El proceso de desarrollo del software

Un proceso de desarrollo de software tiene como propósito la producción eficaz y eficiente de un producto software que reúna los requisitos del cliente. Dicho proceso, en términos globales se muestra en la Figura 2 [3]. Este proceso es intensamente intelectual, afectado por la creatividad y juicio de las personas involucradas [4]. Aunque un proyecto de desarrollo de software es equiparable en muchos aspectos a cualquier otro proyecto de ingeniería, en el desarrollo de software hay una serie de desafíos adicionales, relativos esencialmente a la naturaleza del producto obtenido. A continuación se explican algunas particularidades asociadas al desarrollo de software y que influyen en su proceso de construcción.

Un producto software en sí es complejo, es prácticamente inviable conseguir un 100% de confiabilidad de un programa por pequeño que sea. Existe una inmensa combinación de factores que impiden una verificación exhaustiva de las todas posibles situaciones de ejecución que se puedan presentar (entradas, valores de variables, datos almacenados, software del sistema, otras aplicaciones que intervienen, el hardware sobre el cual se ejecuta, etc.).

Un producto software es intangible y por lo general muy abstracto, esto dificulta la definición del producto y sus requisitos, sobre todo cuando no se tiene precedentes en productos software similares. Esto hace que los requisitos sean difíciles de consolidar tempranamente. Así, los cambios en los requisitos son inevitables, no sólo después de entregado en producto sino también durante el proceso de desarrollo.

Además, de las dos anteriores, siempre puede señalarse la inmadurez de la ingeniería del software como disciplina, justificada por su corta vida comparada con otras disciplinas de la ingeniería. Sin embargo, esto no es más que un inútil consuelo.

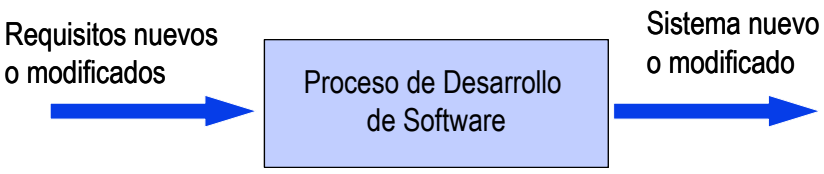


Figura 1: proceso de desarrollo de software.

El proceso de desarrollo de software no es único. No existe un proceso de software universal que sea efectivo para todos los contextos de proyectos de desarrollo. Debido a esta diversidad, es difícil automatizar todo un proceso de desarrollo de software.

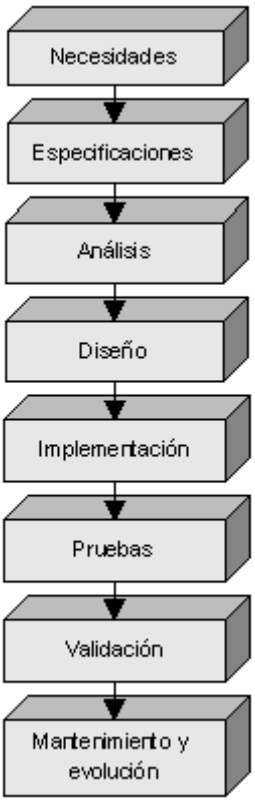
A pesar de la variedad de propuestas de proceso de software, existe un conjunto de actividades fundamentales que se encuentran presentes en todos ellos [4]:

1. **Especificación de software**: Se debe definir la funcionalidad y restricciones operacionales que debe cumplir el software.
2. **Diseño e Implementación**: Se diseña y construye el software de acuerdo a la especificación.
3. **Validación**: El software debe validarse, para asegurar que cumpla con lo que

quiere el cliente.

4. **Evolución**: El software debe evolucionar, para adaptarse a las necesidades del cliente.

Ciclo de Vida del desarrollo de software



El desarrollo de software va unido a un **ciclo de vida** compuesto por una serie de etapas que comprenden todas las actividades, desde el momento en que surge la idea de crear un nuevo producto software, hasta aquel en que el producto deja definitivamente de ser utilizado por el último de sus usuarios.

Etapas en el ciclo.

Veamos, a grandes rasgos, una pequeña descripción de etapas con que podemos contar a lo largo del ciclo de vida del software; una vez delimitadas en cierta manera las etapas, habrá que ver la forma en que estas se afrontan (existen diversos **modelos de ciclo de vida**, y la elección de un cierto modelo para un determinado tipo de proyecto puede ser de vital importancia; el orden de las etapas es un factor importante, p.ej. tener una etapa de validación al final del proyecto, tal como sugiere el modelo en cascada o lineal, puede implicar serios problemas sobre la gestión de determinados proyectos; hay que tener en cuenta que retomar etapas previas es costoso, y cuanto más tarde se haga más costoso resultará, por tanto el hecho de contar con una etapa de validación tardía tiene su riesgo y, por su situación en el ciclo, un posible tiempo de reacción mínimo en caso de tener que retornar a fases previas):

Expresión de necesidades

Esta etapa tiene como objetivo la consecución de un primer documento en que queden reflejados los requerimientos y funcionalidades que ofrecerá al usuario del sistema a desarrollar (qué, y no cómo, se va a desarrollar).

Dado que normalmente se trata de necesidades del cliente para el que se creará la aplicación, el documento resultante suele tener como origen una serie de entrevistas cliente-proveedor situadas en el contexto de una relación comercial, siendo que debe ser comprendido por ambas partes (puede incluso tomarse como base para el propio acuerdo comercial).

Especificaciones

Ahora se trata de formalizar los requerimientos; el documento obtenido en la etapa anterior se tomará como punto de partida para esta fase. Su contenido es aún insuficiente y lleno de imprecisiones que será necesario completar y depurar.

Por medio de esta etapa se obtendrá un nuevo documento que definirá con más precisión el sistema requerido por el cliente (el empleo de los casos de uso, *use cases*, de Jacobson es una muy buena elección para llevar a cabo la especificación del sistema).

Lo más normal será que no resulte posible obtener una buena especificación del sistema a la primera; serán necesarias sucesivas versiones del documento en que irán quedando reflejada la evolución de las necesidades del cliente (por una parte no siempre sabe en los primeros contactos todo lo que quiere realmente, y por otra parte pueden surgir cambios externos que supongan requerimientos nuevos o modificaciones de los ya contemplados).

Análisis

Es necesario determinar que elementos intervienen en el sistema a desarrollar, así como su estructura, relaciones, evolución en el tiempo, detalle de sus funcionalidades, ... que van a dar una descripción clara de qué sistema vamos a construir, qué funcionalidades va a aportar y qué comportamiento va a tener**.**

Diseño

Tras la etapa anterior ya se tiene claro que debe hacer el sistema, ahora tenemos que determinar **como** va a hacerlo (¿cómo debe ser construido el sistema?; aquí se definirán en detalle entidades y relaciones de las bases de datos, se pasará de casos de uso esenciales a su definición como casos expandidos reales, se seleccionará el lenguaje más adecuado, el Sistema Gestor de Bases de Datos a utilizar en su caso, librerías, configuraciones hardware, redes, etc.).

Implementación

Llegado este punto se empieza a codificar algoritmos y estructuras de datos, definidos en las etapas anteriores, en el correspondiente lenguaje de programación y/o para un determinado sistema gestor de bases de datos.

Pruebas

El objetivo de estas pruebas es garantizar que el sistema ha sido desarrollado correctamente, sin errores de diseño y/o programación. Es conveniente que sean planteadas al menos tanto a nivel de cada módulo (aislado del resto), como de integración del sistema (según sea la naturaleza del proyecto en cuestión se podrán tener en cuenta pruebas adicionales, p.ej. de rendimiento).

Validación

Esta etapa tiene como objetivo la verificación de que el sistema desarrollado cumple con los requisitos expresados inicialmente por el cliente y que han dado lugar al presente proyecto (para esta fase también es interesante contar con los *use cases*, generados a través de las correspondientes fases previas, que servirán de guía para la verificación de que el sistema cumple con lo descrito por estos).

Mantenimiento y evolución

Finalmente la aplicación resultante se encuentra ya en fase de producción (en funcionamiento para el cliente, cumpliendo ya los objetivos para los que ha sido creada). A partir de este momento se entra en la etapa de mantenimiento, que supondrá ya pequeñas operaciones tanto de corrección como de mejora de la aplicación (p.ej. mejora del rendimiento), así como otras de mayor importancia, fruto de la propia evolución (p.ej. nuevas opciones para el usuario debidas a nuevas operaciones contempladas para el producto).

La mayoría de las veces en que se desarrolla una nueva aplicación, se piensa sólamente en un ciclo de vida para su creación, olvidando la posibilidad de que esta deba sufrir modificaciones futuras (que tendrán que producirse con casi completa seguridad para la mayor parte de los casos).

Modelos de proceso software

Sommerville define modelo de proceso de software como “Una representación simplificada de un proceso de software, representada desde una perspectiva específica. Por su naturaleza los modelos son simplificados, por lo tanto un modelo de procesos del software es una abstracción de un proceso real.”

Los modelos genéricos no son descripciones definitivas de procesos de software; sin embargo, son abstracciones útiles que pueden ser utilizadas para explicar diferentes enfoques del desarrollo de software, de los cuales podemos nombrar:

- Codificar y corregir
- Modelo en cascada
- Desarrollo evolutivo
- Desarrollo formal de sistemas
- Desarrollo basado en reutilización
- Desarrollo incremental
- Desarrollo en espiral

Modelo Cascada

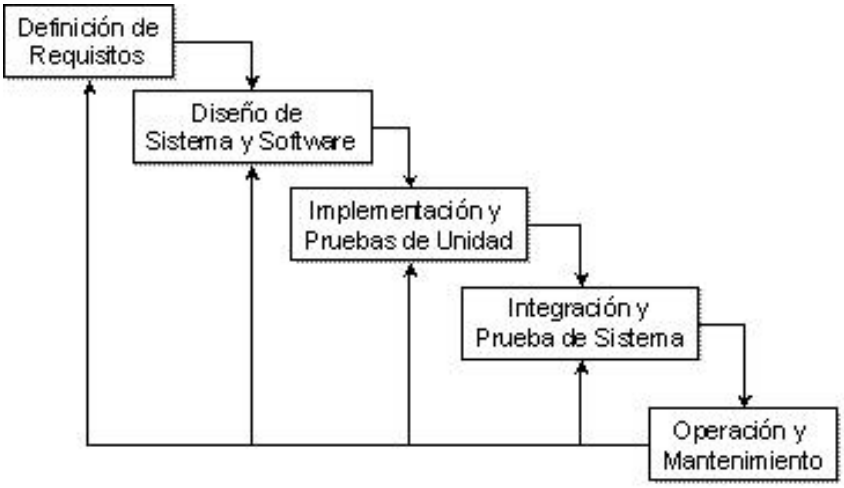
El primer modelo de desarrollo de software que se publicó se derivó de otros procesos de ingeniería . Éste toma las actividades fundamentales del proceso de especificación, desarrollo, validación y evolución y las representa como fases separadas del proceso.

El modelo en cascada consta de las siguientes fases:

1. Definición de los requisitos: Los servicios, restricciones y objetivos son establecidos con los usuarios del sistema. Se busca hacer esta definición en detalle.
2. Diseño de software: Se particiona el sistema en sistemas de software o hardware. Se establece la arquitectura total del sistema. Se identifican y describen las abstracciones y relaciones de los componentes del sistema.
3. Implementación y pruebas unitarias: Construcción de los módulos y unidades de software. Se realizan pruebas de cada unidad.
4. Integración y pruebas del sistema: Se integran todas las unidades. Se prueban en conjunto. Se entrega el conjunto probado al cliente.
5. Operación y mantenimiento: Generalmente es la fase más larga. El sistema es puesto en marcha y se realiza la corrección de errores descubiertos. Se realizan mejoras de implementación. Se identifican nuevos requisitos.

La interacción entre fases puede observarse en la figura a continuación. Cada fase tiene como resultado documentos que deben ser aprobados por el usuario.

Una fase no comienza hasta que termine la fase anterior y generalmente se incluye la corrección de los problemas encontrados en fases previas.



En la práctica, este modelo no es lineal, e involucra varias iteraciones e interacción entre las distintas fases de desarrollo. Algunos problemas que se observan en el modelo de cascada son:

- Las iteraciones son costosas e implican rehacer trabajo debido a la producción y aprobación de documentos.
- Aunque son pocas iteraciones, es normal congelar parte del desarrollo y continuar con las siguientes fases.
- Los problemas se dejan para su posterior resolución, lo que lleva a que estos sean ignorados o corregidos de una forma poco elegante.
- Existe una alta probabilidad de que el software no cumpla con los requisitos del usuario por el largo tiempo de entrega del producto.
- Es inflexible a la hora de evolucionar para incorporar nuevos requisitos. Es difícil responder a cambios en los requisitos.

Este modelo sólo debe usarse si se entienden a plenitud los requisitos. Aún se utiliza como parte de proyectos grandes.