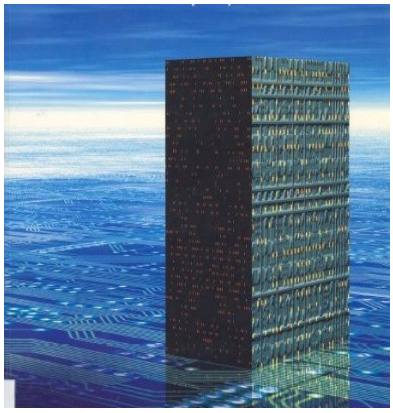
Sumario

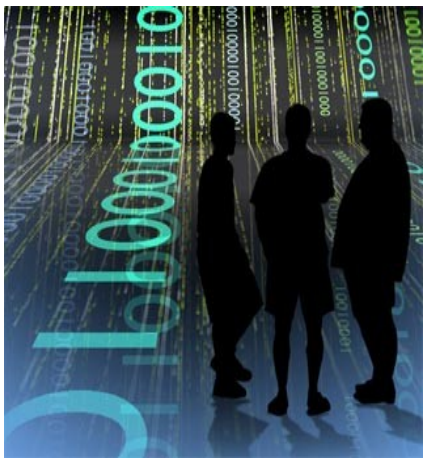
Ingeniería de software

Que es la ingeniería de software?

Se necesita alguna “técnica” para hacer software que nos guie por el camino correcto respecto de las dificultades que se pueden encontrar en el desarrollo de un proyecto. A las mencionadas técnicas se las llamó disciplinas. Cuando se tuvo un conjunto de las mismas se las denominó sub disciplinas y se las definió como los componentes del proceso sistemático del desarrollo de software para que sea tanto sistemático (se posee un sistema para llevarlo a cabo) como cuantificable (se lo puede medir para, fundamentalmente, la toma de decisiones. Un ejemplo de esto es el tiempo de desarrollo).



Llamaremos ingeniería de software a todo lo que está incluido en el proceso de creación del mismo. Existen técnicas clásicas de desarrollo y metodologías ágiles de desarrollo.
El término "ingeniero de software", se utiliza en forma genérica en el ambiente empresarial, y no todos los ingenieros de software poseen realmente títulos de ingeniería de universidades reconocidas.



Metodología tradicional

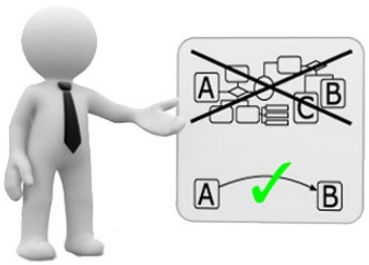
Ciclo de vida del software tradicional

La ingeniería de software requiere llevar a cabo numerosas tareas agrupadas en etapas, al conjunto de estas etapas se le denomina ciclo de vida.
Las etapas comunes a casi todo los modelos del ciclo de vida son:

- Análisis de requerimientos.
- Especificación de requerimientos.
- Arquitectura de software y datos.
- Desarrollo.
- Prueba.
- Documentación.
- Implementación.
- Mantenimiento.

Análisis de requerimientos

Extraer los requisitos de un producto de software es la primera etapa para crearlo. Mientras que los clientes piensan que ellos saben lo que el software tiene que hacer, se requiere habilidad y experiencia para reconocer requisitos incompletos, ambiguos o contradictorios. El resultado del análisis de requisitos con el cliente se plasma en el documento.



Especificación de requerimientos

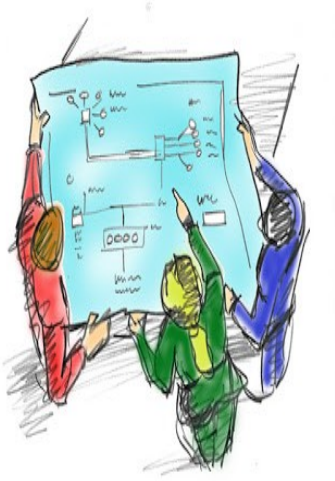
La especificación de requisitos describe el comportamiento esperado en el software una vez desarrollado.
Entre las técnicas utilizadas para la especificación de requisitos se encuentran:

- Caso de uso.
- Historias de usuario.

Siendo los primeros más rigurosas y formales, los segundas más ágiles e informales.

Arquitectura de software y datos

La integración de infraestructura, desarrollo de aplicaciones, bases de datos. El rol en el cual se delegan todas estas actividades es el del Arquitecto.
El arquitecto de software es la persona que añade valor a los procesos de negocios gracias a su valioso aporte de soluciones tecnológicas.
La arquitectura de sistemas en general, es una actividad de planeación, ya sea a nivel de infraestructura de red y hardware, o de software.



Diseño de software y datos

Diseño de software consiste en el diseño de componentes de una aplicación (entidades del negocio), generalmente utilizando patrones de arquitectura. El diseño arquitectónico debe permitir visualizar la interacción entre las entidades del negocio y además poder ser validado, por ejemplo por medio de diagramas de secuencia. Un diseño arquitectónico describe en general el cómo se construirá una aplicación de software.

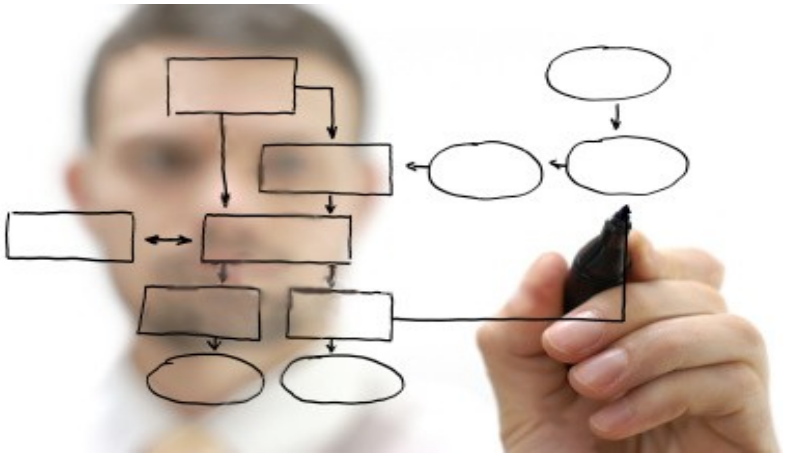
Arquitectura de software y datos

Para ello se documenta utilizando diagramas en metodología tradicional, por ejemplo:

- Diagramas de clases
- Diagramas de base de datos
- Diagrama de despliegue
- Diagrama de secuencia

Las herramientas para el diseño y modelado de software se denominan CASE, (Computer Aided Software Engineering) entre las cuales se encuentran:

- Enterprise Architect
- Microsoft Visio for Enterprise Architects



Desarrollo de software

Reducir un diseño a código puede ser la parte más obvia del trabajo de ingeniería de software, pero no necesariamente es la que demanda mayor trabajo y ni la más complicada. La complejidad y la duración de esta etapa está íntimamente relacionada al o a los lenguajes de programación utilizados, así como al diseño previamente realizado.

Testing de software

Las pruebas de software son las investigaciones empíricas y técnicas cuyo objetivo es proporcionar información objetiva e independiente sobre la calidad del producto a la parte interesada.

Tipos de pruebas por su ejecución

- Pruebas manuales (QC Quality Check Tester)
- Pruebas automáticas (QA Quality Automatizer Tester)

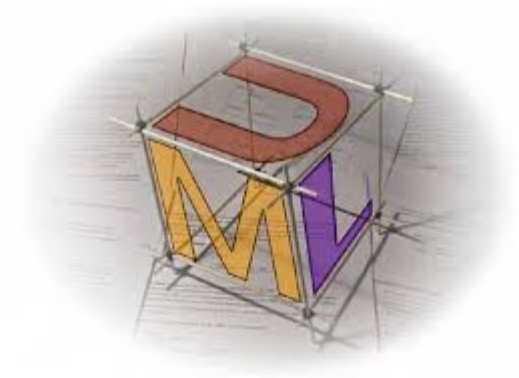
Enfoques de pruebas

- Pruebas de Caja blanca
- Pruebas de Caja negra

En java disponemos de varias herramientas para realizar testing, alguna de ellas es JUnit y Mockito.

Documentación

Todo lo concerniente a la documentación del propio desarrollo del software y de la gestión del proyecto, pasando por modelaciones (UML), diagramas de casos de uso, pruebas, manuales de usuario, manuales técnicos, etc; todo con el propósito de eventuales correcciones, usabilidad, mantenimiento futuro y ampliaciones al sistema.
En la actualidad se usan sistemas automáticos de documentación, y se prefiere el código auto documentado, (Java Doc /** */)



Implementación

La implementación se refiere al proceso post-venta de guía de un cliente sobre el uso del software o hardware que el cliente ha comprado. Esto incluye el análisis de requisitos, análisis del impacto, optimizaciones, sistemas de integración, política de uso, aprendizaje del usuario, marcha blanca y costes asociados.

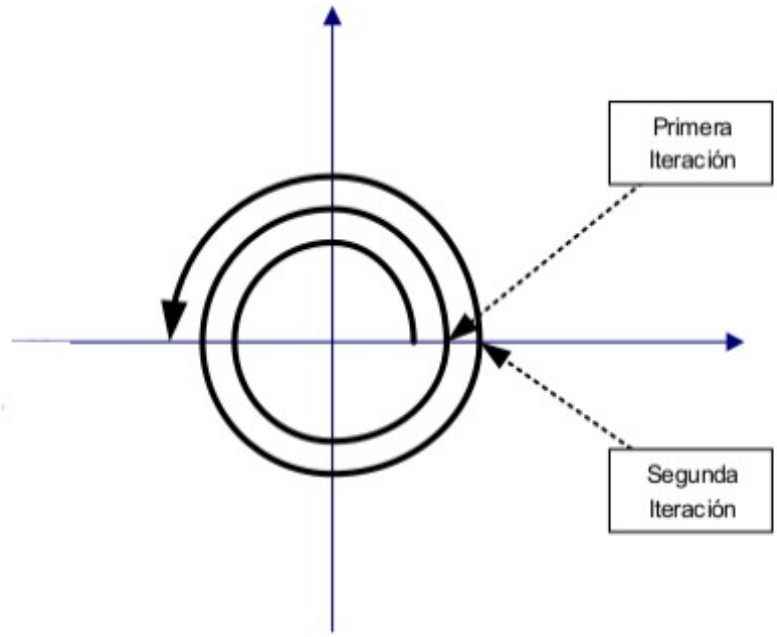
Mantenimiento y escalabilidad.

Fase dedicada a mantener y mejorar el software para corregir errores descubiertos e incorporar nuevos requisitos. Esto puede llevar más tiempo incluso que el desarrollo del software inicial. Alrededor de 2/3 del tiempo de ciclo de vida de un proyecto está dedicado a su mantenimiento. Una pequeña parte de este trabajo consiste eliminar errores (bugs); siendo que la mayor parte reside en extender el sistema para incorporarle nuevas funcionalidades y hacer frente a su evolución.

Metodología Ágil

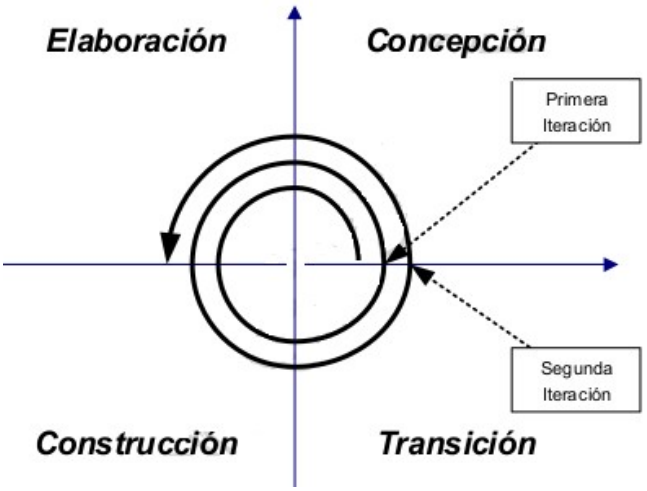
Ciclo de vida iterativo e incremental de los proyectos

Los proyectos pueden desarrollarse utilizando una aproximación iterativa e incremental. Esta es la forma en la cual se desarrollan los proyectos modernos.
Una iteración es un paso en el ciclo de vida de todo el proyecto y resulta en un incremento de dicho proyecto. Los incrementos se logran cumpliendo cuatro fases. Una vez realizadas las mismas se reitera el ciclo.
Algo que es muy conocido en el lenguaje popular es el concepto de “versión de un sistema o producto de software. Una versión se logra a través de cumplir una cierta cantidad de iteraciones (una o varias) luego de las cuales el proyecto se considera en condiciones para presentarlo a aquel que lo recibirá.

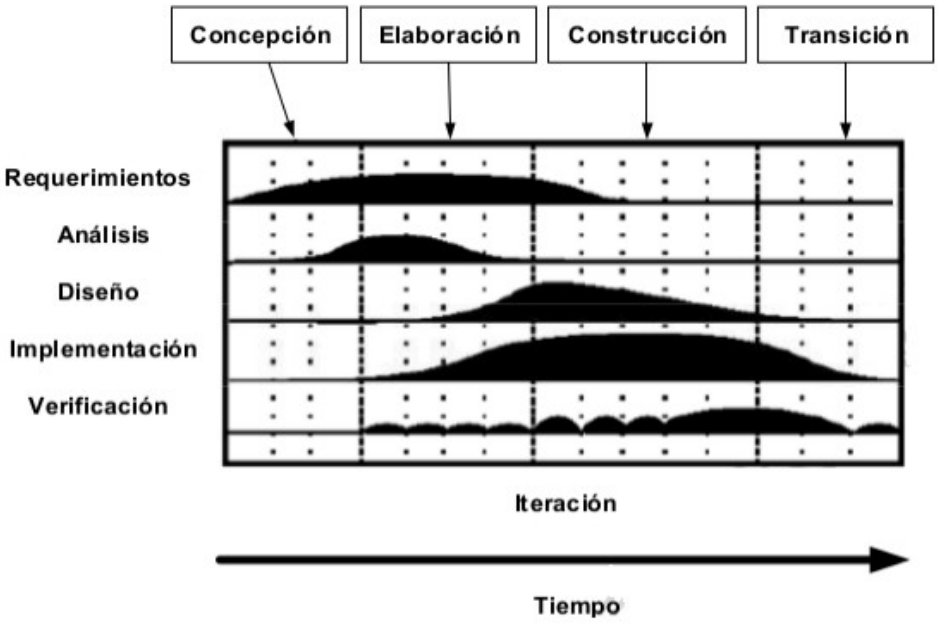


Un incremento se refiere al crecimiento, en términos del resultado final general de todo el proyecto. La sugerencia es siempre es que en lugar de atacar el problema en un solo paso, se itere a través de un ciclo y se produzcan actualizaciones incrementales. Cada incremento se construye sobre el otro, mejorando y corrigiendo los incrementos previos.

Fases principales del ciclo de vida iterativo e incremental
El ciclo de vida del desarrollo consiste en las siguientes cuatro fases:



- Concepción
- Elaboración
- Construcción
- Transición



Información relevada (Iteración 1 – fase de concepción – requerimientos)

La información relevada es aquella a partir de la cual se comienza a analizar cuáles son las abstracciones principales. Por lo general se encuentra en forma de una serie de párrafos que se obtienen al tratar de entender el problema a resolver y que se obtiene de un cliente, interesado o definición que indican en forma narrativa sus necesidades de solución.
Por lo general la información de este tipo no sigue un orden, sino que viene en un formato no secuencial, desordenado y confuso. También parte de ella se puede descartar si no es relevante a la solución que se desea obtener.
A la simplificación de la información relevada se la denomina extracción de requisitos y ordena en forma sencilla las necesidades de solución planteadas en la información relevada. Cuando el conjunto de requerimientos se ordena en un enunciado claro y conciso, sin redundancias ni perdida de información de necesidades a resolver, se lo llama enunciado de un problema.

Delimitar la solución

Cada vez que se realiza un análisis de un problema es importante determinar cuáles son los límites de la solución. Es muy fácil cuando se crean modelos concluir que otros objetos se pueden modelar a partir de las abstracciones principales. De alguna manera hay que establecer donde se debe parar y hay que tener presente en todo momento dichas limitaciones, mientras se analiza o se diseña una clase.
Es entonces fundamental encontrar dónde se “ubica” el sistema comparado con otros posibles, cuáles son las tareas que se tienen que realizar para desarrollarlo y hasta dónde llega cada una de las soluciones brindadas. Para determinar todo esto se debe comprender el significado de tres definiciones fundamentales:

- **Dominio:** Es el universo donde se encuentran todas las posibles soluciones de un determinado problema.
- **Contexto:** Es el subconjunto dentro del universo definido por el dominio que determina la solución del sistema a realizar.
- **Alcance:** El alcance de un sistema es la suma de las soluciones funcionales que presta y las que no. Esto implica que es un subconjunto del contexto delimitado por las funcionalidades a crear

dentro del mismo. **(Hasta donde llegamos con la iteración)**