educación IT
CODE YOUR FUTURE

UML y UP: Análisis y Diseño Orientado a Objetos

Módulo 3

educación IT
CODE YOUR FUTURE

Introducción al Proceso Unificado de Desarrollo de Software

Introducción al Proceso Unificado

Definición

El Proceso Unificado de Desarrollo, o UP, es una **metodología orientada a la construcción de *software***.

Nace como la unión de distintas metodologías antes separadas por distintos autores y concentra lo mejor de cada una. Es posible definirlo como
un proceso que define "quién está haciendo qué", cuándo lo está haciendo, y cómo alcanzar un objetivo. **Es el proceso utilizado para guiar a los desarrolladores**.

Se lo conoce inicialmente como **RUP** (*Rational Unified Process*), que es la propuesta propietaria de la empresa *Rational*, y en su propuesta genérica se lo denomina **UP** (*Unified Process*).

La diferencia con UML es que UML es un medio, y no un fin. **UML tiene como objetivo documentar, visualizar y modelar un sistema,** no define quién realiza cada actividad, tampoco define tiempos ni coordinación entre los roles de los distintos trabajadores del sistema. **UML no es una metodología (como sí lo es UP) sino un lenguaje**.

Historia

El proceso *Objectory*

La metodología **RUP** tiene una larga historia que nace en la compañía *Objectory Systems*, que Ivar Jacobson fundó en 1987. El ingeniero sueco decidió independizarse de la empresa Ericsson, donde trabajaba haciendo análisis y diseño de sistemas de información.

Su objetivo fue crear una compañía para dedicarse a la investigación de una metodología que permitiera el desarrollo de sistemas de manera organizada.

Así, creó *Objectory Systems*, usando toda la experiencia que había adquirido realizando tareas similares en su anterior empresa.

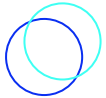
El proyecto se desarrolló en forma independiente, hasta el año 1996. En esos años logró una gran evolución y captó la atención de *Rational*, compañía que luego compró a *Objectory Systems* para comenzar a marcar el camino hacia lo que luego se conocería como **RUP**. *Objectory* representaba el concepto de *Object Factory* (o fábrica de objetos).

El proceso *Objectory* de *Rational*

En el año 1996, *Rational* compra *Objectory* y comienza a unificar los principios básicos de *Objectory* con los propios.

Entre los aspectos más importantes, se destacan::

- Utilización de fases dentro de un proyecto.
- Uso de iteraciones controladas dentro de las fases.
- Establecimiento de la arquitectura como la parte más significativa de la organización del sistema.
- Incorporación de UML como lenguaje de modelado (estaba en fase de desarrollo, versión 1.1)



Estos cambios fueron realizados entre los años 1996 y 1997.

El Proceso Unificado de Rational (RUP)

En el año 1998, *Rational* decide expandirse en la búsqueda de una metodología genérica, y comienza a comprar empresas que se encargaban de actividades puntuales y estaban bien desarrolladas en campos como la gestión de requisitos, gestión de configuración, testing, entre otros.

Rational absorbe toda la experiencia y conocimientos de esas compañías. Luego, construye la metodología denominada **RUP**.

Para establecer la sigla se basa en estas ideas:

- **R** de ***Rational***, correspondiente al nombre de la empresa.
- **U** de ***Unified***, qué significa “unificado” ya que representa la unificación de las diferentes técnicas de desarrollo y de todas las metodologías utilizadas anteriormente.
- **P** de ***Process***, correspondiente al proceso que indica paso a paso las tareas a realizar para cumplir un objetivo.

La necesidad de una metodología

En todo desarrollo de *software* es necesaria una **metodología para coordinar a los trabajadores** que forman parte del proyecto.

Resulta necesario un proceso que:

- Proporcione una guía para ordenar actividades de un equipo, dirigiendo tareas para cada desarrollador por separado y del equipo como un todo, para lograr que no se solapan tareas junto con el cumplimiento de objetivos en las fechas planificadas.
- Ofrezca criterios para el control de calidad.
- Ofrezca criterios en cuanto a la medición de productos y actividades del proyecto en términos de tiempo y su evolución correspondiente.
- Maximice la productividad.

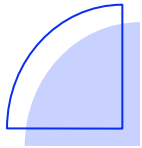
Fundamentos del Proceso Unificado de Desarrollo

El **Proceso Unificado de Desarrollo** es un proceso de desarrollo de *software* que especifica las actividades necesarias para transformar los requisitos de un usuario en un sistema de *software*. Proporciona un marco genérico de trabajo, y utiliza como herramienta visual al UML.

Está basado en tres aspectos claves:

- Está dirigido por casos de uso.
- Está centrado en una arquitectura.
- Es iterativo e incremental.

Los tres conceptos tienen la misma importancia. La arquitectura proporciona la estructura para las iteraciones que se realizarán durante el proceso de desarrollo para evolucionar y refinar el producto. Los casos de uso definirán los objetivos y la dirección del trabajo en cada iteración.



Dirigido por Casos de Uso

El Proceso Unificado de Desarrollo está dirigido por el conjunto de casos de uso. Un caso de uso es una funcionalidad bien definida del sistema que proporciona al usuario un resultado.

Representa a los requisitos funcionales, es decir, lo que debería hacer el sistema.

El conjunto de casos de uso constituye el Modelo de Casos de Uso, que describe la funcionalidad integral del sistema. Se utilizan para guiar el dise-
ño y la implementación, y representan el punto de partida para la construcción de los casos de prueba (*test cases*).

Se utiliza la palabra "dirigido" ya que los casos de uso sirven como hilo conductor del desarrollo del *software*.

Centrado en una arquitectura

El Proceso Unificado de Desarrollo está centrado en una arquitectura ya que ésta representa los cimientos del *software* a desarrollar, es donde el *software* será ejecutado. Una arquitectura comprende la definición de los siguientes conceptos:

- Arquitectura de *hardware*.
- Sistema Operativo.
- *DataBase Management System* (DBMS).
- Protocolos de comunicación de red.
- Estrategia de diseño a seguir (multicapa, orientados a servicios, etc.).

Para su definición, el arquitecto de *software* debe comprender los casos de uso claves del sistema. Esto se debe a que una arquitectura tiene una relación directa con los casos de uso, ya que estos deben "encajar" en la arquitectura, es decir que **la arquitectura debe permitir el desarrollo normal de todos los casos de uso**.

Para evaluar la viabilidad técnica de los casos de uso sobre una arquitectura, generalmente se eligen entre el 5% al 10% del total, siendo estos casos de uso los más representativos del sistema o los que constituyen las funciones claves.

Para definir una arquitectura es conveniente, por un lado, esquematizarla según los factores no dependientes de los casos de uso, y por el otro, trabajar con los casos de uso del sistema, especificarlos y realizarlos en términos de subsistemas, componentes y clases. A medida que los casos de uso se especifican y se van refinando, la arquitectura irá "madurando".

Una arquitectura debe satisfacer también a los requisitos no funcionales, como ser rendimiento, performance y velocidad de respuesta.

El objetivo es encontrar una arquitectura estable y escalable, construyéndola de tal manera que permita que el sistema de *software* evolucione.

Iterativo e incremental

El Proceso Unificado de Desarrollo es iterativo e incremental pues divide a un proyecto en "mini proyectos", también denominados *iteraciones.* El proyecto se estructura en **cuatro grandes fases** (presentadas a continuación) donde en cada fase se realizan una gran cantidad de iteraciones.

Cada iteración deberá planificarse y ejecutarse en forma controlada, basada en casos de uso bien especificados para mitigar el riesgo y lograr así, en cada iteración, un incremento real del sistema. Se intenta planificar todas las iteraciones de la fase (o la mayor cantidad posible) y su correspondiente orden lógico.

Una iteración comienza con la detección de los casos de uso (Requisitos), luego Análisis, Diseño, Implementación y Pruebas, es decir que cada iteración convierte un conjunto de casos de uso en código fuente.

Por cada iteración:

- Se identifican y especifican los casos de uso más relevantes.
- Se crea un diseño respetando la arquitectura.
- Se implementa el diseño con componentes.
- Se verifica que el resultado satisface los casos de uso.

Teniendo en cuenta que las necesidades del usuario (los requisitos) no pueden definirse completamente desde el comienzo, las distintas iteraciones sirven como un refinamiento de requisitos.

Ciclo de vida del Proceso Unificado

El Proceso Unificado de Desarrollo establece **la construcción de un sistema en cuatro fases** bien definidas:

- Inicio.
- Elaboración.
- Construcción.
- Transición.

Cada fase termina con un **hito**, donde se decide si se avanza a la fase siguiente. En este punto se prevén presupuestos, planificaciones y requisitos. Se suele reunir el personal técnico con el de gestión para hacer una evaluación de la situación.

Los hitos se utilizan para **estimar tiempos y recursos necesarios** para la próxima fase.

Sirven también para **controlar el progreso** con las planificaciones previamente realizadas. Si se encuentra alguna deficiencia, debe ser corregida antes de pasar a la siguiente fase.

Cada fase está formada por iteraciones, y cada iteración está compuesta por flujos fundamentales de trabajo. **Una iteración representa el conjunto de actividades** que tienen como objetivo realizar un avance real y “tangible” del sistema.

Para completar una iteración es necesario llevar a cabo todos los flujos fundamentales de trabajo, presentados a continuación:

- Requerimientos.
- Análisis y Diseño.
- Implementación.
- Testing.
- Configuración.

Fase de Inicio

En esta fase se genera una **descripción del producto** y se presenta el **análisis del negocio**.

Se definen las funciones principales del sistema para los usuarios más representativos, el plan de proyecto y el costo del producto. Esto permite identificar los riesgos más importantes. Se debe realizar una estimación aproximada del proyecto en cuanto a tiempos, costos y recursos.

Se comienza a pensar en una posible arquitectura y se planifica en detalle la *Fase de Elaboración.*

Fase de Elaboración

En esta fase se genera una **especificación detallada de la mayoría de los casos de uso** y se **construye la arquitectura alineada con los más críticos**.

Con la especificación de los casos de uso realizada se planifican las actividades y se estiman los recursos necesarios hasta la terminación del proyecto.

Fase de Construcción

En la fase de Construcción **se utilizan la mayoría de los recursos**. A esta altura la arquitectura ya es estable, aunque se pueden incorporar pequeñas mejoras.

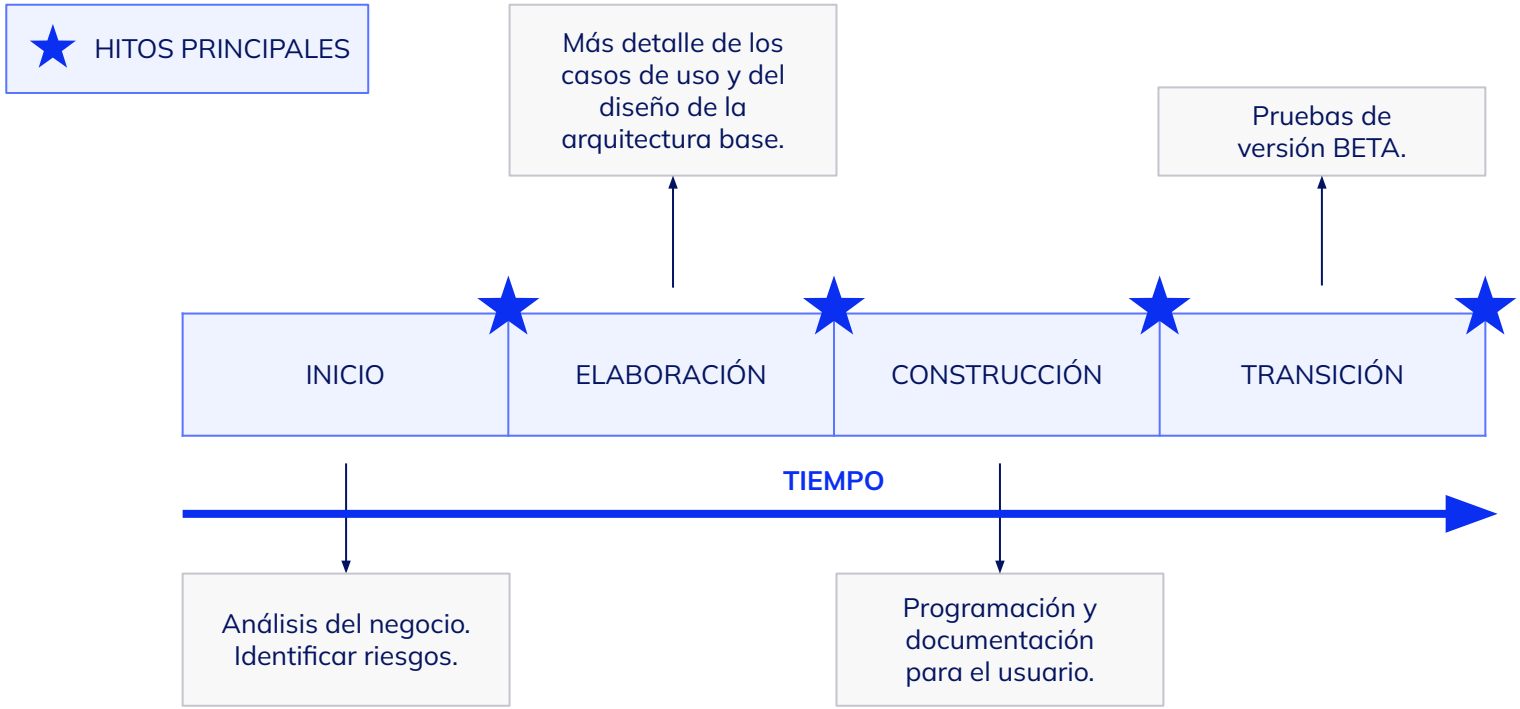
En esta fase se construye el producto hasta convertirse en un sistema funcional.

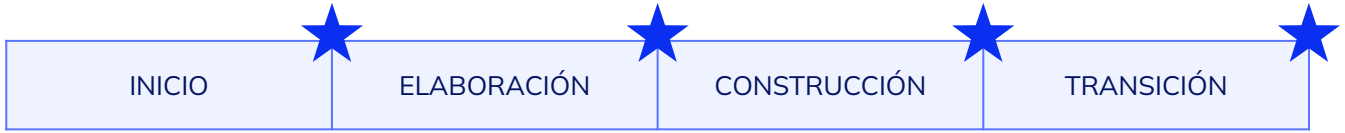
Al finalizar esta fase, el sistema responderá por todos los casos de uso que han pactado entre la empresa y el cliente. El sistema final podría contener defectos (*bugs*) que se solucionarán en la próxima fase: la fase de Transición.

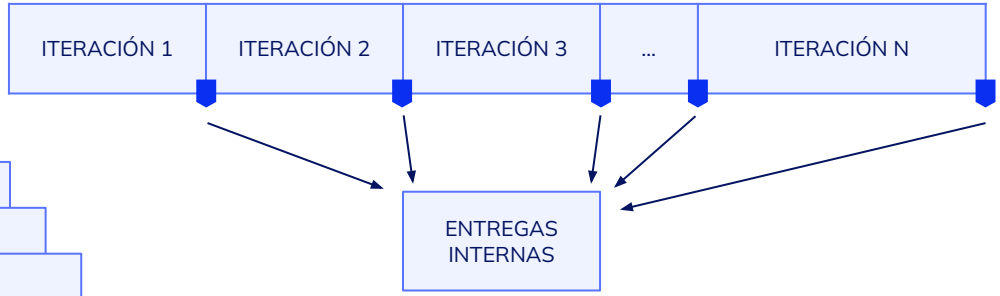
Fase de Transición

En la fase de Transición, **el producto se convierte en versión beta**. Los usuarios experimentados utilizan esta versión para probar el producto en busca de defectos. Los desarrolladores corrigen los errores a medida de que se van detectando.

Esta fase incluye también la capacitación a usuarios y se redactan formalmente los manuales.


HITOS PRINCIPALES
Más detalle de los casos de uso y del diseño de la arquitectura base.
Pruebas de versión BETA.
INICIO
ELABORACIÓN
CONSTRUCCIÓN
TRANSICIÓN
TIEMPO
Análisis del negocio. Identificar riesgos.
Programación y documentación para el usuario.


INICIO
ELABORACIÓN
CONSTRUCCIÓN
TRANSICIÓN


ITERACIÓN 1
ITERACIÓN 2
ITERACIÓN 3
...
ITERACIÓN N
ENTREGAS INTERNAS
REQUERIMIENTOS
ANÁLISIS Y DISEÑO
CODIFICACIÓN
PRUEBAS
ADM DEL PROYECTO
GESTIÓN CONFIGURACIÓN Y CAMBIOS
DISCIPLINA

educación IT
CODE YOUR FUTURE

¡Sigamos trabajando!