Calidad en entornos ágiles

Juan Gabardini
75.46 Administración y Control de Proyectos Informáticos II
Facultad de Ingeniería - UBA

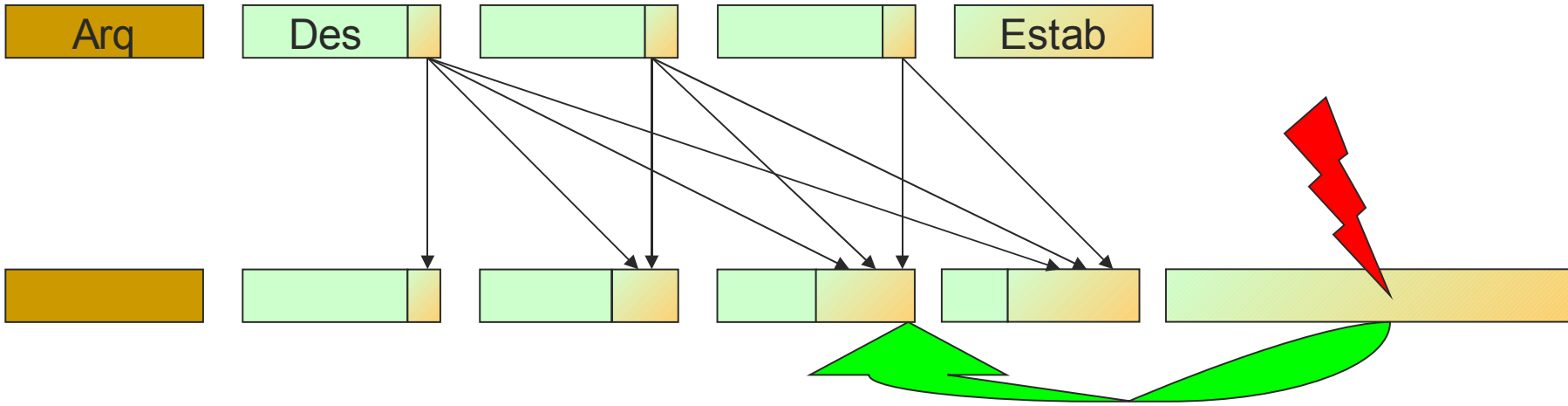
¿Que queremos lograr?

- Minimizar los riesgos y optimizar uso recursos
  - Planificar / predecir
    - Arquitectura detallada
    - Lista de tareas y dependencias estimadas
    - Especialización en las tareas
  - Inspeccionar / adaptar
    - Producto con calidad cercana a producción
    - Grupos auto-organizados

Calidad cercana a Producción

- Es la calidad definida por el cliente
  - Muy pocas veces es explicitada
    - Que cosas hay que corregir: todas
    - Cuanta prueba es necesaria: toda
  - Lleva a un mal uso de recurso
    - Mientras dura el proyecto, se corrige todo, cuando llega la fecha, salimos con lo que tenemos.
- Por qué mantenernos cerca?
  - Hay que lograr que en la balanza del cliente estén tanto la calidad cómo la funcionalidad

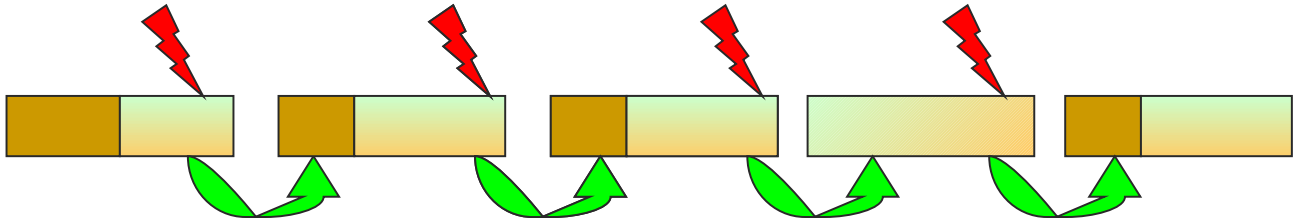
Desarrollo iterativo



Problemas del desarrollo iterativo

- El skill del grupo cambia a lo largo del tiempo
  - Más difícil adaptarse, hace más costoso los cambios.
- La prueba se vuelve costosa y repetitiva
  - Pérdida de motivación
  - Recorte de la prueba, pérdida de confianza

Desarrollo ágil



- Diseño en (casi) cada iteración
- La prueba con costo constante
- Siempre cerca de calidad de liberación

Consecuencias

- Grupo multidisciplinario y flexible
  - El grupo no puede cambiar continuamente, pero las necesidades cambian
  - La carga de trabajo por tipo de tarea son difíciles de predecir
- Los costos de los cambios deben mantenerse acotados
  - Se debe automatizar la prueba
  - Se debe refactorear

¿Que significa probar?

- Medir la calidad del producto para
  - Ayudar a mejorar la calidad
  - Tomar decisiones de liberación
  - Ayudar en el soporte
  - ...
- Sólo probar mientras aporte valor.

¿Que significa probar?

- Planificar
  - ¿Que y cómo probamos?
- Diseñar y construir
  - Condiciones, Datos entrada, Resultados
- Ejecutar
  - Prueba en sí misma
- Administrar
  - Defectos, Estado de Casos de prueba
- Informar resultado de la prueba

Tipos de prueba

- Unitaria
- Manual
  - Exploratoria
  - Basada en requerimientos
- Automática
  - Funcional
  - Stress
- …

Unitaria

Ventajas

- Ambiente de desarrollo: detección temprana
- Sencible a cambios de código
- Buena pruebas de caja blanca
- Cobertura de código

Desventajas

- Prueba no independiente
- No detecta problemas de instalación y ambiente

Exploratoria

Ventajas

- Rápido inicio y resultado
- Sin requerimientos detallados
- Buena prueba de usabilidad
- Conocimiento de la aplicación

Desventajas

- Muy dependiente del tester
- Difícil de reproducir
- Malo para funcionalidades complejas
- ¿Cuando terminar?

Manual - Basado en Req.

Ventajas

- Cobertura de requerimientos
- Bueno para funcionalidad
- Costo de casos bajo

Desventajas

- Dependiente del tester
- Requerimientos y aplicación conocidos.
- Alto costo ejecución y tedioso

Automático - Funcional

Ventajas

- Cobertura de requerimientos y código
- Bueno para funcionalidad
- Costo de ejecución bajos
- Oportunidades multiplicativas
- Independientes del tester

Desventajas

- Requerimientos y aplicación conocidos.
- Alto costo desarrollo y mantenimiento
- Respuesta lenta

Justificación pruebas automát.

- Costo Caso prueba
- Costo mantenimiento
- Frecuencia mantenimiento
- Costo Ejecución
- Administrativo
- Nro de builds

- (CTC + (cTC x f)) / nroBuilds + (cEjec + cAdm)