
# Universidad Tecnológica del Centro de Veracruz

**Nombre del tema: El proceso unificado de desarrollo de software**

**Nombre del alumno: Hernández Chávez Celia**

**Matricula: 6770**

**Materia: Ingeniería de software**

**Carrera: Tecnologías de la información y la comunicación**

**Área: Sistemas Computacionales**

**Grado: 6° A**

**Modalidad: Despresurizado**

# Contenido

# El proceso unificado de desarrollo de software2

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

## 1.1 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, es un marco de trabajo genérico que puede especializarse en diferentes áreas.

El proceso unificado está basado en componentes, el software en construcción está formado en componentes interconectados a través de interfaces bien definidas.

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

Los verdaderos aspectos definitorios del proceso unificado se resumen en tres fases clave, dirigidos en casos de uso, centrado en la arquitectura, iterativo e incremental.

## 1.2 El proceso unificado está dirigido por casos de uso

Para construir un sistema con éxito debemos conocer lo que sus futuros usuarios necesitan y desean. El termino usuario 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. Los casos de uso representan los requisitos funcionales. Todos los casos de uso juntos constituyen el modelo de casos de uso, el cual describe la funcionalidad total de sistema.

Los casos de uso no son solo una herramienta para especificar los requisitos del sistema, también guían su diseño, implementación y prueba. Los desarrolladores revisan cada uno de los modelos para que sean conformes al modelo de casos de uso. Los ingenieros de prueba prueban la implementación para garantizar que se implementen adecuadamente los casos de uso.

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

El concepto de arquitectura de software incluye los aspectos estáticos y dinámicos más significativos del sistema. Lar arquitectura surge de las necesidades de la empresa, como las perciben los usuarios y se refleja en los casos de uso.

También se ve influida por muchos otros factores, como la plataforma en la que tiene que funcionar el software (hardware, sistema operativo, base de datos) y requisitos no funcionales por ejemplo: rendimiento y fiabilidad.

La arquitectura es una vista del diseño completo con las características más importantes resaltadas, dejando los detalles de lado. Debido a que lo que es significativo depende en parte de una valoración que se adquiere con la experiencia, el valor de una arquitectura depende de las personas que se hayan responsabilizado se su creación.

Debe haber interacción entre los casos de uso y la arquitectura, los casos de uso deben encajar en la arquitectura cuando se llevan a cabo, por otro lado, la arquitectura debe permitir el desarrollo de todos los casos de uso requeridos ahora y en el futuro. En realidad tanto la arquitectura como los casos de uso deben evolucionar en paralelo.

Podemos decir que el arquitecto crea un esquema en borrador de la arquitectura comenzando por la parte de la arquitectura que no es especifica de los casos de uso. A continuación el arquitecto trabaja con un subconjunto de los casos de uso especificados con aquellos que representen las funciones claves del sistema en desarrollo.

## 1.4 El proceso unificado es iterativo e incremental

El desarrollo de un producto software comercial supone un grande esfuerzo que puede durar varios meses hasta años. Es práctico dividir el trabajo en partes más pequeñas o miniproyectos. Cada miniproyecto es una iteración que resulta en un incremento. Las iteraciones hacen referencia a pasos en el flujo de trabajo y los incrementos al crecimiento del producto.

Las iteraciones deben estar controladas deben seleccionarse y ejecutarse en una forma planificada.

Los desarrolladores deben basan la selección de lo que se implementará en una iteración de dos factores. En primer lugar la iteración trata de 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. Las iteraciones sucesivas se construyen sobre los artefactos de desarrollo tal como quedaron al final de la última iteración.

Beneficios de un proceso iterativo controlado:

- La iteración controlada reduce el riesgo del coste o los costes de un solo incremento.
- 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.
- La iteración controlada acelera el ritmo de esfuerzo del desarrollo en su totalidad debido a que los desarrolladores trabajan de manera más eficiente para obtener resultados claros a corto plazo.
- La iteración controlada reconoce una realidad que a menudo se ignora

La arquitectura proporciona la estructura sobre la cual guiar 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 reducirá el valor del proceso unificado.

## 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 constituye una versión del producto para los clientes.

Cada ciclo consta de cuatro fases que son inicio, elaboración, construcción y transición, cada fase se subdivide a su vez en iteraciones.

### 1.5.1 El producto

Cada ciclo produce una nueva versión del sistema y cada versión es un producto preparado para su entrega. Consta de un código fuente incluido en componentes que pueden compilarse y ejecutarse, además de manuales y otros productos asociados.

El producto terminado no solo debe ajustarse a las necesidades del usuario sino también a las de todos los interesados, incluyendo los requisitos, los casos de uso, especificaciones no funcionales y casos de prueba. Incluye el modelo de la arquitectura y el modelo visual. Son estos elementos los que le permiten a los usuarios utilizar y modificar el sistema de generación en generación.

El sistema también debe tener un modelo del dominio o modelo del negocio que describa el contexto del negocio en el que se haya el sistema.

### 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 negocio para el producto. En esta fase se identifican y priorizan los riesgos más importantes, se planifica en detalle la fase de elaboración y se estima el proyecto de manera aproximada.

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 el propio sistema es primordial.

La arquitectura se expresa en forma de vistas de todos los modelos del sistema los cuales juntos representan el sistema entero.

Al final de la fase de elaboración, el director del 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, en esta fase la línea base de la arquitectura crece hasta convertirse en el sistema completo. La arquitectura del sistema es estable, aunque los desarrolladores pueden descubrir formas mejores de estructurar el sistema.

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 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 ha establecido un marco de trabajo que integra todas las facetas. Este marco de trabajo sirve también 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

## 2.1 Las personas son decisivas

Hay personas implicadas en el desarrollo de un producto software durante todo su ciclo de vida. Financian el producto, lo planifican, lo desarrollan, lo gestionan, lo prueban, lo utilizan y se benefician de él.

### 2.1.1 Los procesos de desarrollo afectan a las personas

E l modo en que se organiza y gestiona un proyecto software afecta a las personas implicadas en el.

- Viabilidad del proyecto: Una aproximación iterativa en el desarrollo del software permite juzgar la viabilidad del proyecto.
- Gestión del riesgo: La exploración de los riesgos significativos en las primeras fases atenúa este problema.
- Estructura de los equipos: La gente trabaja de manera más eficaz en grupos pequeños de seis a ocho miembros.
- Planificación del proyecto: Cuando la gente cree que la planificación de un proyecto no es realista la moral se hunde, la gente no quiere trabajar a sabiendas de que por mucho que lo intenten, nunca podrán obtener los resultados deseados.
- Facilidad de comprensión del proyecto: La descripción de la arquitectura proporciona una visión general para cualquiera que se encuentre implicado en el proyecto.
- Sensación del proyecto: Un ritmo de trabajo rápido combinado con conclusiones frecuentes aumenta la sensación de progreso de la gente.

### 2.1.2 Los papeles cambiarán

Para comprender y dar soporte a los procesos de negocio y para implementarlos en software, los desarrolladores deberán trabajar con muchos otros desarrolladores. Para trabajar cada vez más eficazmente en equipos cada vez más grandes, se necesita un proceso que sirva como guía. Esta guía tendrá como resultados desarrolladores que trabajan de manera más inteligente, 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 completo gracias a los procedimientos automatizados y a los componentes reutilizables. Las personas serán decisivas en el desarrollo de software.

### 2.1.2 Convirtiendo “recursos” en “trabajadores”

Cada trabajador es responsable de un conjunto completo de actividades, para el diseño de un sistema. Para trabajar eficazmente los trabajadores necesitan la información requerida para llevar a cabo sus actividades. Necesitan comprender cuáles son sus roles en relación con los de otros trabajadores.

La herramientas no solo deben ayudar a los trabajadores a llevar a cabo sus propias actividades sino que también puede aislarles de información que no sea relevante.

Al asignar los trabajadores a un proyecto, el jefe de proyecto debe identificar las aptitudes de las personas y casarlas con las aptitudes requeridas de los trabajadores.

## 2.2 Los proyectos construyen el producto

Un proyecto de desarrollo da como resultado una nueva versión de un 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. Cada iteración implementa un conjunto casos de uso relacionados o atenúa algunos riesgos.
- Un patrón organizativo: Un proyecto implica aun equipo de personas asignadas para lograr un resultado dentro de las restricciones del negocio, tiempo, coste y calidad.

## 2.3 El producto es más que código

El termino producto aquí hace referencia al sistema entero y no solo 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 maquinas u hombres. La maquinas son las herramientas, compiladores u ordenadores.

Entre los trabadores tenemos arquitectos, directores, desarrolladores, ingenieros de prueba, administradores y otros.

Los interesados son los inversores, usuarios, jefes de proyecto, personal de producción, etcétera.

### 2.3.2 Artefactos

Artefacto es un término general para cualquier tipo de información creada, producida, cambiada, o utilizada por los trabajadores en el desarrollo del sistema. Algunos ejemplos son diagramas UML, los bocetos de interfaz de usuario y los prototipos.

Hay dos tipos de artefactos: artefactos de ingeniería y artefactos de gestión.

### 2.3.3 Un sistema posee una colección de modelos

Cuando diseñamos el proceso unificado, identificamos todos los trabajadores y cada una de sus perspectivas que posiblemente podrían necesitar.

El proceso unificado proporciona un conjunto de modelos cuidadosamente seleccionados con el cual comenzar. Este conjunto de modelos hace claro el sistema para todos los trabajadores.

Fue elegido para satisfacer las necesidades de información de esos trabajadores.

### 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.

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

### 2.3.5 Cada modelo es una vista autocontenida del sistema
Un modelo es una vista autocontenida en el sentido de que un usuario de un modelo no necesita más información para interpretarlo.

La idea de ser autocontenido significa que los desarrolladores trataron de que hubiera una sola interpretación de lo que ocurrirá en el sistema cuando se dispare un evento descrito en el modelo.

Un modelo debe describir las interacciones entre el sistema y los que los rodean.

### 2.3.6 Dentro de un modelo
Este elemento del sistema es el contenedor de los demás elementos. El subsistema de más alto nivel, representa al sistema en construcción.

### 2.3.7 Relaciones entre modelos
Un sistema contiene todas las relaciones y restricciones entre elementos incluidos en diferentes modelos. Por lo tanto un sistema no es solo la colección de sus modelos sino que contiene también las relaciones entre ellos.

El hecho de que los elementos en dos modelos estén conectados no cambia lo que hacen en los modelos a los que pertenecen.

## 2.4 El proceso dirige los proyectos
El término se refiere a los procesos de negocio, claves en una empresa de desarrollo de software, es decir una organización que desarrolla y da soporte al 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 ella. Es comparable a una clase que puede utilizarse para crear objetos en el paradigma de la programación orientada a objetos. Instancia del proceso es un sinónimo del proyecto.

### 2.4.2 Las actividades relacionadas conforman flujos de trabajo
Describimos el proceso entero en partes llamadas flujos de trabajo, un flujo de trabajo es un estereotipo de colaboración en el cual los trabajadores y los artefactos son los participantes. Por lo tanto los trabajadores y artefactos que participan en un flujo de trabajo pueden participar también en otros flujos de trabajo.

### 2.4.3 Procesos especializados
El proceso unificado se diseñó para poder ser adaptado, cada organización que utilice el proceso unificado en algún momento lo especializara para ajustarlo a su situación.

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

- Factores organizativos: La estructura organizativa, la cultura de la empresa, la gestión del proyecto y sistemas software existentes.
- Factores del dominio: El dominio de la aplicación, procesos de negocio y las ofertas disponibles de la competencia.
- Factores del ciclo de vida: El tiempo de salida al mercado, el tiempo de vida esperado del software y las versiones planificadas futuras.
- Factores técnicos: Lenguaje de programación, herramientas de desarrollo, bases de datos, etc.

### 2.4.4 Méritos del proceso

- Los desarrolladores pueden comprender mejor lo que los otros están haciendo
- Los supervisores y directores pueden comprender lo que los desarrolladores están haciendo gracias a los esquemas de la arquitectura
- La formación puede estandarizarse dentro de la empresa
- El devenir del desarrollo de software es repetible

## 2.5 Las herramientas son esenciales en el proceso

El proceso y las herramientas viene en el mismo paquete, es decir las herramientas son esenciales en el proceso.

### 2.5.1 Las herramientas influyen en el proceso

Las herramientas son buenas para automatizar procesos repetitivos, mantener las cosas estructuradas, gestionar grandes cantidades de información y para guiarnos a lo largo de un camino de desarrollo concreto.

Con poco soporte por herramientas, un proceso debe sostenerse sobre una gran cantidad de trabajo manual y será por tanto menos formal.

### 2.5.2 El proceso dirige las herramientas

Las herramientas que implementan un proceso automatizado deben ser fáciles de usar.

Hacerlas muy manejables significa que los desarrolladores de herramientas deben considerar seriamente la forma en que se lleva a cabo el desarrollo de software.

Es esencial que las herramientas sean capaces de tanto de soportar la automatización de actividades repetitivas como de gestionar la información representada por la serie de modelos y artefactos.

### 2.5.3 El equilibrio entre el proceso y las herramientas

El proceso dirige el desarrollo de las herramientas, por otro lado, las herramientas dirigen el desarrollo del proceso. El proceso del desarrollo y su soporte por desarrollo de herramientas debe tener lugar de manera simultánea.

En cada versión del proceso debe haber también una versión de las herramientas. En cada versión debe darse este equilibrio. Alcanzar el ideal de este equilibrio nos llevara varias iteraciones.

### 2.5.4 El modelado visual soporta UML

UML incluye un cierto número de reglas que también requieren soporte. Estas herramientas pueden incluirse en la herramienta de modelado, bien como de cumplimiento obligado inmediato o como rutinas, bajo demanda que recorren el modelo y comprueban si hay errores comunes.

La herramienta de modelado debe incorporar más que conocimiento de UML, debe permitir que los desarrolladores trabajen creativamente con UML.

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

Hay herramientas que soportan todos los aspectos del ciclo de vida del software:

- Gestión de requisitos: Se utiliza para almacenar, examinar, revisar, hacer seguimiento y navegar por los diferentes requisitos de un proyecto de 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, detector de errores y analizadores de rendimiento.
- Aseguramiento de la calidad: Se utiliza para probar aplicaciones y componentes, para registrar y ejecutar casos de prueba.

### 2.5.6 Bibliografía

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