

Y qué pasa con la aproximación tradicional



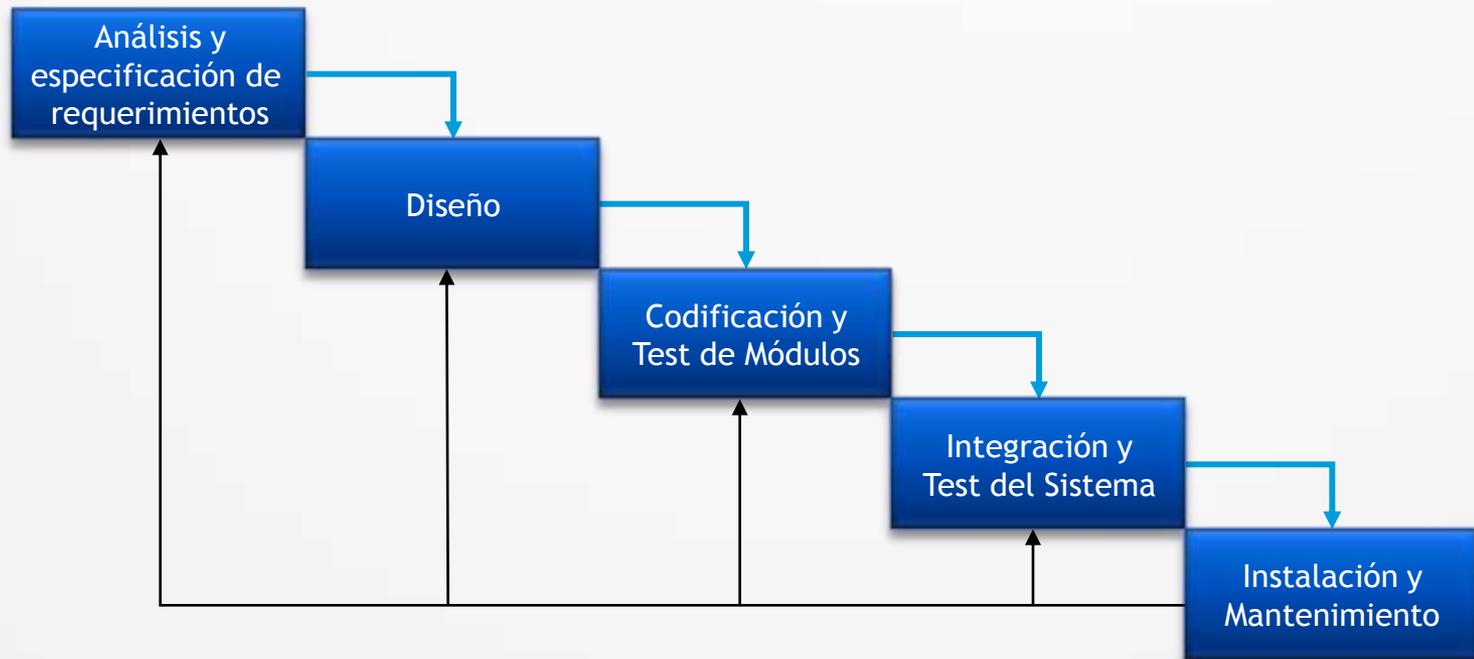
Big Design Upfront

- ▶ Tradicionalmente, para atacar la incertidumbre,
 - se trata de **definir todo desde un principio**
 - generando un **diseño y plan detallado**, que luego el desarrollador debe **ejecutar**
 - Conformado por requerimientos **abstractos y detallados**
 - En este modelo, el cliente **delega** su responsabilidad en el desarrollador
 - Y si este no cumple lo definido es **castigado** por multas en el contrato



Modelo de Cascada

- ▶ Esto da origen al Modelo de Cascada de desarrollo



Modelo de Cascada

- ▶ NO hay retroalimentación entre el problema de negocio y el producto desarrollado
 - Es fácil caer en el sesgo tecnológico (*cool features*)
 - Se pierde sincronía con las necesidades reales del negocio
 - Hasta que es ***demasiado tarde***

La paradoja de la previsibilidad

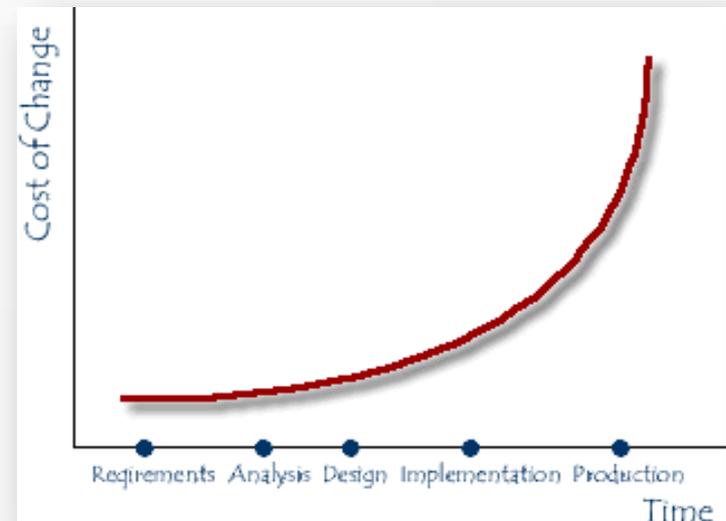
- ▶ Mientras más se trate de definir **a priori** un problema de software que posee incertidumbre, más probabilidad hay de error
- ▶ Y se agrega inflexibilidad ante los (muy) posibles cambios

Problema típico del ciclo de vida “cascada”

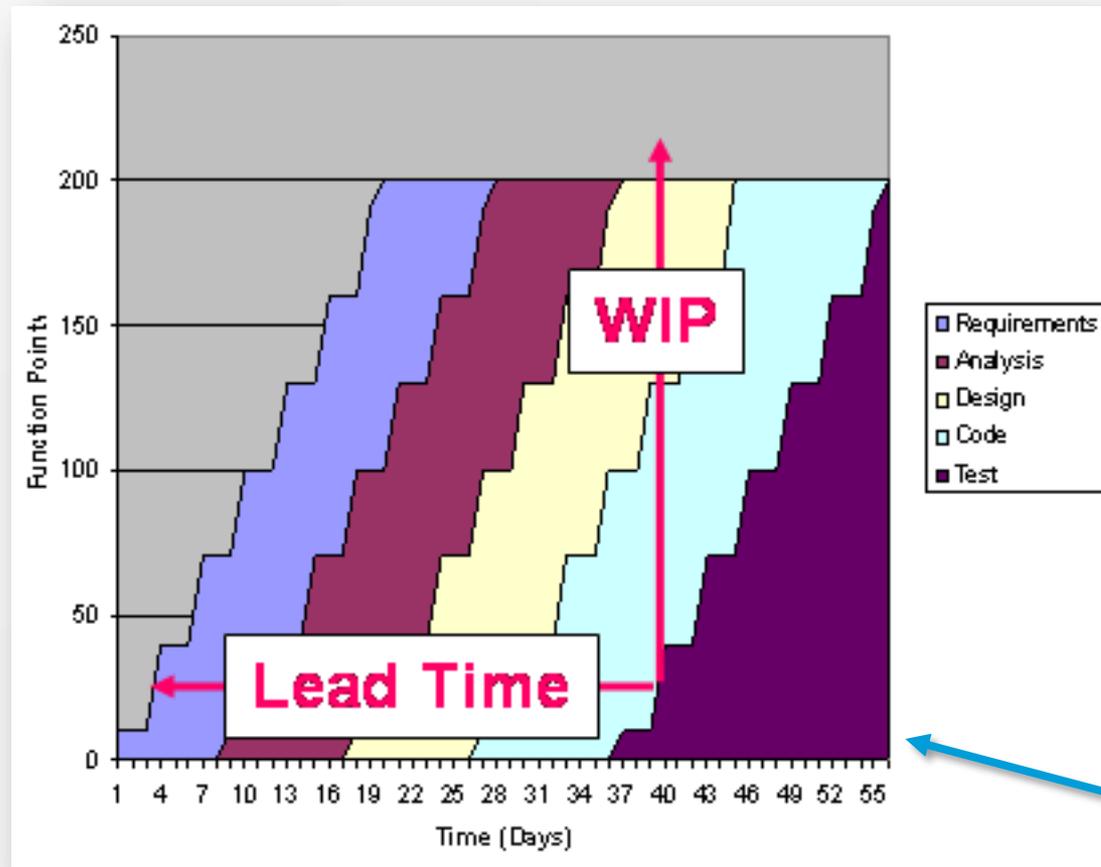


Paradigmas tradicionales del desarrollador

- ▶ Cree que a medida que avanza un proyecto, los cambios son más caros
 - Se tiende a desarrollar cosas que **se cree** que alguna vez se necesitarán
 - Tiende a priorizar según sus propia visión, no la del negocio
 - *Primero diseñamos **toda** la BD y sus mantenedores, y entonces implementamos las aplicaciones cliente*



Cómo se afecta la generación de valor



Única zona donde hay Valor para el cliente