7. Modelos de procesos de desarrollo

Introducción

Los modelos de proceso brindan una estructura para gestionar y guiar el desarrollo de software. Cada modelo establece una serie de etapas y pasos que deben llevarse a cabo para desarrollar de forma efectiva.

Modelos de procesos de desarrollos comunes ⭐

1. Cascada ⭐

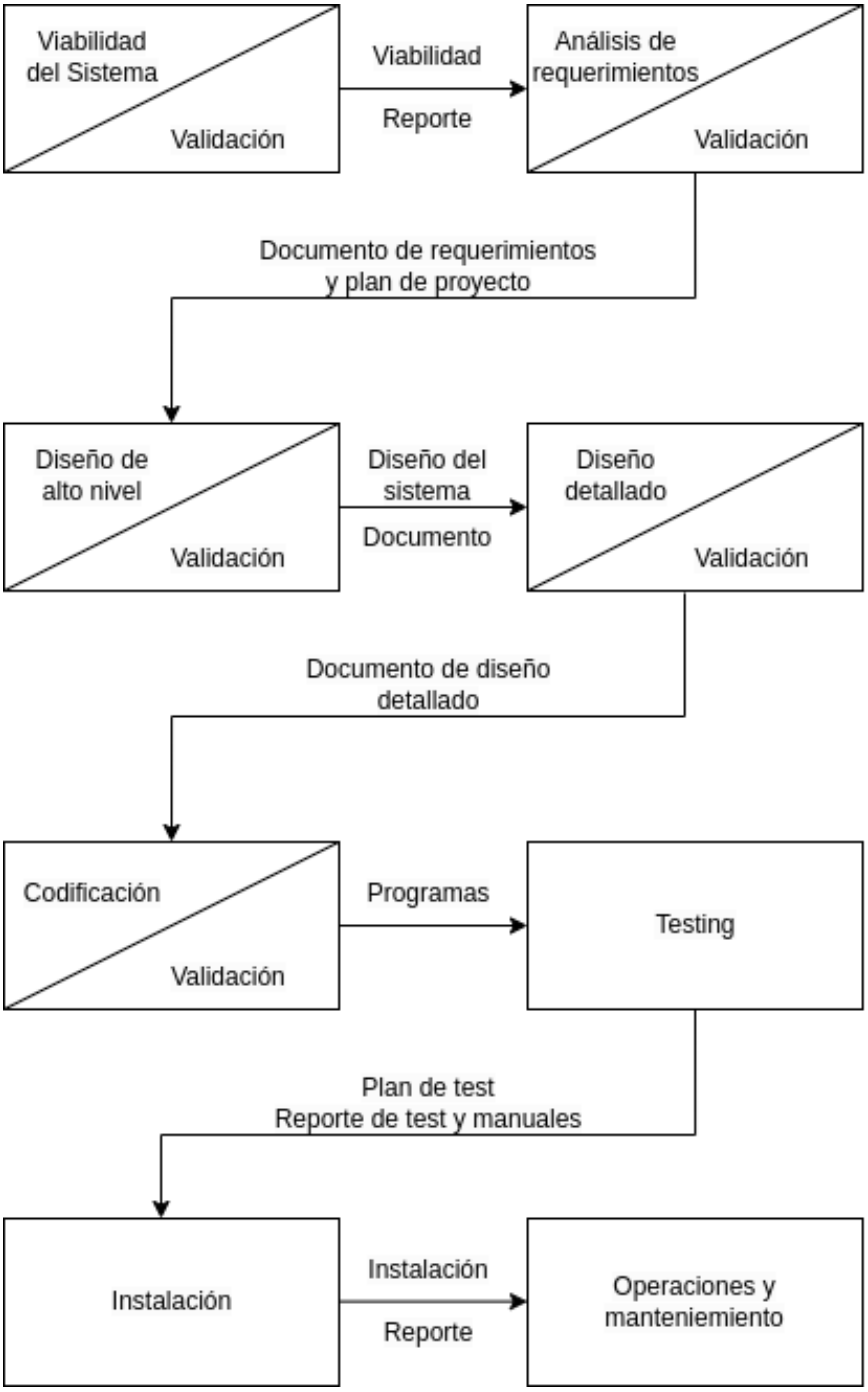
Es el modelo más viejo y muy usado. Divide el proceso de desarrollo en etapas que se ejecutan de forma lineal y secuencial.

Éste modelo es adecuado para proyectos donde los requerimientos son bien comprendidos y las decisiones sobre tecnologías son tempranas.

Etapas del modelo:

1. Análisis de requerimientos
2. Diseño de alto nivel
3. Diseño detallado
4. Codificación
5. Testing
6. Instalación

> 💡 Las etapas son secuenciales, debe completarse una para seguir con la siguiente.



Siguiendo el diagrama de arriba como guía, tenemos entonces que lo que éste modelo entrega en fase en fase es:

- Documento de requisitos / SRS
- Plan de proyecto
- Documento de diseño (arquitectura, sistema, diseño detallado)
- Programas
- Plan de test y reportes de test

- Código final
- Manuales del software (usuario, instalación, etc)

Además: reportes de revisión, de estado, etc…

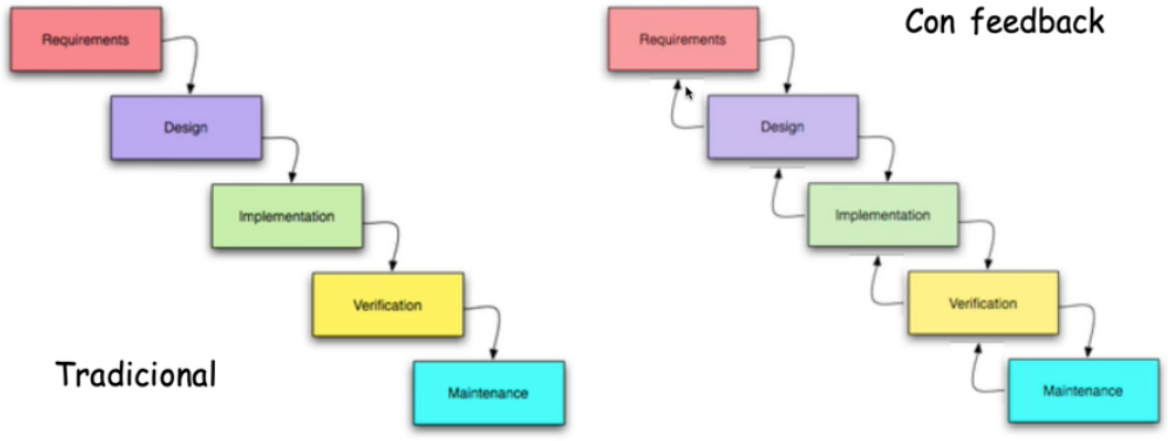
Ventajas

- Conceptualmente simple: al tener una división en fases, podemos trabajar sobre cada una de ellas de forma independiente. (éste es la ventaja más importante para mí)
- Enfoque natural a la solución del problema
- Fácil de administrar en un contexto contractual: esto quiere decir, que en contextos donde se usan contratos formales entre clientes y desarrolladores, se puede identificar fácilmente las responsabilidades y limitaciones de cada parte (tanto cliente como desarrollador) ya que las fronteras entre las fases están bien definidas y documentadas, lo que facilita la gestión.

Desventajas

El modelo de cascada tradicional no tiene en cuenta el feedback de cada etapa para la etapa anterior. Por ejemplo, en la etapa de requerimientos, los requisitos recopilados se desean mantener sin cambios a lo largo del resto del proyecto. Ésto no es muy bueno ya que puede que en otras etapas obtengamos feedback para una anterior que requiera modificaciones.

Para ello, se tiene un approach de cascada distinto donde en cada fase se tiene la posibilidad de volver a UNA anterior (no es que desde verificación se puede saltar a requerimientos, sólo un salto).

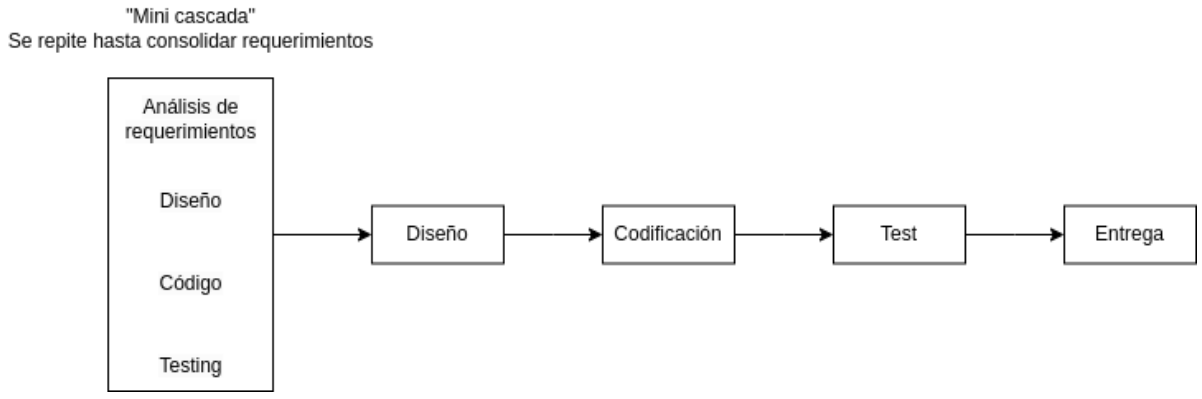


2. Prototipado ⭐

Usado para abordar las limitaciones del modelo en cascada, especialmente lo que respecta a la especificación de los requisitos del software. En lugar de intentar congelar los requisitos del software solo mediante conversaciones y debates, en este modelo se construye un prototipo del sistema antes de desarrollar la versión final.

✅ En este modelo, a través del prototipado, se consigue feedback del cliente.

Consiste en tener una **“mini-cascada”** que se repite hasta que los requerimientos se consolidan. En este proceso de “mini-cascada” es en donde se desarrollan los **prototipos** para su análisis. Luego de la consolidación de requerimientos, se pasa a las etapas finales de diseño, codificación, testeo y entrega.



Los prototipos deben ser descartados luego de la consolidación de requerimientos, ya que no pueden ser productos finales debido a su incompletitud.

> 💡 Los costos de prototipado deben mantenerse bajos, por lo que sólo se deben prototipar aspectos que necesiten aclararse. No hace falta desarrollar un prototipo de calidad por lo que es conveniente omitir manejo de excepciones, recuperación, estándares, testing, entre otros.

Ventajas

- Mayor estabilidad en los requerimientos. (esta es la ventaja más importante para mí)
- Los requerimientos se congelan más tarde, por lo que se entiende lo que uno quiere.

- La experiencia de construcción de los prototipos ayudan a idear el producto final.

Desventajas

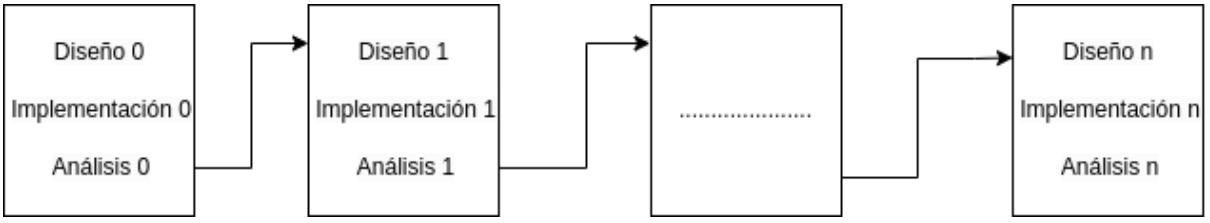
- Costos en tiempo y $$$$ (ya que hacer cada prototipo cuesta money).

Aplicación

Éste modelo debe aplicarse sólo cuando los requerimientos son difíciles de determinar y la confianza en ellos es baja.

3. Desarrollo Iterativo ⭐

Usado para abordar el problema de “todo o nada” del modelo cascada. Éste modelo fusiona las ventajas de cascada y prototipado, y consiste en realizar un desarrollo incremental, en donde cada incremento es completo en sí mismo. Puede verse como una “secuencia de cascadas”.



Éste modelo, al tener lo que se desarrolla acotado en incrementos, es más fácil y organizado para realizar tests también.

Tenemos dos enfoques para el desarrollo iterativo:

Enfoque a: Modelo con mejora iterativa

Bien, el modelo con mejora iterativa esta organizado en **iteraciones** y utiliza lo que se llaman **listas de control de proyectos LCP**.

- Iteraciones: comprenden una serie de pasos (diseño, implementación y análisis) sobre un sistema parcial en donde se tratan funcionalidades a implementar.
- Lista de control de proyectos LCP: contiene una lista **ordenada** de las tareas a realizar/implementar para lograr el producto final.

Pasos:

1. Se crea una lista de control de proyecto LCP que contiene en orden las tareas a realizar para lograr la implementación final.

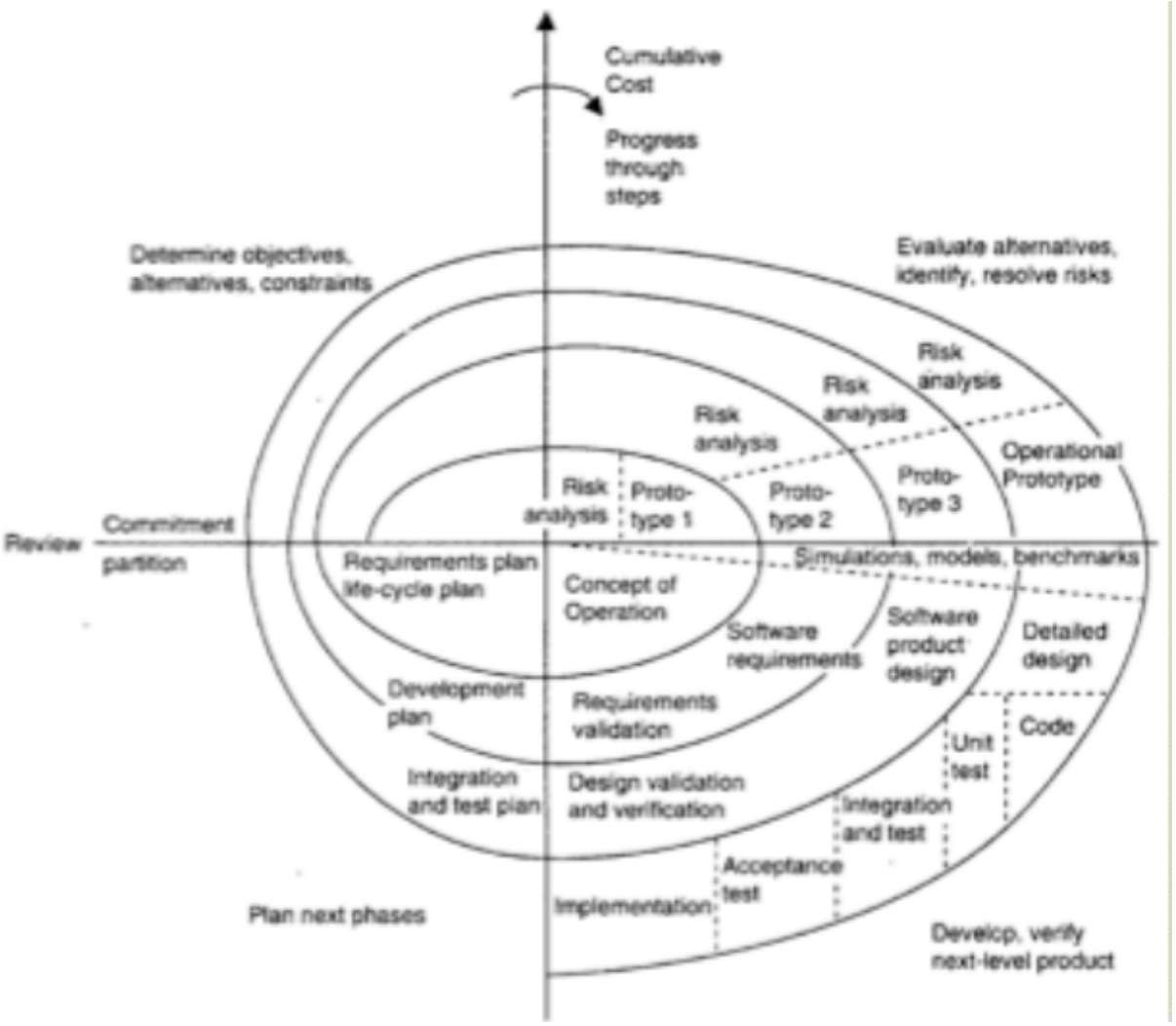
2. Se agarra un item/funcionalidad de la lista LCP y se busca eliminarla de la misma a través de su desarrollo haciendo diseño, implementación y análisis.
3. Se actualiza la lista
4. Se repite el proceso hasta vaciar la lista.

> 💡 La lista guía los pasos de las iteraciones y lleva las tareas a realizar. Cada entrada en la LCP es una tarea a hacerse en un paso de iteración. Deben ser lo suficientemente simple como para poder completarse en una iteración.

Enfoque b: Modelo en espiral

Usado al inicio cuando apareció el modelo iterativo. Se tiene una espiral con ciclos, donde en cada ciclo:

1. Se identifican objetivos y alternativas para conseguir esos mismos objetivos, al igual que las restricciones.
2. Se evalúan las alternativas en base a objetivos y restricciones. Aquí se desarrollan estrategias para solucionar incertidumbre y riesgos.
3. Se desarrolla y se verifica el SW.
4. Se planea el próximo ciclo.



no se ve un choto

Ventajas

- Pagos y entregas incrementales: debido a que se organiza en iteraciones, se puede ir entregando parte del proyecto a medida que las iteraciones se completen. (esta es la ventaja más importante a mi parecer)
- Existe feedback para mejorar lo desarrollado.

Desventajas

- La arquitectura y el diseño pueden no ser óptimos.
- La revisión del trabajo hecho puede incrementarse debido a que cada iteración requiere de revisiones, ajustes y mejoras.
- El costo total puede ser mayor. Afectado también por el aumento en revisión y ajuste continuo. (la desventaja más importante para mi)

Aplicación

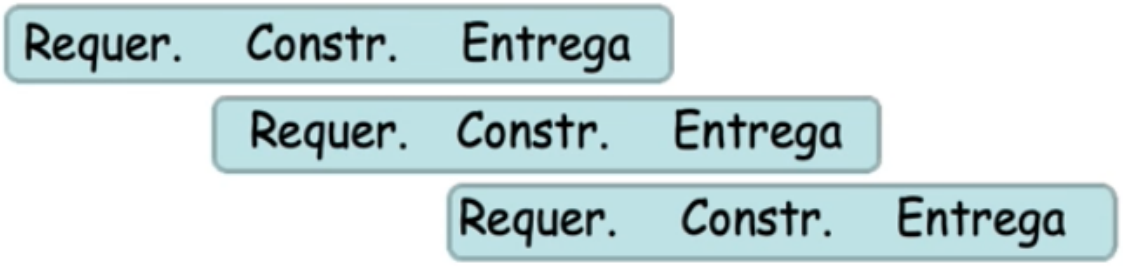
- Cuando el tiempo de respuesta es importante: es útil cuando se tienen proyectos que requieren respuestas rápidas a las necesidades del cliente.
- Cuando no se puede tomar el riesgo de proyectos largos: reduce incertidumbre y los riesgos al permitir que el cliente vea y pruebe el sistema gradualmente, contrario a proyectos largos.
- Cuando no se conocen todos los requerimientos: el desarrollo iterativo permite adaptar el software a medida que se obtiene una comprensión más profunda de los requisitos.

Nuevos enfoques 🚫

Desarrollo iterativo: Scrum 🚫

4. Timeboxing ⭐

Es un estilo iterativo donde tenemos iteraciones. Cada iteración se divide en partes iguales de tiempo (de ahí viene timeboxing: time boxes de igual tiempo), y comprenden individualmente una mini-cascada de distintas etapas (en este caso: requerimientos, construcción y entrega).



Este modelo lo que hace es fijar la duración de las iteraciones y asignando un grupo a cada etapa (es decir, un grupo a requerimientos, otro a construcción y otro a entrega), permite que puedan trabajarse varios proyectos al mismo tiempo. En la imagen de arriba, hay 3 proyectos en paralelo.

Cuando un equipo de una etapa finaliza, se lo pasa al equipo de la siguiente etapa.

Ventajas

- Todas las ventajas del modelo iterativo.

- Menor tiempo de entrega de proyectos. (la más importante para mí)
- Ejecución del proyecto distribuida.

Desventajas

- Grandes equipos de trabajo.
- Administración del proyecto mucho más compleja.
- Se necesita mucha sincronización. (la más importante para mi)

Aplicación

- Cuando los tiempos de entrega son cortos y muy importantes.

Resumen de modelos

Cascada

Fortalezas	Debilidades	Aplicación
Simple. Fácil de ejecutar. Intuitivo y lógico.	"Todo o nada": muy riesgoso. Req. se congelan muy temprano. Puede escoger hw/ tecno. vieja. No permite cambios. No hay feedback del usuario.	Problemas conocidos. Proyectos de corta duración. Automatización de procesos manuales existentes.

Prototipado

Fortalezas	Debilidades	Aplicación
Ayuda a la recolección de requerimientos. Reduce el riesgo. Sistemas finales mejores y más estables.	Comienzo pesado. Posiblemente mayores costos y tiempos. No permite cambios tardíos.	Sistemas con usuarios novatos. Cuando hay mucha incertidumbre en los requerimientos. Cuando las interfaces con el usuario son muy importantes.

Iterativo

Fortalezas	Debilidades	Aplicación
Entregas regulares y rápidas. Reduce riesgo. Acepta cambios naturalmente. Permite feedback del usuario. Prioriza requisitos.	Sobrecarga de planeamiento en cada iteración. Posible incremento costo total (trabajo en una iteración puede deshacerse en otra). Arq. y diseño pueden ser afectados con tantos cambios.	Para empresas donde el tiempo es esencial. Donde no puede enfrentarse el riesgo de proyectos largos. Cuando los requerimientos son desconocidos y sólo se comprenderán con el tiempo.

Timeboxing

Fortalezas	Debilidades	Aplicación
Todas las fortalezas del iterativo. Planeamiento y negociación un poco más fácil. Ciclo de entrega muy corto.	La administración del proyecto es compleja. Es posible el incremento de los costos. Equipos de trabajo muy grandes.	Donde es necesario tiempos de entrega muy cortos. Hay flexibilidad en agrupar características (features).