y no solo en términos de funciones que sería bueno tener. Los casos de uso, también guían su diseño, implementación y prueba: esto es, guían el proceso de desarrollo. Basándose en el modelo de casos de uso, los desarrolladores crean una serie de modelos de diseño e implementación que llevan a cabo los casos de uso.

Dirigido por casos de uso quiere decir que el proceso de desarrollo sigue un hilo –avanza a través de una serie de flujos de trabajo que parten de los casos de uso. Los casos de uso se especifican, se diseñan y los casos de uso finales son la fuente a partir de la cual los ingenieros de prueba construyen sus casos de prueba.

El proceso unificado está centrado en la arquitectura

La *arquitectura* en un sistema software se describe mediante diferentes visitas del sistema en construcción. El concepto de arquitectura software incluye los aspectos estáticos y dinámicos del sistema, también se ve influida por muchos otros factores, como la plataforma en la que tiene que funcionar el

software, arquitectura hardware, sistema operativo, sistema de gestión de base de datos, protocolos para comunicarse en red, los bloques de construcción reutilizables de que se disponen consideraciones de implantación, sistemas heredados y requisitos no funcionales.

¿Cómo se relacionan los casos de uso y la arquitectura? Estas dos fuerzas deben equilibrase para obtener un producto con éxito, la función corresponde a los casos de uso y la forma a la arquitectura. Debe haber interacción entre los casos de uso y la arquitectura. En realidad, tanto la arquitectura, como los casos de uso deben evolucionar en paralelo. La arquitectura es la que debe diseñarse para permitir que el sistema evolucione, no solo en su desarrollo inicial, sino también a lo largo de las futuras generaciones. Podemos decir que el arquitecto:

- Crea un esquema en borrador de la arquitectura, comenzando por la parte de la arquitectura, que no especifica de los casos de uso (por ejemplo: la plataforma).
- A continuación, el arquitecto trabaja con un subconjunto de los casos de uso especificados, con aquellos que representen las funciones clave del sistema en desarrollo. Cada caso de uso seleccionado se especifica en detalle y se realiza en términos de subsistemas, clases y componentes.
- A medida que los casos de uso se especifican y maduran, se descubre más de la arquitectura. Esto, a su vez, lleva a la maduración de más casos de uso.

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

El proceso unificado es iterativo e incremental

Los desarrolladores basan la selección de lo que se implementará en una iteración en dos factores. En primer lugar, la iteración trata un grupo de casos de uso que juntos amplían la utilidad del producto desarrollado hasta ahora. En segundo lugar, la iteración trata los riesgos más importantes.

En las primeras fases del ciclo de vida, los desarrolladores pueden tener que reemplazar un diseño superficial por uno o más detallado o sofisticado. En fases posteriores, los incrementos son típicamente aditivos.

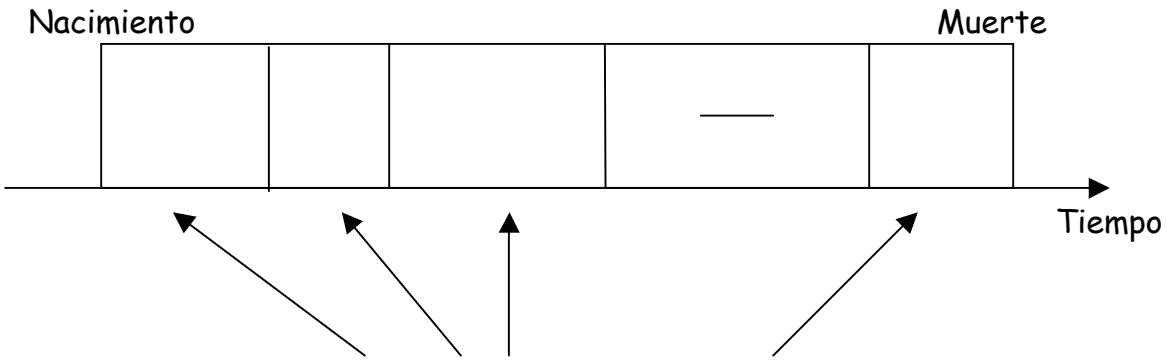
Los beneficios de un proceso iterativo controlado:

- La iteración controlada reduce el coste del riesgo a los costes de un solo incremento. Si los desarrolladores tienen que repartir la iteración. La organización sólo pierde el esfuerzo mal empleado de la iteración, no el valor del producto entero.
- La iteración controlada reduce el riesgo de no sacar al mercado el producto en el calendario previsto. Mediante la identificación de riesgos en fases tempranas del desarrollo, el tiempo que se gasta en resolverlos se emplea al principio de la planificación, cuando la gente está menos presionada por cumplir los plazos.
- La iteración controlada acelera el ritmo del esfuerzo de desarrollo en su totalidad debido a que los desarrolladores trabajan de manera más eficiente para obtener resultados claros a corto plazo, en lugar de tener un calendario largo, que se prolonga eternamente.
- La iteración controlada reconoce una realidad que a menudo se ignora – que las necesidades del usuario y sus correspondientes requisitos no pueden definirse completamente al principio.

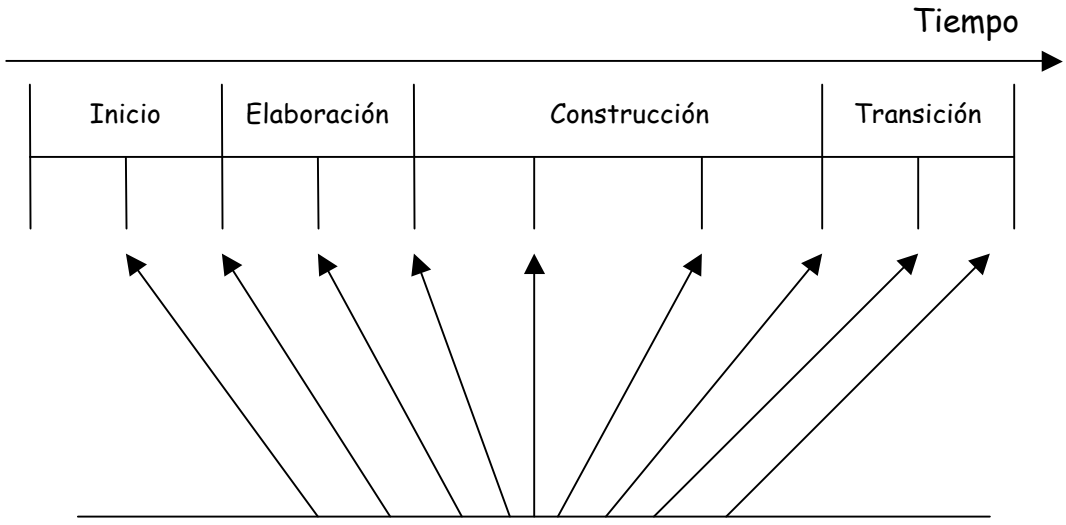
La arquitectura proporciona la estructura sobre la cual guían las iteraciones, mientras que los casos de uso definen los objetivos y dirigen el trabajo de cada iteración. La eliminación de una de las tres ideas reduciría drásticamente el valor del proceso unificado.

La vida del proceso unificado

El proceso unificado se repite a lo largo de una serie de ciclos que constituye en la vida de un sistema. Cada ciclo concluye con una versión del producto. Cada ciclo consta de cuatro fases: inicio, elaboración, construcción y transición. Cada fase se subdivide a su vez en iteraciones, como se ha dicho anteriormente.



Los ciclos concluyen con una versión.
La vida de un proceso consta de ciclos desde su nacimiento hasta su muerte.



Versiones
Un ciclo con sus fases e iteraciones

El producto

Cada ciclo produce una nueva versión del sistema y cada versión, es un producto preparado para su entrega. Consta de un cuerpo de código fuente incluido en componentes que puede compilarse y ejecutarse.

Sin embargo el producto no solo debe ajustarse a las necesidades de los usuarios, sino también a la de todos los interesados, es decir, toda la gente que trabajará con el producto.

El producto terminado incluye los requisitos, casos de uso, especificaciones no funcionales y casos de prueba. Incluye el modelo de la arquitectura y el modelo visual –artefactos modelados con el lenguaje unificado de modelado.

- 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.

Todos estos modelos están relacionados, juntos representan al sistema como un todo. Los elementos de un modelo poseen dependencias de traza, hacia atrás y hacia delante, mediante enlaces hacia otros modelos.

Fases dentro de un ciclo

Cada ciclo se desarrolla a lo largo del tiempo. Este tiempo, a su vez, se divide en cuatro fases. A través de una secuencia de modelos, los implicados visualizan lo que está sucediendo en esas fases. Dentro de cada fase, los directores o los desarrolladores pueden descomponer adicionalmente el trabajo –en iteraciones con sus incrementos resultantes. Cada frase termina con un *hito*. Existen hitos principales y secundarios. Un *hito principal* es el punto en donde tienen que tomarse importantes decisiones de negocio. Cada fase acaba en un hito principal en el cual los gestores han de tomar decisiones

cruciales de continuar o no en el proyecto, y decidir sobre la planificación, presupuesto y requisitos del mismo. Se consideran puntos de sincronización en los que coinciden una serie de objetos bien definidos, se completan artefactos, se toman decisiones de pasar o no a la fase siguiente, y en los que las esferas técnica y de gestión entran en conjunción. Un *hito secundario* es un hito intermedio entre dos hitos principales. Puede existir, por ejemplo, al acabar una iteración, o cuando finaliza una construcción en una iteración. Cada hito se determina por la disponibilidad de un conjunto de artefactos: es decir, ciertos modelos o documentos que han sido desarrollados hasta alcanzar un estado predefinido.

Al final se obtiene un conjunto de datos a partir del seguimiento del tiempo y esfuerzo consumido en cada fase. Estos datos son útiles en la estimación del tiempo y los recursos humanos para otros proyectos, en la asignación de los recursos durante el tiempo que dura el proyecto y en el control del progreso contrastado con las planificaciones.

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?

Durante 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.

Durante 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.

Un proceso integrado

El proceso unificado está basado en componentes. Utiliza el nuevo estándar de modelado visual, el lenguaje unificado de modelado (UML). Y se sostiene sobre tres ideas básicas –casos, de usos, arquitectura, y desarrollo iterativo e incremental.

El proceso unificado ha establecido un marco de trabajo que integra todas esas diferentes facetas. Este marco de trabajo integra todas esas diferentes facetas y sirve también como paragüas bajo el cual los fabricantes de herramientas y los desarrolladores pueden construir herramientas que soporten la automatización del proceso entero, de cada flujo de trabajo individualmente, de la construcción de los diferentes modelos, y de la integración del trabajo a lo largo del ciclo de vida y a través de todos los modelos.