DISEÑO ORIENTADO A OBJETOS

Prof. Esp. Ing. Agustín Fernandez

Testing

- **Conceptos Previos:**
  - El Testing de Software nace aproximadamente en el año 1960 a partir de la crisis del desarrollo del software.
  - Se da en está época porque el software era muy complicado para elaborar, no se entregaba a tiempo, era muy costoso, y difícil identificar su avance porque **no es un tangible**.
  - Por está situación se empiezan a generar múltiples soluciones a la **crisis del software**, como:
    - La Calidad de los procesos de desarrollo de software.
    - El mejoramiento a la infraestructura del software.
    - Los frameworks.
    - Y por supuesto el Testing que nace como una respuesta para poder sobrellevar esta crisis.

Testing

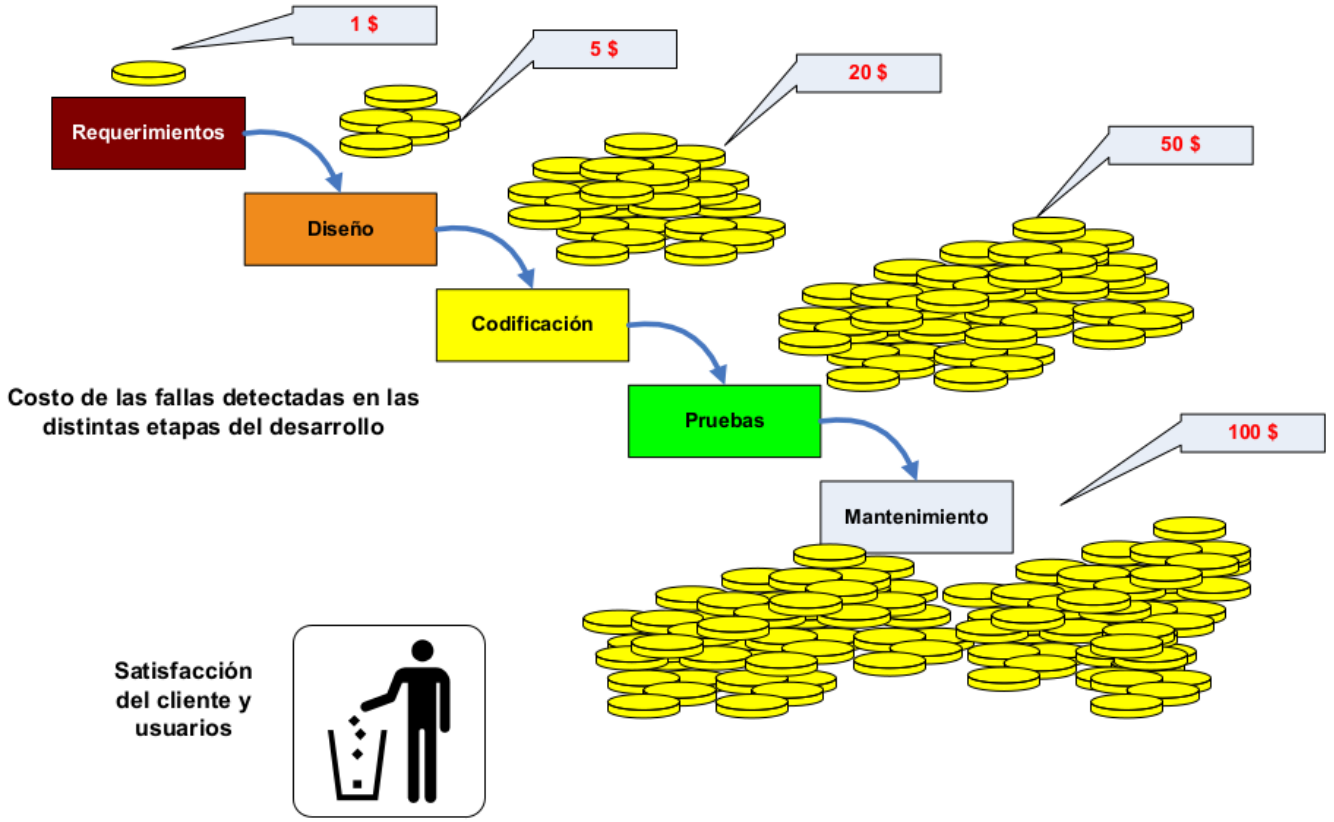
- **Conceptos Previos (Cont.)**
  - **Testing**: es el proceso orientado a demostrar que un programa no tiene errores:
    - Imposible.
    - Tentación a diseñar tests que no detecten errores.
  - **Testing**: es la tarea de demostrar que un programa realiza las funciones para las cuales fue construido.
  - **Testing**: es la tarea de probar que un programa realiza lo que se supone debe hacer. Aún haciendo lo esperado, puede contener errores.
  - **Testing**: es la ejecución de programas de software con el objetivo de detectar defectos y fallas. Proceso a veces destructivo o sádico.
  - **Test Exitoso**: aquel que detecta errores.
  - **Test No exitoso**: aquel que no los detecta.

Testing

- **Conceptos Previos (Cont.)**
  - **Error**: una equivocación de una persona al desarrollar alguna actividad de desarrollo de software.
  - **Defecto**: se produce cuando una persona comete un error.
  - **Falla**: es un desvío respecto del comportamiento esperado del sistema, puede producirse en cualquier etapa.
  - **Defecto** es una vista interna, lo ven los desarrolladores. **Falla** es una vista externa, la ven los usuarios.

Testing

- **Conceptos Previos (Cont.)**
  - La realización de tareas de pruebas conlleva un costo asociado que puede inducir a tomar decisiones de no realizarlas.
  - Ojo!!! no realizarlas también conlleva un costo asociado.



Testing

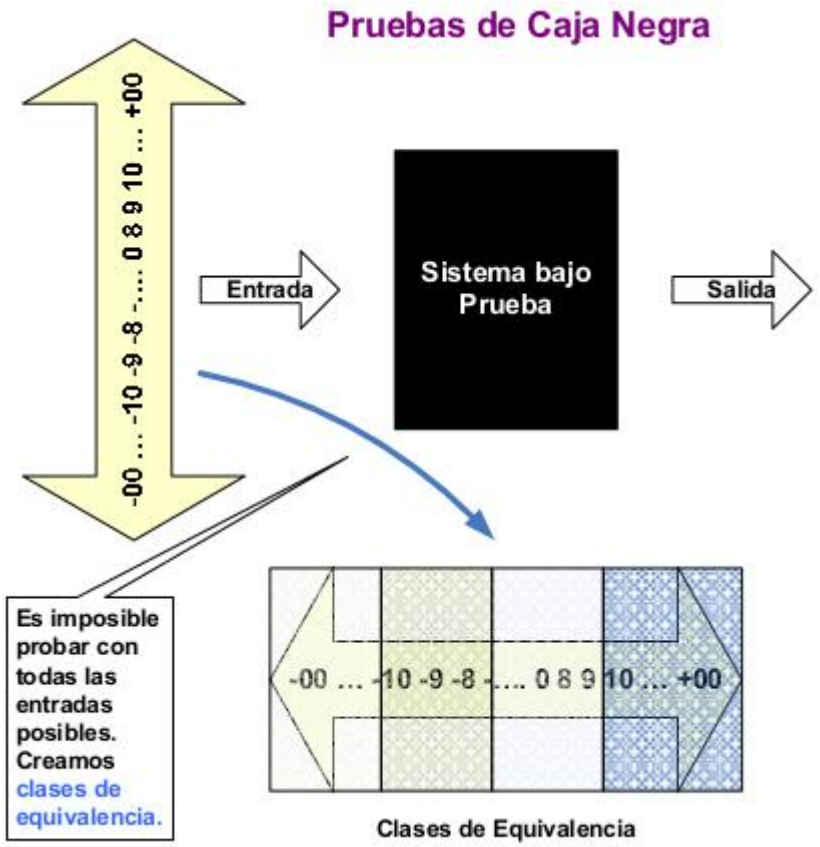
- **Principios**
  - Una parte necesaria de un test es la definición de los resultados esperados.
  - Un programador debe evitar probar su propio desarrollo.
  - Una organización no debe probar sus propios desarrollos.
  - Revise los resultados de los test en profundidad.
  - Los test deben incluir entradas inválidas e inesperadas así como las válidas y esperadas.
  - Revisar un programa para verificar que hace lo que se espera que haga es sólo la mitad de la prueba; la otra mitad consiste comprobar que no haga lo que no se espera.
  - No tirar los test a la basura a menos que el programa sea basura.
  - No planear esfuerzos de pruebas asumiendo que no se encontrarán errores.
  - La probabilidad de encontrar errores en una sección de un programa es proporcional al número de errores ya encontrados en esa sección.
  - El “testing” constituye una tarea creativa e intelectualmente desafiante.

Testing

- **Tipos de prueba**
  - Test caja negra: Pruebas funcionales sin acceso al código fuente de las aplicaciones, se trabaja con entradas y salidas.
  - Test de caja blanca: Pruebas con acceso al código fuente (datos y lógica). Se trabaja con entradas, salidas y el conocimiento interno.

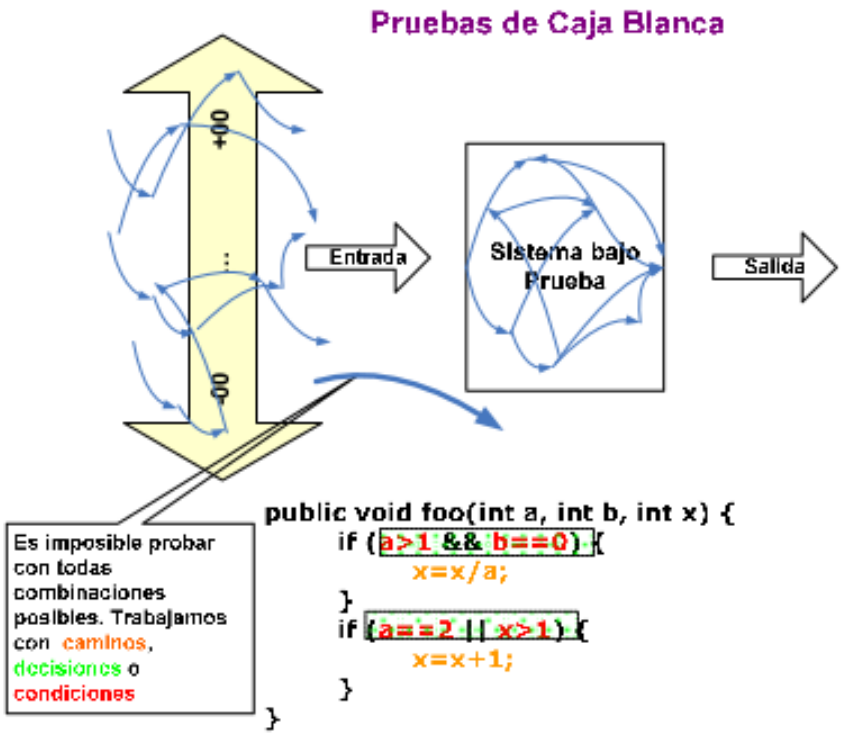
Testing

- **Tipos de prueba (cont.)**
- Test caja negra:



Testing

- **Tipos de prueba (cont.)**
- Test caja blanca:



Testing

- **Niveles de prueba**
  - Test Unitarios.
  - Test de Componentes/Test de Integración.
  - Test de Funcionalidad.
  - Test de Sistema.
  - Test de Aceptación.
  - Test de Instalación.

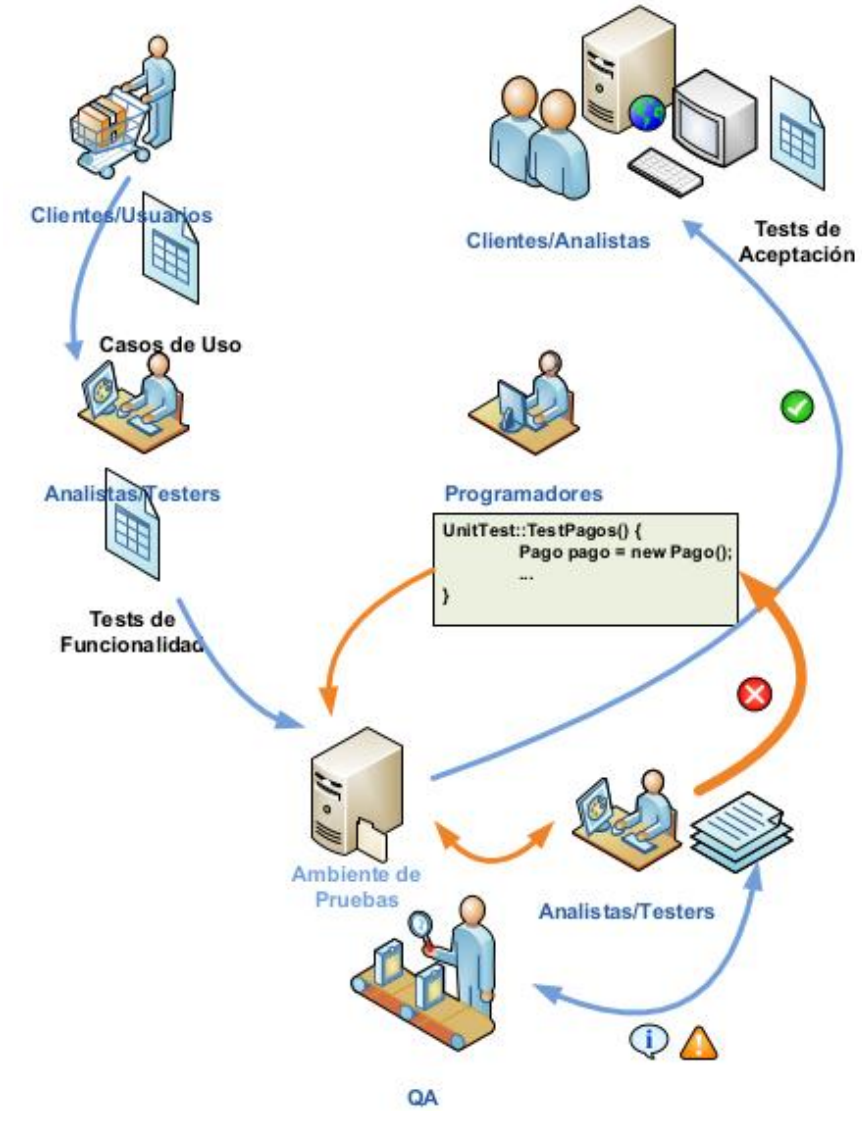
Testing

- **Niveles de prueba (Cont.)**

Niveles de pruebas			
Test	**Objetivo**	**Participantes**	**Ambiente**
Unitario	Detectar errores en los datos, lógica, algoritmos	Programadores	Desarrollo
Integración	Detectar errores de interfaces y relaciones entre componentes	Programadores	Desarrollo
Funcional	Detectar errores en la implementación de requerimientos	Testers, Analistas	Desarrollo
Sistema	Detectar fallas en el cubrimiento de los requerimientos	Testers, Analistas	Desarrollo
Aceptación	Detectar fallas en la implementación del sistema	Testers, Analistas, Cliente	Productivo

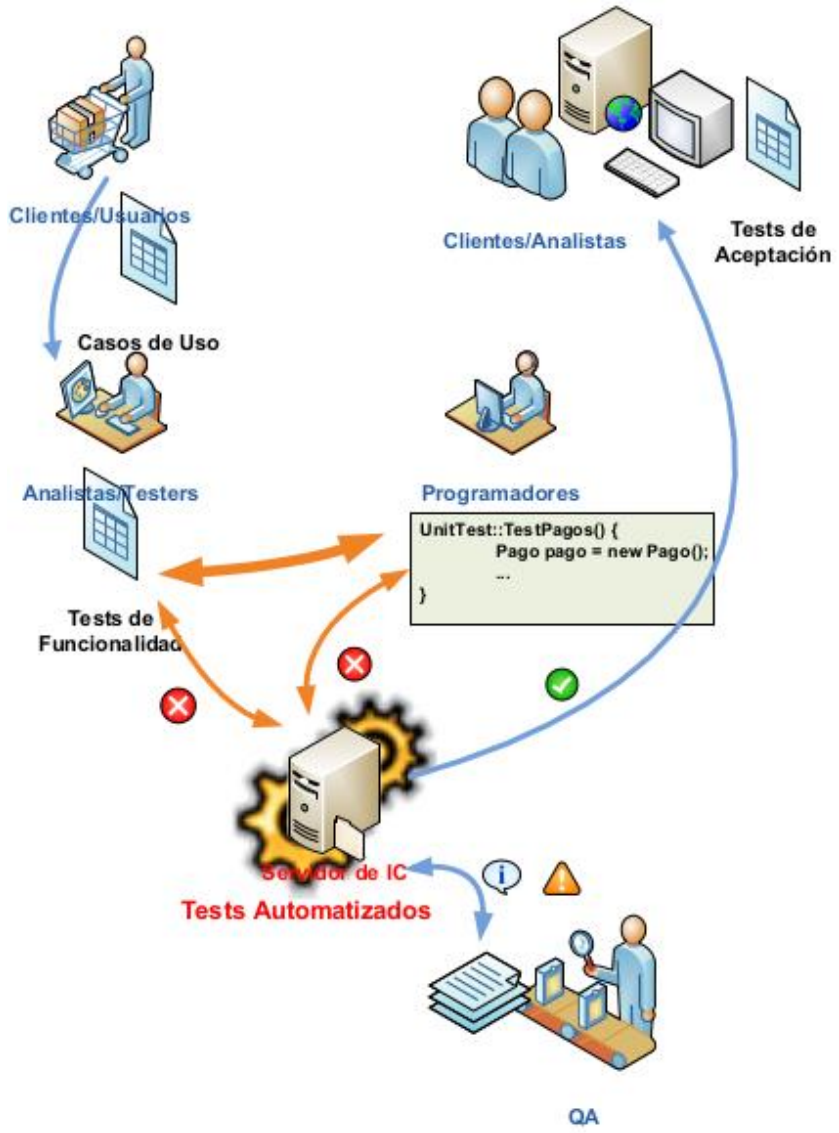
Testing

- **Trabajo con tests manuales**



Testing

- Trabajo con tests automatizados:



Testing

- Métodos de prueba:
  - **Test incrementales**: Testeo continuo, distribuye las pruebas de integración en la integración diaria del código compartido.
  - **Top Down:**
    - Se requieren Stubs para suplantar los módulos inferiores aún no implementados.
    - Los Stubs se quitan a medida que se desarrollan los diferentes módulos.
    - Realizar test de regresión sobre los módulos.
    - **Desventajas**:
      - Se retrasa la prueba del procesamiento real realizado generalmente en módulos de más bajo nivel.
      - Desarrollar Stubs que emulen a los módulos es mucho trabajo.

Testing

- Métodos de prueba: (Cont.)
  - **Bottom Up:**
    - Las pruebas comienzan en el más bajo nivel con la integración de algoritmos que realizan procesamiento.
    - Se escriben test que dan el contexto de ejecución a los módulos.
    - Se prueban los módulos.
    - Se desarrolla e integran funcionalidades del módulo superior y se repite
  - **Desventajas**:
    - Hasta que se logra un nivel determinado, la aplicación no es visible.
    - Problemas asociados a volumen, recursos y tiempo se prueban en etapas tardías.

Testing

- Lectura recomendada (en ingles): https://www.theatlantic.com/technology/archive/2017/09/saving-the-world-from-code/540393/

Testing

- Realizaremos:
  - Test unitarios usando Junit4 (https://junit.org/junit4/)
  - Test funcionales usando assertj-swing (https://joel-costigliola.github.io/assertj/assertj-swing-basics.html)
  - Test funcionales usando Sikuli IDE 1.1.1.3 (http://sikulix.com/)