CHAPTER 2

Procesos de Desarrollo de Software

2.1 La idea de Proceso

Imagina que deseas preparar una deliciosa torta. No basta con tener todos los ingredientes requeridos, sino que también necesitas seguir un procedimiento detallado para obtener el resultado deseado. Desde mezclar la masa hasta hornearla adecuadamente y agregar la decoración final, cada paso tiene su propia secuencia y herramientas específicas. El proceso debe ser seguido con precisión para garantizar que la torta resultante sea consistente en sabor y apariencia.

Este mismo principio se aplica en la industria, donde la construcción de casi cualquier objeto, ya sea un automóvil completo o incluso un simple neumático (puedes observar este proceso en este video), sigue un proceso meticuloso y organizado.

Por lo tanto, es lógico deducir que el desarrollo de software, siendo un producto potencialmente más complejo que un automóvil, también debe basarse en un proceso bien definido. De hecho, a lo largo de los años, se han identificado y delineado las principales etapas involucradas en este proceso:

- **Estudio de Factibilidad:** Determinar si el proyecto es viable en términos económicos, riesgos y necesidades.
- **Ingeniería de Requisitos:** Especificación exhaustiva de los requisitos del software.
- **Diseño de la Interfaz de Usuario:** Creación de la experiencia visual y funcional para los usuarios.

Figure 2-1 Ilustración: Proceso de Construcción de un Automovil

- **Diseño de la Arquitectura:** Definir la estructura y componentes del software.
- **Diseño Detallado (Módulos):** Desglose detallado de componentes individuales.
- **Programación (Codificación):** Escribir el código según las especificaciones.
- **Integración (Construcción):** Ensamblar y combinar los módulos en un sistema completo.
- **Verificación (Pruebas Unitarias y de Sistema):** Evaluar cada componente y el sistema en su conjunto.
- **Validación (Pruebas de Aceptación):** Confirmar que el software cumple con los requisitos.
- **Puesta en Producción:** Implementar el software en el entorno de uso real.
- **Mantenimiento:** Realizar actualizaciones, correcciones y mejoras.
- **Documentación:** Crear manuales y documentación técnica.
- **Gestión del Proyecto:** Supervisar y coordinar todas las etapas del proceso.

A pesar de esta estructura, durante mucho tiempo se ha desarrollado software sin seguir un proceso definido. A menudo denominado “hacking”, este enfoque solía requerir individuos altamente talentosos para lograr resultados exitosos. Sin embargo, su mayor desventaja radica en su falta de repetibilidad; repetir el mismo proceso podría no generar los mismos resultados exitosos.

En las siguientes secciones, nos sumergiremos en ejemplos específicos de procesos de desarrollo de software. Analizaremos detenidamente las ventajas y desventajas de cada enfoque, lo que te permitirá comprender mejor

Figure 2-2 Ilustración: Modelo en Cascada

cómo elegir el proceso más adecuado según las necesidades del proyecto y los objetivos comerciales.

2.2 El Primer Modelo de Proceso: Cascada

El Modelo en Cascada, también conocido como el enfoque secuencial lineal, se compone de una serie de etapas interdependientes (cada etapa depende de la anterior) que guían el desarrollo de proyectos. Estas etapas, que incluyen la definición de requisitos, diseño, implementación, pruebas y mantenimiento, se despliegan de manera secuencial, donde cada etapa sienta las bases para las siguientes. A lo largo de más de tres décadas, este modelo ha sido ampliamente utilizado y, sorprendentemente, continúa aplicándose en algunas ocaciones particulares a pesar de sus notables limitaciones.

¿Por qué goza de tanta popularidad?

- Su simplicidad conceptual y facilidad de comprensión.
- Analogía con procesos de manufactura o construcción.
- Claridad en cada fase con entradas y salidas definidas.
- Facilita la asignación de recursos.
- Permite el seguimiento del progreso y la estimación de lo que falta.

¿Por qué abandonar este enfoque?

- Su eficacia es cuestionable en la práctica, excepto para proyectos muy breves.
- Los requisitos iniciales no siempre son completamente conocidos.
- Incluso si se conocen, pueden cambiar a lo largo del proyecto.

- Existe un alto riesgo hasta las etapas avanzadas del proyecto.

2.3 Procesos Iterativos

Las actividades de desarrollo avanzan en forma de una secuencia de iteraciones, a través de las cuales nos aproximamos gradualmente a nuestro objetivo. Cada iteración no siempre resulta en la generación de nuevo código o en un incremento visible y útil. En este capitulo veremos tres ejemplos de modelos de procesos iterativos comprenden:

- El Modelo Espiral.
- El Modelo de Prototipos.
- El Modelo Unificado (inicialmente denominado RUP).

Modelo en Espiral

El modelo en espiral puede considerarse una evolución del modelo en cascada. Se caracteriza por su capacidad de operar en ciclos y fases iterativas. Una de sus contribuciones más destacadas es haber sido el primero en introducir consideraciones de riesgo de manera explícita. De acuerdo con este enfoque, cada iteración se compone de cuatro fases fundamentales: análisis de requisitos, construcción, evaluación y análisis de riesgo, y planificación de la siguiente iteración.

La espiral se divide en cuatro cuadrantes:

- **Determinación de objetivos**: En este cuadrante, se recopilan los requisitos provenientes de los clientes.
- **Identificación y resolución de riesgos**: Aquí se evalúan todas las posibles soluciones para seleccionar la más adecuada.
- **Desarrollo y pruebas**: Durante el tercer cuadrante, se desarrollan las funcionalidades identificadas y se verifican a través de pruebas exhaustivas. Al final de esta fase, se busca liberar una nueva versión del software.
- **Planificación para la siguiente iteración**: En este cuadrante, los clientes evalúan el progreso realizado en la nueva versión del software.

Algunas ventajas del modelo en espiral son:

- Manejo de riesgo.
- Flexibilidad en los requerimientos. Los cambios de requerimientos pueden ser presentados e incorporados.
- Involucra retroalimentación del clinete.
- Desarrollo en partes donde el riesgo se puede manejar de forma separada.

Algunas desventajas:

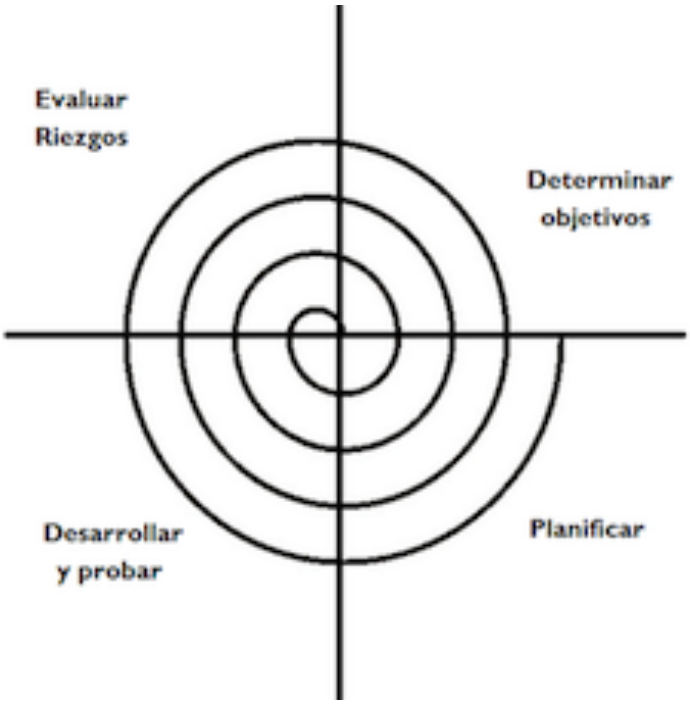

Figure 2-3 Ilustración: Modelo en Espiral

- Depende mucho del análisis de riesgo.
- Dificultades en el manejo del tiempo. El numero de fases es desconocido, por lo que es dificil estimar el tiempo de finalización del proyecto.
- Requiere una inversión importante en la planeación, análisis de riesgo y evaluaciones.

Modelo basado en Prototipos

En un proceso basado en prototipos, la idea central es la construcción física de un producto que "luce" similar al objetivo pero que en realidad está bastante distante de serlo. La idea es que el usuario pueda tener una mirada concreta del producto en forma temprana y resolver el dilema de "el usuario solo sabrá lo que quería cuando le muestres el resultado".

Hay dos formas de trabajar con prototipos: prototipos desechables y prototipos evolutivos. En el primer caso, el prototipo solo es usado para asegurar que el usuario valide lo que se está construyendo y el prototipo se desecha comenzando el desarrollo desde cero. En el segundo caso, el mismo prototipo evoluciona hasta convertirse en el producto final (discutir ventajas y desventajas de uno y otro enfoque)

Algunas ventajas de desarrollar con prototipos son:

- permite enfrentar falta de claridad en requerimientos
- menor tiempo de desarrollo porque no hay grandes sorpresas al final

- involucramiento del usuario es más fácil de lograr
- menor frustración y ansiedad durante el desarrollo
- permite introducir nuevo sistema gradualmente
- facilita entrenamiento de usuarios
- aumento de satisfacción del usuario

Pero hay también desventajas:

- puede generar expectativas de que el *software* está "casi listo"
- producto final no queda igual que prototipo que se desecha
- usuario puede no estar nunca satisfecho (proyecto no tiene fin)
- análisis incompleto de requisitos (superficialmente entendido)
- omisión de requerimientos no funcionales
- difícil de planificar (cuantos prototipos, cuanto tiempo)
- no hay entregables claros
- poco apropiado para sistemas embebidos, sistemas de tiempo real, *software* científico, etc.

Proceso de Desarrollo Unificado

El Proceso de Desarrollo Unificado, anteriormente conocido como RUP (*Rational Unified Process*), se fundamenta en un enfoque que comprende cuatro fases distintivas: inicio, elaboración, construcción y transición. En cada una de estas fases, se ejecutan diversas actividades que abarcan desde la definición de requisitos hasta el diseño, la programación y otros procesos afines. Cada fase, no obstante, se caracteriza por la predominancia de ciertas actividades específicas. Por ejemplo, en la etapa de inicio, el desarrollo o la programación son limitados.

Estas fases se estructuran en base a objetivos fundamentales:

- **Inception** (Inicio) - Establecimiento del caso y la viabilidad del sistema propuesto.
- **Elaboration** (Elaboración) - Verificación de la capacidad para construir el sistema dentro de las restricciones existentes.
- **Construction** (Construcción) - Edificación de un sistema que funcione de manera exitosa (versión beta).
- **Transition** (Transición) - Entrega de un sistema completamente funcional a los usuarios.

Cada fase puede atravesar múltiples iteraciones hasta alcanzar los objetivos mencionados anteriormente. La figura siguiente proporciona una representación visual de esta idea:

El Modelo Unificado proporciona una definición exhaustiva de múltiples *workflows*, en los cuales se establecen roles sumamente específicos, artefactos de entrada, artefactos de salida y otros componentes clave. Este enfoque

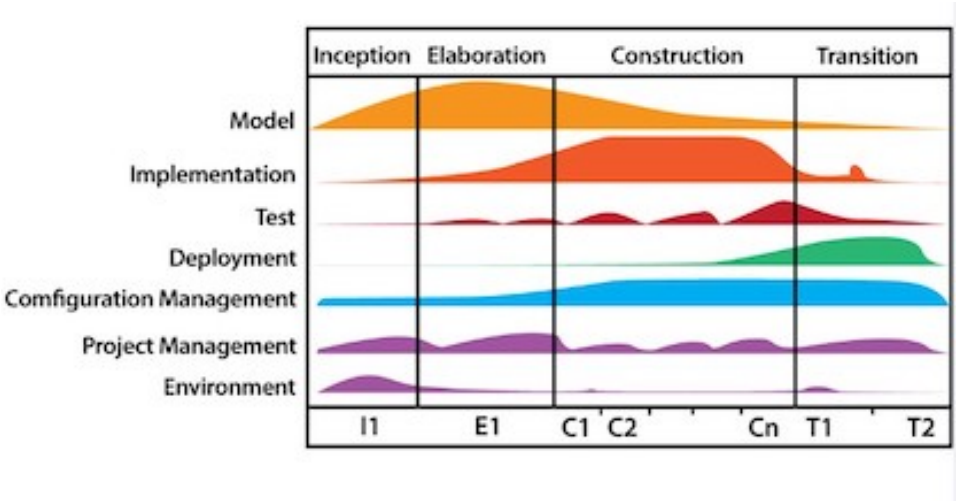

Figure 2-4 Ilustración: Proceso Unificado

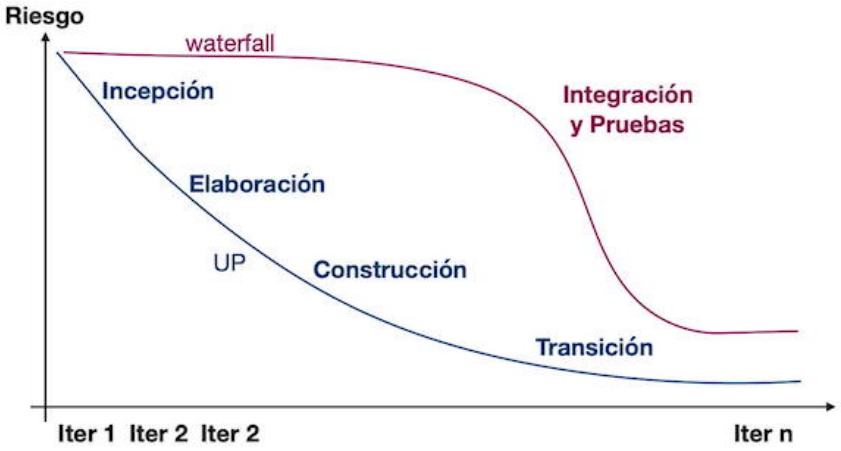

Figure 2-5 Riesgo a lo largo del desarrollo iterativo. RUP versus Cascada

marca uno de los primeros intentos por crear un proceso de desarrollo iterativo de manera sumamente completa.

Uno de los aspectos en los que este proceso demuestra una gran eficacia es en la mitigación temprana del riesgo. Al priorizar el desarrollo de los casos de uso más arriesgados durante las primeras iteraciones, es posible reducir la exposición al riesgo sin necesidad de esperar hasta que el proyecto alcance etapas avanzadas (en contraposición al enfoque en Cascada).

Este modelo enfrenta ciertos desafíos. Por un lado, su complejidad puede resultar abrumadora para equipos de desarrollo reducidos, ya que una sola persona podría tener que asumir múltiples roles. Por otro lado, a pesar de ser un proceso iterativo, su enfoque no siempre es incremental, lo que limita su adaptabilidad a las metodologías ágiles, que son las más ampliamente reconocidas y utilizadas en la actualidad.

2.4 Procesos Iterativos Incrementales

En un proceso iterativo incremental, cada iteración genera un aumento de valor tangible para el cliente, presentando funcionalidades que están listas para ser utilizadas. El producto se considera completo con la entrega del último de estos incrementos. Estos incrementos, en general, se relacionan con la implementación de uno o más nuevos relatos de usuario (detallados más adelante).

Procesos ágiles

Los modelos de proceso caracterizados por su enfoque iterativo incremental suelen recibir la denominación de "ágiles". El origen de estos procesos ágiles se encuentra en el Manifiesto Ágil, un documento concebido por un grupo destacado de 17 desarrolladores que respondieron a los problemas evidentes en los procesos tradicionales, como el modelo en cascada. Este manifiesto se basa en cuatro valores fundamentales:

- Priorizamos individuos e interacciones sobre procesos y herramientas.
- Valoramos el software funcionando más que la documentación exhaustiva.
- Fomentamos la colaboración con el cliente en lugar de la negociación contractual.
- Nos adaptamos a los cambios en vez de aferrarnos a un plan rígido.

Además, se presenta una serie de 12 principios que profundizan en diversos aspectos:

- **Satisfacción del cliente**: Priorizamos la satisfacción del cliente a través de la entrega temprana y continua de software valioso.
- **Cambios en los requisitos**: Apreciamos los cambios en los requisitos, incluso en etapas avanzadas, y utilizamos estos cambios para entregar un producto competitivo.
- **Entrega frecuente**: Entregamos frecuentemente software funcionando, con preferencia por intervalos más cortos.
- **Colaboración con el cliente**: Colaboramos estrechamente con los clientes y usuarios a lo largo del proceso de desarrollo.
- **Motivación de individuos**: Generamos un entorno en el que los individuos motivados pueden dar lo mejor de sí.
- **Comunicación cara a cara**: La comunicación directa y en persona es esencial para un entendimiento efectivo.
- **Software funcionando**: El software funcionando es la medida principal de progreso.
- **Desarrollo sostenible**: Se enfoca en un ritmo sostenible de trabajo, promoviendo la agilidad a largo plazo.
- **Excelencia técnica**: La atención constante a la excelencia técnica y al buen diseño mejora la agilidad.

- **Simplicidad**: Se valora la simplicidad y se busca maximizar la cantidad de trabajo no realizado.
- **Equipos autoorganizados**: Los equipos tienen la autoridad para tomar decisiones y adaptarse a las circunstancias cambiantes.
- **Reflexión y ajuste continuo**: A intervalos regulares, el equipo reflexiona sobre su desempeño y ajusta su comportamiento en consecuencia.

Extreme Programming

Uno de los primeros modelos de desarrollo ágil, *Extreme Programming* (XP), no es tanto un modelo de proceso en sí, sino más bien una colección de buenas prácticas alineadas con los valores y principios del Manifiesto Ágil. Entre sus pilares se encuentran:

- Mantener al cliente como parte integral del equipo de desarrollo.
- Formular casos de uso concisos (tres frases).
- Que el cliente especifique los casos de prueba.
- Buscar la simplicidad máxima en el software.
- Fomentar la programación en pares.
- Fomentar la propiedad compartida del código.
- Mantener pruebas continuas.

La programación en pares implica que dos desarrolladores trabajen simultáneamente en el mismo código, compartiendo un solo teclado y ratón. Uno actúa como conductor y el otro como revisor. Esta práctica ha demostrado ser muy eficiente. Ver video.

Entre sus ventajas se incluyen:

- La revisión de código por pares detecta errores tempranamente.
- Al menos dos personas comprenden profundamente el código.
- El código tiende a ser más legible y de mejor calidad.
- Facilita la aplicación de estándares de codificación.

Modelos Ágiles Destacados

Dos de los modelos ágiles más populares son Scrum y Kanban. Scrum ha alcanzado tal popularidad que, en muchos casos, se ha convertido en sinónimo de metodología ágil.

En resumen, Scrum opera bajo un modelo iterativo incremental, donde en cada iteración de duración fija (generalmente 2 o 3 semanas), el equipo de desarrollo elige un nuevo incremento basado en prioridades establecidas por el representante del cliente o usuario (*product owner*). El equipo tiene amplia libertad para gestionar la iteración y, al final, se llevan a cabo dos reuniones fundamentales: revisión del producto y retrospectiva del proceso.

Por otro lado, Kanban no emplea iteraciones de longitud fija. Su enfoque radica en optimizar el flujo de tareas desde la fase de “por hacer” hasta “hecho”.

Esto se logra mediante una visualización efectiva del progreso y la limitación del trabajo en curso.

Estos dos modelos de proceso serán analizados con mayor profundidad en las siguientes secciones.

La Esencia de la Agilidad y la Agilidad Moderna

Alistair Cockburn, uno de los autores del Manifiesto Ágil, ha expresado su decepción ante la evolución que ha experimentado la sencilla idea de agilidad. Según Cockburn, se ha complicado en exceso una idea que, en su origen, era muy simple: colaborar, entregar, reflexionar, mejorar. Su propuesta se resume en el concepto del corazón de la agilidad.

Siguiendo una línea similar, Joshua Kerievsky ha simplificado la agilidad en su propuesta de agilidad moderna, que se basa en solo cuatro prácticas:

- Potenciar al equipo para alcanzar su máximo potencial.
- Entregar valor de manera continua.
- Establecer la seguridad como requisito previo.
- Experimentar y aprender de manera ágil.

2.5 Ejercicios

Preguntas Cerradas:

- El Modelo en Cascada es un proceso iterativo e incremental. (Falso/Verdadero)
- El desarrollo de software puede llevarse a cabo siguiendo un proceso caótico sin ninguna estructura definida. (Falso/Verdadero)
- El Modelo en Espiral fue el primero en introducir consideraciones de riesgo de manera explícita. (Falso/Verdadero)
- En el Modelo en Espiral, cada iteración consta de cuántas fases fundamentales?
- El Proceso de Desarrollo Unificado (RUP) se divide en cuántas fases distintivas?
- El desarrollo de prototipos evolutivos implica desechar el prototipo original y comenzar desde cero. (Falso/Verdadero)
- Extreme Programming (XP) es un modelo de proceso altamente estructurado y rígido. (Falso/Verdadero)
- En Kanban, las iteraciones tienen una duración fija y predefinida. (Falso/Verdadero)
- En Scrum, el product owner establece las prioridades para los nuevos incrementos. (Falso/Verdadero)
- El Manifiesto Ágil se basa en valores y principios que promueven la documentación exhaustiva como prioridad. (Falso/Verdadero)

Preguntas Abiertas:

- ¿Cuál es la importancia de contar con un proceso bien definido en el desarrollo de software, similar al proceso de preparar una torta?
- ¿Qué ventajas y desventajas ves en la programación en pares como se presenta en Extreme Programming?
- ¿Cuál crees que es la principal razón por la que el Modelo en Cascada sigue siendo utilizado en ciertos contextos a pesar de sus limitaciones?
- ¿Por qué crees que el enfoque de desarrollo basado en prototipos puede ser especialmente adecuado para proyectos con requisitos poco claros?
- ¿Cómo consideras que los valores y principios del Manifiesto Ágil se alinean con la filosofía de desarrollo de software?