Ciclo de vida de un sistema

Prof. Ernesto Rodriguez

Universidad del Itsmo

erodriguez@unis.edu.gt

Metodologia de desarrollo

- Conjunto de practicas y lineamientos para estructurar el proceso de desarrollo.
- Existen varias opciones, cada una con ventajas y desventajas según la naturaleza del pryecto.
- Muchas metodologias tienen herramientas asociadas a ellas.
- Generalmente, las metodologias no se siguen al pie de la letra, queda a criterio del equipo de desarrollo decidir como utilizar la metodologia.

Grados de libertad en el Ciclo de vida de un Sistema

La metodologia se debe seleccionar y adaptar según la naturaleza del sistema a desarrollar. Algunas de las varaibles son:

Participación del cliente	Ninguna participación $\Leftrightarrow$ Parte del equipo
Requisitos	Rígidos $\Leftrightarrow$ Flexibles
Planeación	Predictivo $\Leftrightarrow$ Adaptivo
Confiabilidad del sistema	con errores $\Leftrightarrow$ sin errores
Validación	automatica $\Leftrightarrow$ manual $\Leftrightarrow$ formal
Equipo	Pequeño y centrado $\Leftrightarrow$ Grande y distribuido
Tamaño del sistema	usuarios contados $\Leftrightarrow$ millones de usuarios
Tiempo de desarollo	dias $\Leftrightarrow$ años
Presupuesto del cliente	start-up $\Leftrightarrow$ multi-nacional

Waterfall

- **Principio fundamental:** No se avanza a la siguiente etapa hasta terminar la etapa anterior.
- No es un modelo realista para la mayoria de sistemas.
- Sin embargo, puede utilizarse en algunos casos:
  - El costo de comunicación con el cliente es alto.
  - Alto grado de certeza en los requisitos. Ejemplo:
    - Migrar un sistema a tecnologia nueva.
    - Delegar un componente de un sistema a otro equipo.
    - Un sistema con propositos demostrativos.
  - El sistema requiere un grado muy alto de confiabilidad: Modificar requisitos $\Rightarrow$ nuevos puntos de fallo.

Waterfall

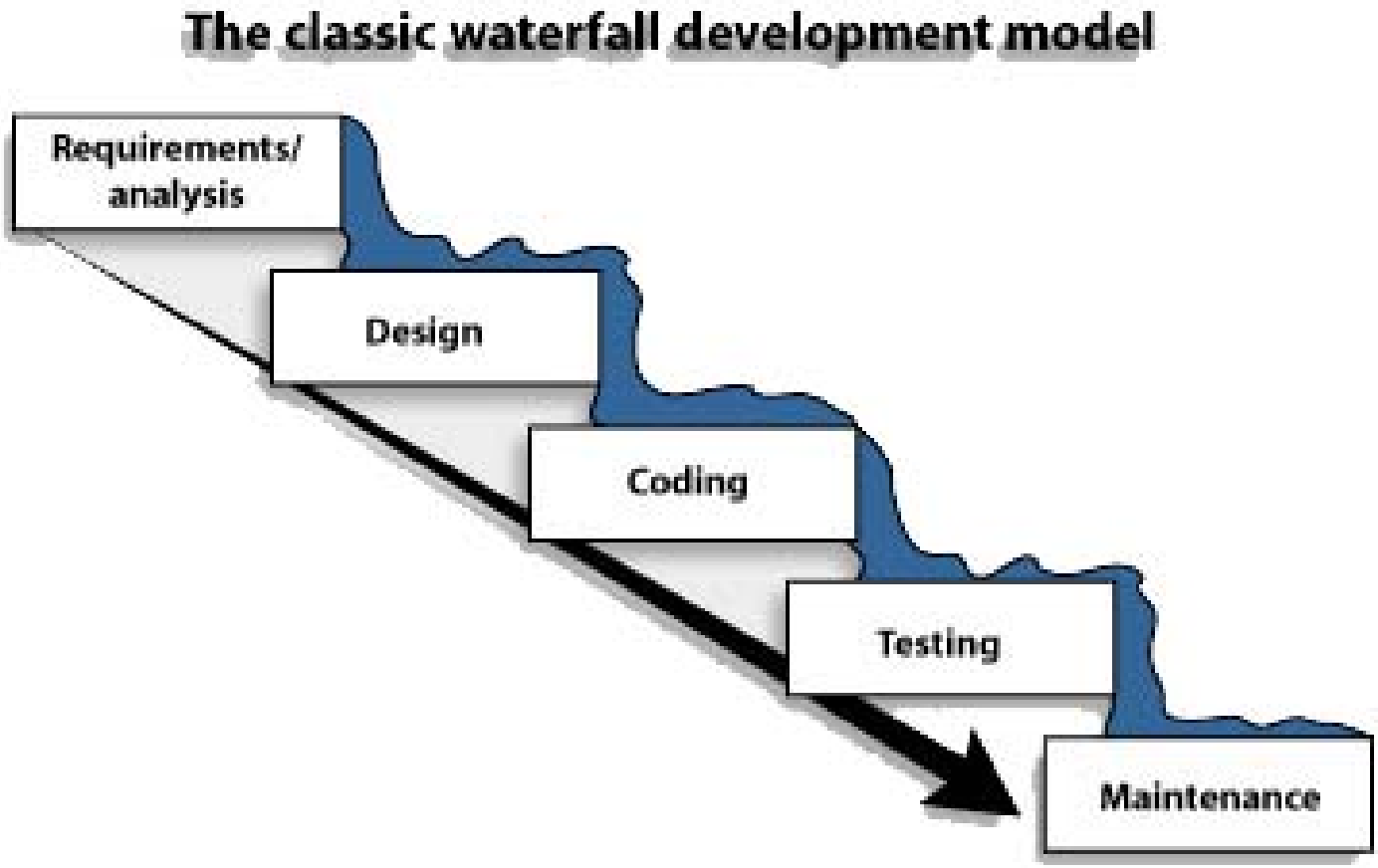


Figure: *Obtenido de [2]*

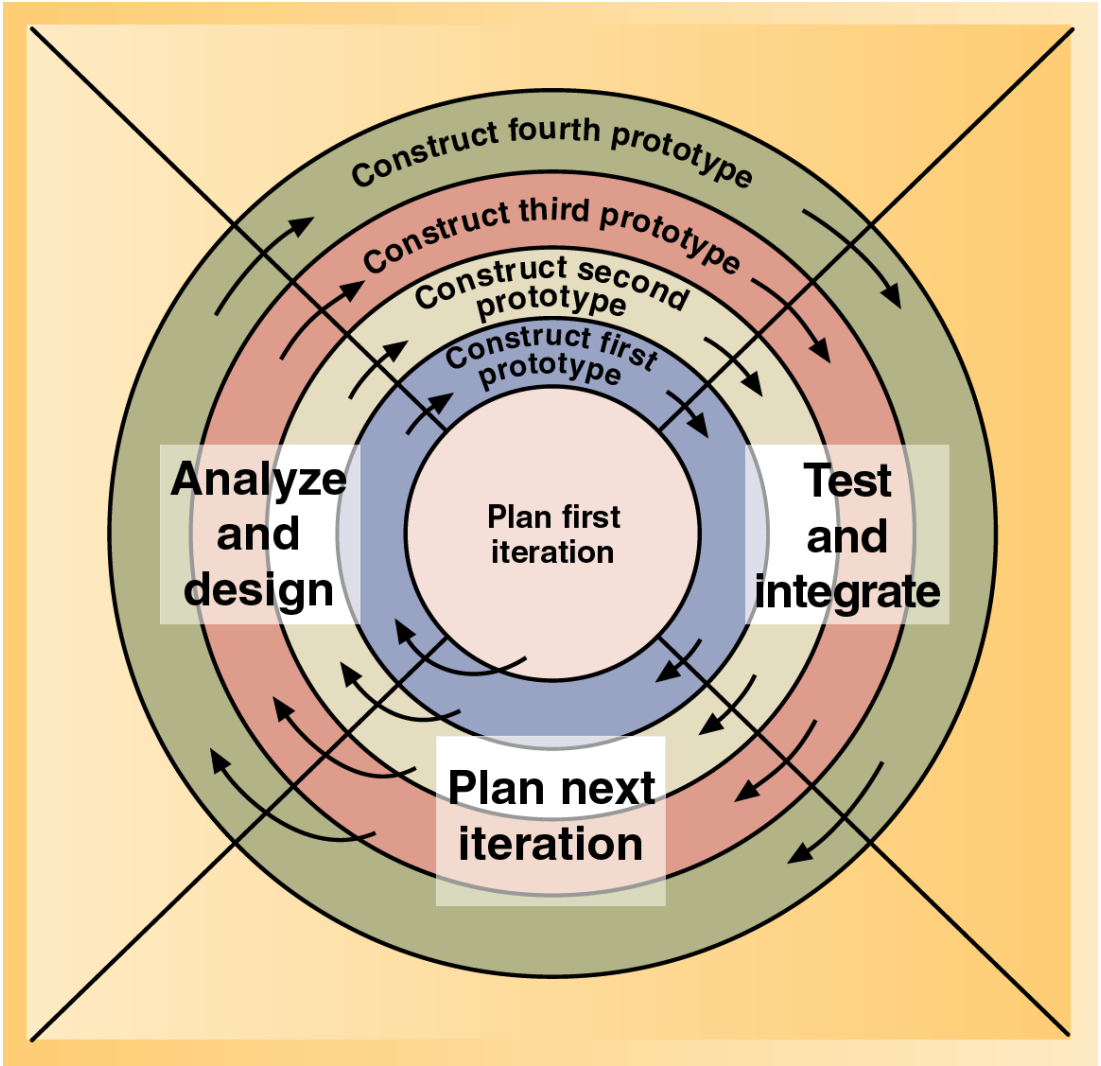
Problemas con Waterfall

- Los requisitos cambian constantemetne.
- Dificil utilizar los recursos humanos eficientemente.
- Los errores cometidos en una etapa se amplifican en las siguientes etapas.
- Muy dificil corregir los errores, en especial cuando el sistema ya se termino. Ej. Xorg

Metodologias espirales

- El proceso de desarrollo se estructura mediante iteraciones.
- En cada iteración se repiten los mismos pasos.
- Cada iteración se retro-alimenta de las iteraciones anteriores.
- Las iteraciones se pueden ordenar segun diferentes criterios:
  - Enfocarse en un sub-sistema a la vez.
  - En orden de riesgo.
  - Implementar funciones claves primero, luego complementos.
- Con buenas practicas y un buen pipieline de desarrollo, los errores causados por cambios de requisitos se pueden mitigar.
- Existen varios modelos[1]:
  - Extreme programming.
  - Rational unified process (RUP)
  - Scrum

Metodologias espirales



Ventajas de las metodologias espirales

- Inclúyen el cambio como parte del proceso:
  - Aprender de iteraciones pasadas.
  - Minimizan el riesgo y costo de introducir cambios.
  - Minimizan la propagación de errores.
- Entregan valor temprano:
  - Satisfacer al cliente desde temprano.
  - Reduce el riesgo de fallar.
  - Se enfoca en la funcionalidad más importante en cada etapa.
- En algunos casos, el cliente es parte del equipo:
  - Le permite al cliente dar retro-alimentación desde temprano.
  - Mantiene al cliente informado.
  - Facilita el diseño del sistema.
- Mayor utilizacion de recursos humanos:
  - Cada miembro del equipo puede trabajar en su campo durante todas las iteraciónes.
  - No hay necesidad de esperar a que completen fases anteriores para trabajar.
  - Todos los miembros tienen la misma importancia en todas las iteraciones.

Scrum

- Organiza el desarrollo en sprints de 2-4 semanas de duración.
- Los requisitos son capturados como historias en un backlog.
- Cada sprint debe producir un producto entregable.
- Antes de cada sprint debe haber una session de planeación.
- Los errores se van agregando al backlog según van siendo reportados.
- Requiere mucha transparencia y comunicación del equipo, para poder corregir y agendar errores.

Referencias

Aniruddha Ray.
Scrum and agile.
https://es.slideshare.net/ekmusibat/scrum-and-agile-sdlc-101.

Embeded Systems.
Waterfall model.
https://embeddedsystemsvvce.wordpress.com/tag/waterfall-model/.