UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática
Universidad Técnica Federico Santa María

Pruebas de Software

Ingeniería de Software

Hernán Astudillo & Gastón Márquez
Departamento de Informática
Universidad Técnica Federico Santa María

Muchas organizaciones concentran sus esfuerzos en arreglar los errores en vez de prevenirlos

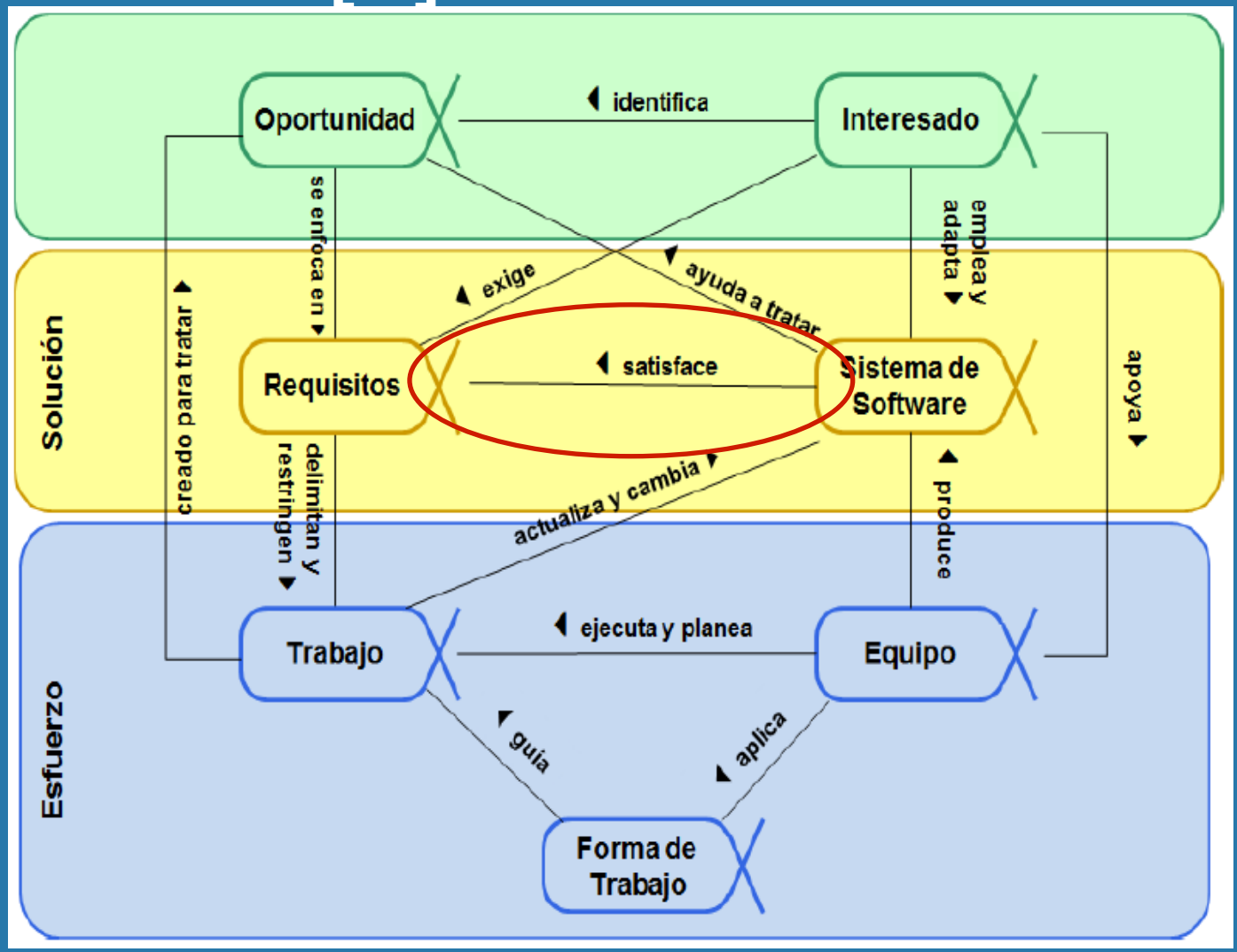
Contexto [1]

- A&DSW
  - ¿Validación del proyecto?
  - Si fue validado, ¿Cómo se hizo?
  - Algunas ideas: prueba y error, múltiples accesos, otros
- ¿Existirá algún proceso/modelo que ayude a realizar pruebas y asegurarnos que nuestro sistema de software fallará lo menos posible?

Contexto [2]

- ¿Por qué falla el Software?
  - Requerimientos equivocados <- testing
  - Requerimientos faltantes <- testing
  - Requerimientos imposibles de implementar <- testing
  - Diseño con defectos
  - Código con defectos
  - Diseño incorrectamente implementado

Contexto [3]


Oportunidad
identifica
Interesado
se enfoca en
exige
ayuda a tratar
emplea y adapta
Solución
creado para tratar
Requisitos
satisface
Sistema de Software
apoya
delimitan y restringen
actualiza y cambia
produce
Trabajo
ejecuta y planea
Equipo
Esfuerzo
guía
aplica
Forma de Trabajo

Distinción fundamental

Error

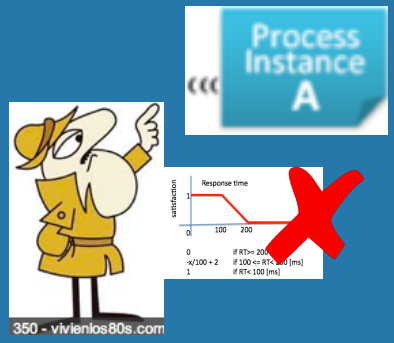
Defecto

```
try {
        URL url = new URL(query);
        HttpURLConnection con = (HttpURLConnection) url
                        .openConnection();
        con.connect();

        switch (con.getResponseCode()) {
        case HttpURLConnection.HTTP_OK:
                String s = streamToString(con.getInputStream());
                //converting json file to an string format
                s = processJsonFeed(
                String returnString =
                try {
                        JSONObject
                        JSONArray
                        //System.out.println ("There were " + data.

                        String friendID;
    for (int i = 0; i < data.length(); i++) {
        returnString="";
        returnString += "\n\t" + data.getString(i);
        JSONObject friend = new JSONObject(returnString);
        friendID = friend.getString("id");
        System.out.println("friendID = " + friendID);
        //perfil de mi amigo
        url = new URL(FB_BASE_URL
```

Falla



Una falla (*failure*) ocurre cuando un programa no se comporta adecuadamente - representa una propiedad del sistema en ejecución

Un defecto (*fault*) existe en el código - si se encuentra puede causar una falla - no hay defecto si el sistema no puede fallar

Un error (*error, mistake*) es una acción humana que resulta en software que contiene un defecto - un error puede llevar a incluir un defecto en el sistema, haciéndolo fallar

Los 7 principios de las Pruebas de Software [1]

1. La Prueba exhaustiva no es posible
   - Se necesita una cantidad óptima de pruebas basadas en la evaluación del riesgo del software
2. Clustering de defectos
   - Pequeños modulos que contienen la mayoría de los defectos detectados (Principio de Pareto, el 80% de los problemas son encontrados en el 20% de los modulos)
3. Paradoja del pesticida
   - Generalmente, se usa el mismo pesticida para eliminar insectos. Pero, los insectos pueden desarrollar una resistencia a través del tiempo, por lo tanto, se debe ocupar otro pesticida. En el fondo, no se pueden utilizar siempre la misma prueba
4. La Prueba muestra presencia de defectos
   - La Prueba habla sobre presencia de defectos, no sobre cómo eliminarlos

Los 7 principios de las Pruebas de Software [2]

5. Ausencia de error/defecto
   - Las pruebas de software no son meros defectos de detección, sino también sirven para comprobar que el software responde a las necesidades del negocio. La ausencia de error es una falacia, es decir, encontrar y corregir los defectos no ayuda si la construcción del sistema es inutilizable y no cumple con las necesidades y requisitos del usuario
6. Pruebas tempranas
   - El Testing debería empezar lo más pronto posible en el ciclo de desarrollo de software
7. Las Pruebas son un contexto dependiente
   - Las Pruebas dependen de un contexto lo que básicamente significa que, por ejemplo, la forma de probar un sitio de comercio electrónico será diferente de la forma en que se prueba una aplicación comercial

Proceso de Prueba

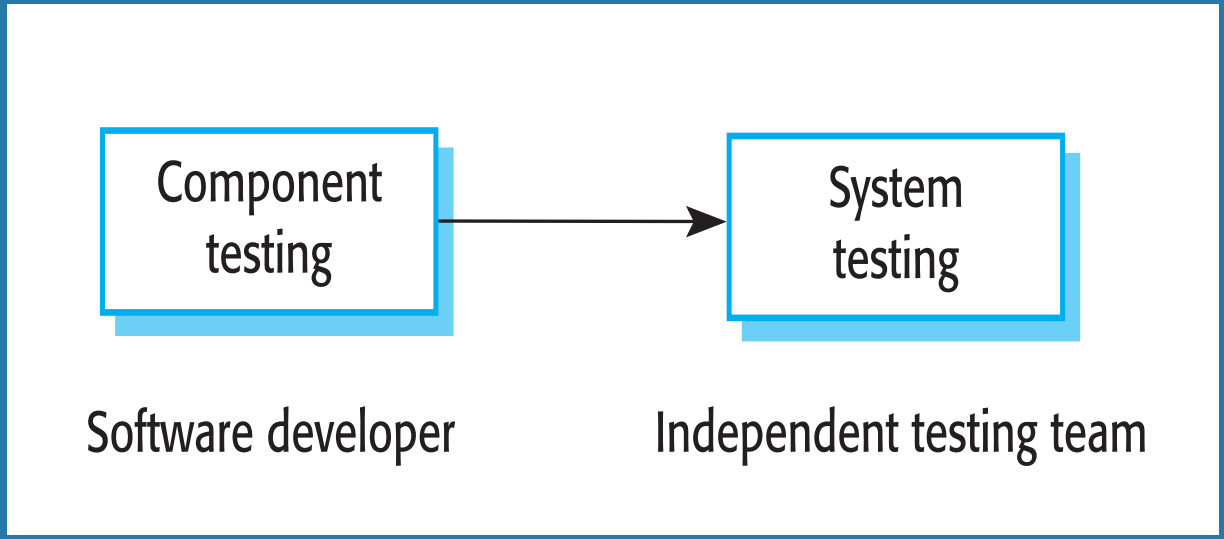
Objetivo de las Pruebas

- Descubrir defectos
- Una prueba será exitosa sólo cuando un defecto es descubierto
  - Identificación del defecto
  - Corrección del defecto -> NO (debugging)

El proceso de Prueba [1]

- Existen dos enfoques a la hora de hablar de pruebas
  - Prueba de componentes
    - Se verifica individualmente cada componente
    - Se analiza la responsabilidad del desarrollo del componente
    - Las pruebas se realizan, generalmente, de la experiencia de los desarrolladores
  - Prueba del sistema
    - Se hace prueba a un grupo de componentes integrados para crear sistemas o sub-sistemas
    - La responsabilidad de la prueba se traspasa a un equipo
    - Los test se basan en la especificación del sistema

El proceso de Prueba [2]



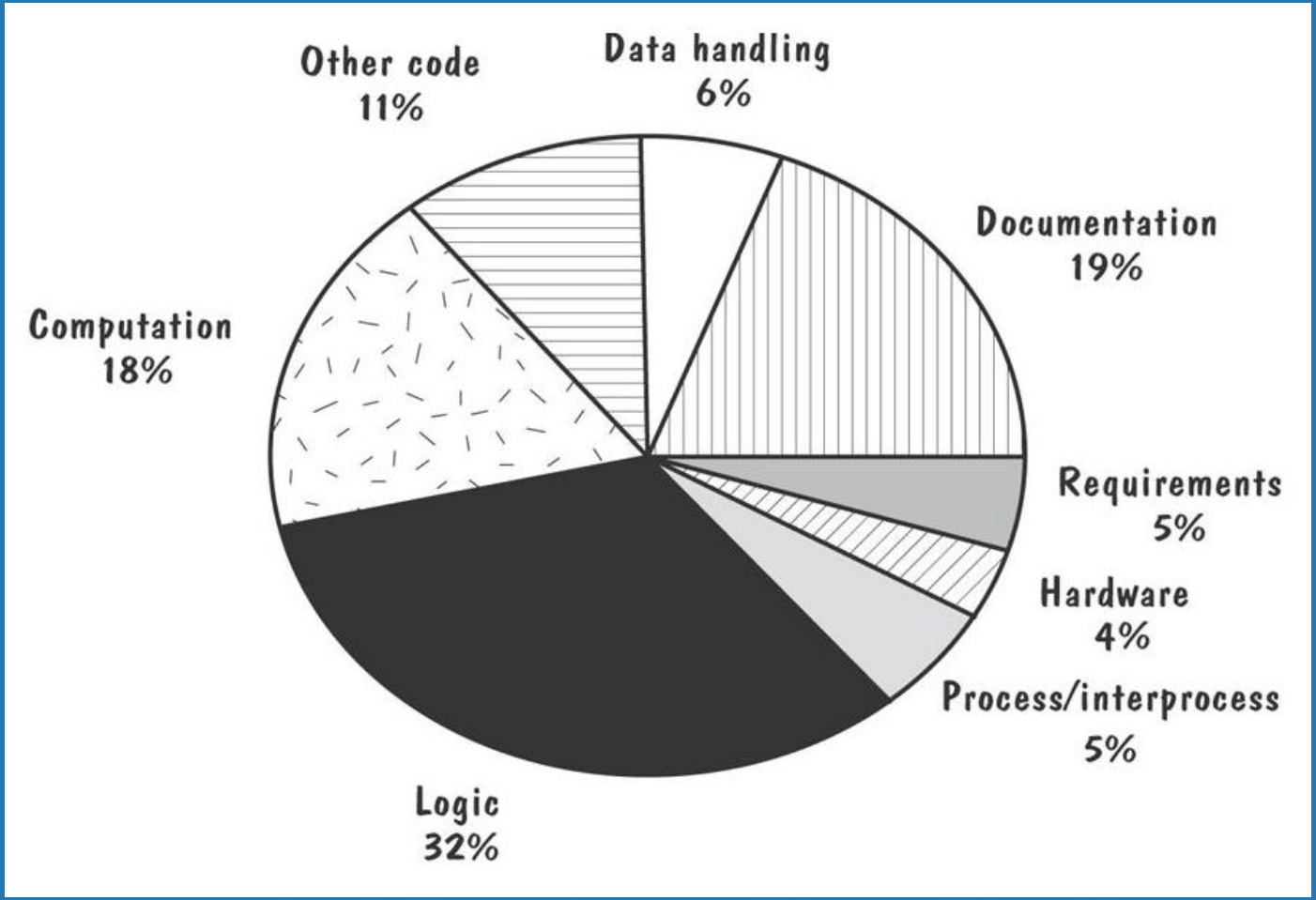
Defectos de las Pruebas

- El objetivo de las pruebas es encontrar defectos en los sistemas de software
- Una prueba exitosa es una prueba en donde se encuentran las causas que producen un comportamiento anómalo en un software
- Pero, la prueba sólo muestra la presencia de un defecto, no la solución

¿Qué tipos de defectos debería descubrir?

- Defectos en algoritmos
- Defectos en la precisión y cómputo
- Defectos en la documentación
- Defectos en capacidad o límites
- Defectos en el rendimiento
- Defectos acorde al estándar
- Entre otros

¿Dónde se repiten los defectos?



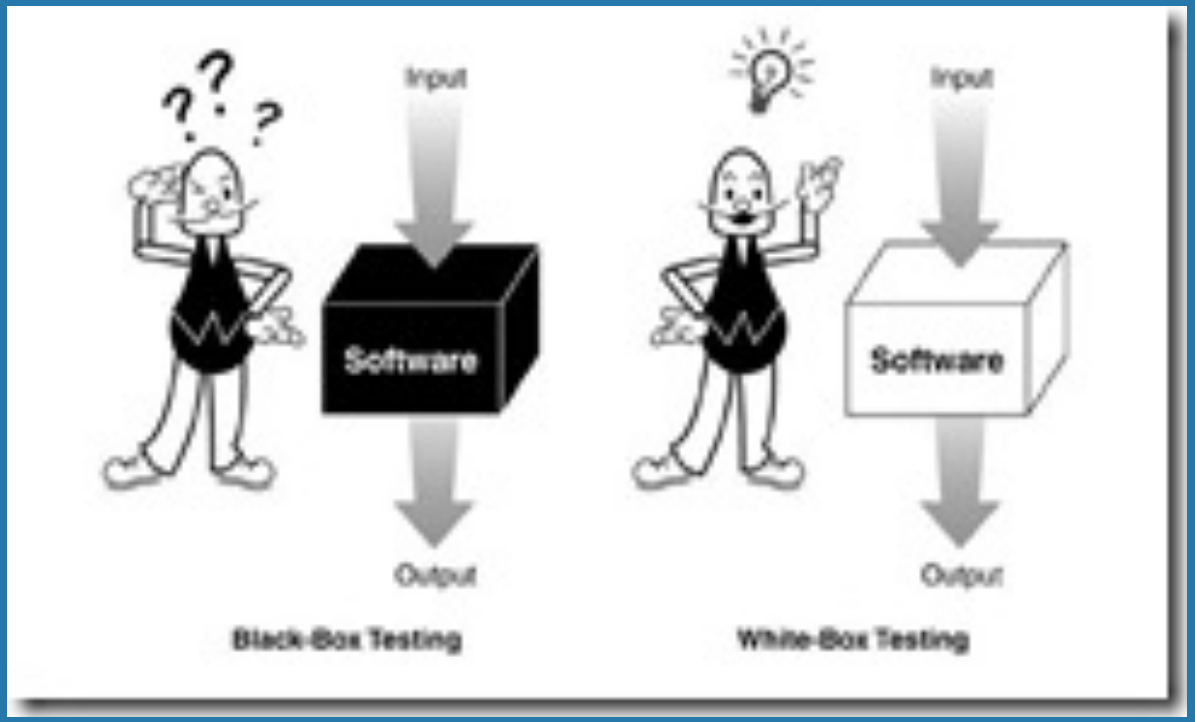
[Hewlett-Packard Division]

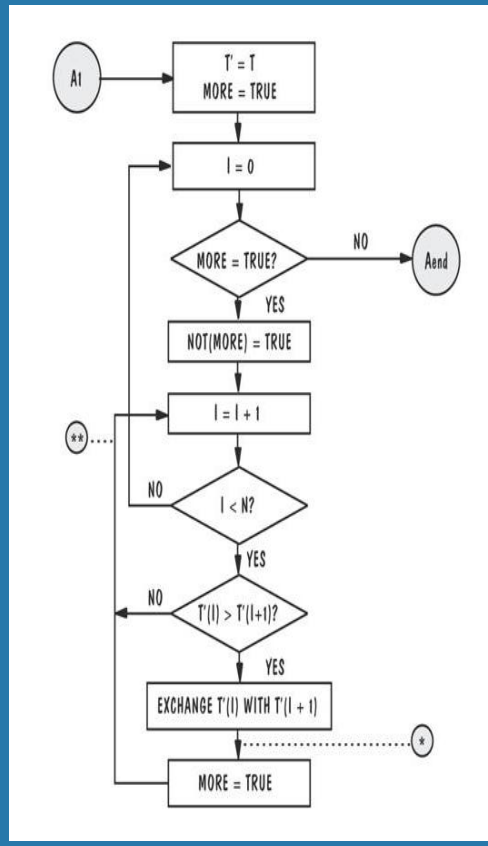
Objetivos de las Pruebas

- Validación
  - Demostrar a los desarrolladores y el sistema que los requerimientos establecidos han sido realizados exitosamente
  - Una prueba exitosa demuestra que el sistema de software funciona como se había establecido
- Defectos
  - Descubrir una falta o defecto en el sistema de software donde el comportamiento es incorrecto o no es conforme a las especificaciones establecidas

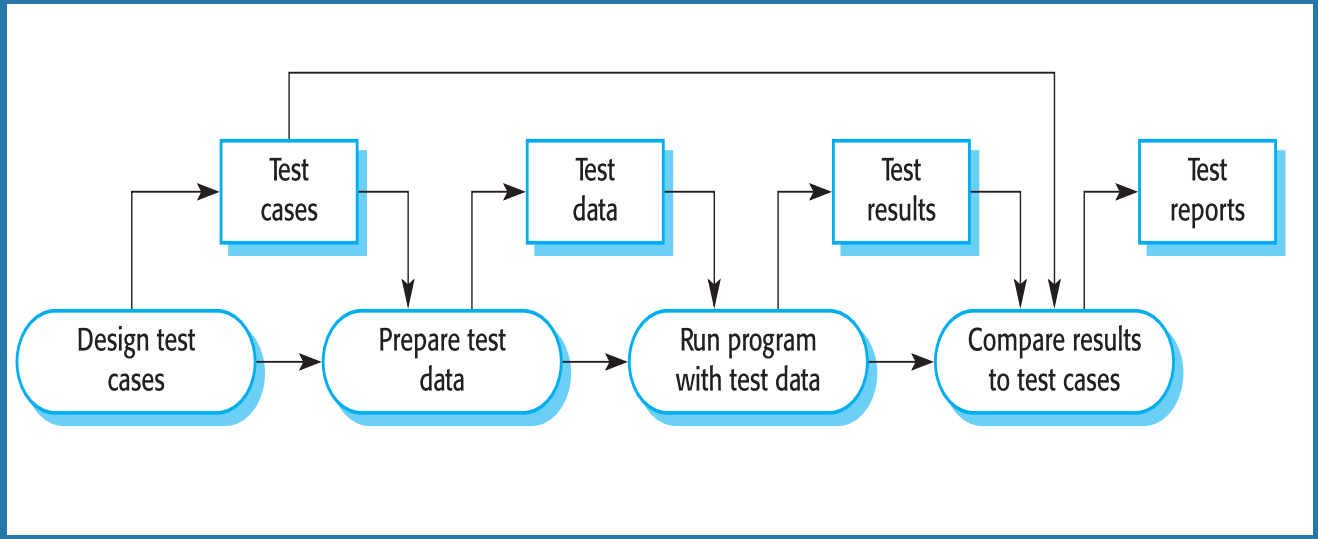
Vistas de los objetos bajo testing

- Caja negra: functionalidad de los objetos a prueba
- Caja blanca: estructura interna de los objetos a prueba





El proceso de Testing de Software



Políticas de Pruebas

- Solamente una prueba exhaustiva de Prueba puede mostrar que un sistema de software está libre de defectos. Sin embargo, una prueba exhaustiva es imposible
- Las políticas de Pruebas definen los enfoques que deben ser utilizados al momento de seleccionar las pruebas en un sistema
  - Todas las funcionalidades accedidas a través de menú deben ser testeadas
  - Las combinaciones de funcionalidad accedidas a través del mismo menú deben ser testeadas
  - Cuando el usuario ingresa peticiones al sistema, todas las funcionalidades deben ser testeadas

Tipos de Pruebas

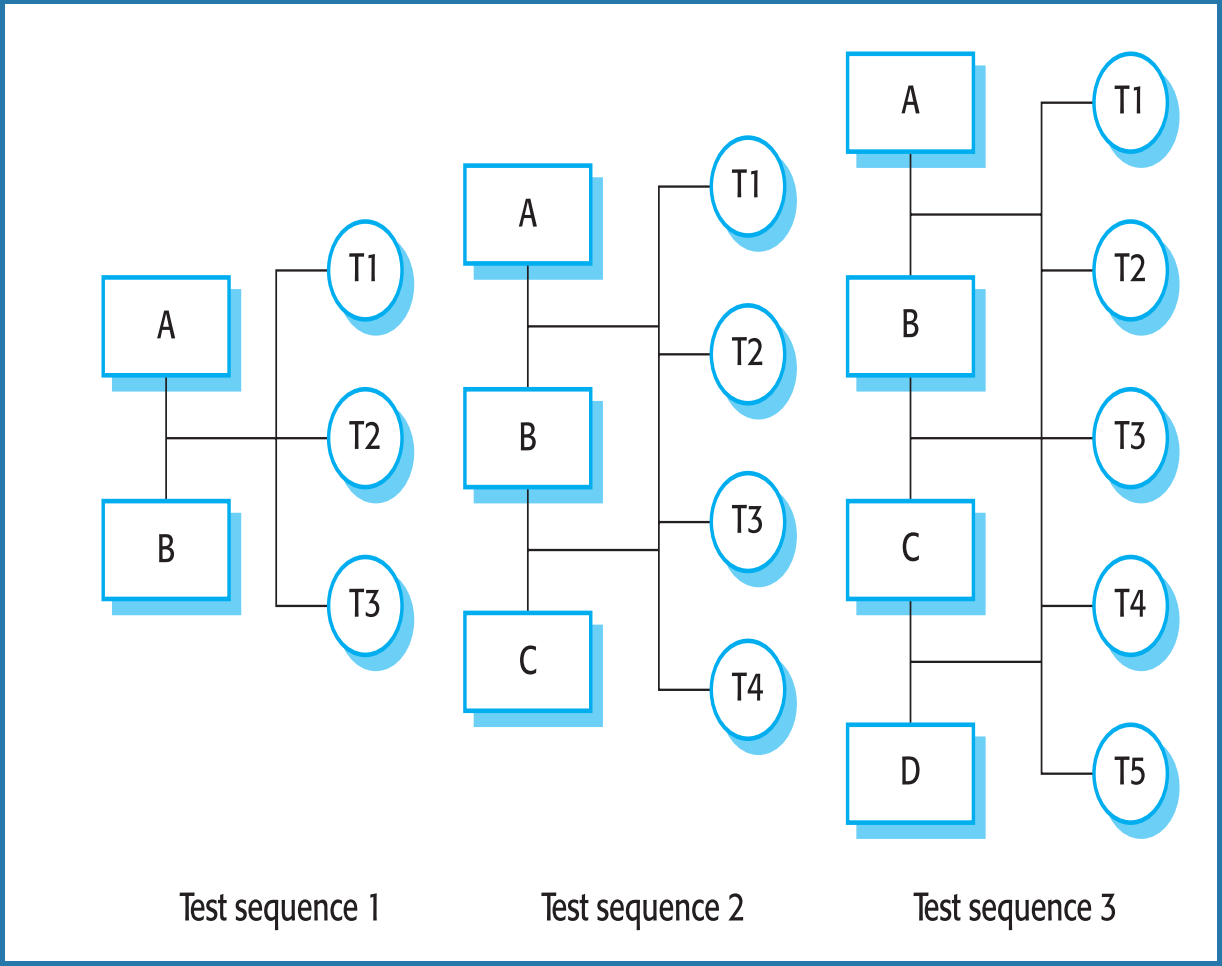
Prueba del sistema

- Involucra la integración de componentes para crear sistemas o sub-sistemas
- Se definen dos fases
  - Pruebas de Integración: el equipo de testing tiene acceso a todo el código fuente del sistema. El sistema es testeado en función de la integración de los componentes
  - Pruebas de Entrega: el equipo de testing prueba el sistema completo para ser entregado como una caja negra

Pruebas de Integración

- Involucra la creación del sistema desde sus componentes y se hacen pruebas para identificar problemas que puedan surgir desde la interacción de los componentes
- Integración top-down
  - Desarrollo del esqueleto del sistema y se agregan componentes
- Integración bottom-up
  - Integra la infraestructura de componentes y luego se le agregan funcionalidades
- Para simplificar las pruebas, el sistema debe ser integrado incrementalmente

Pruebas de Integración incremental



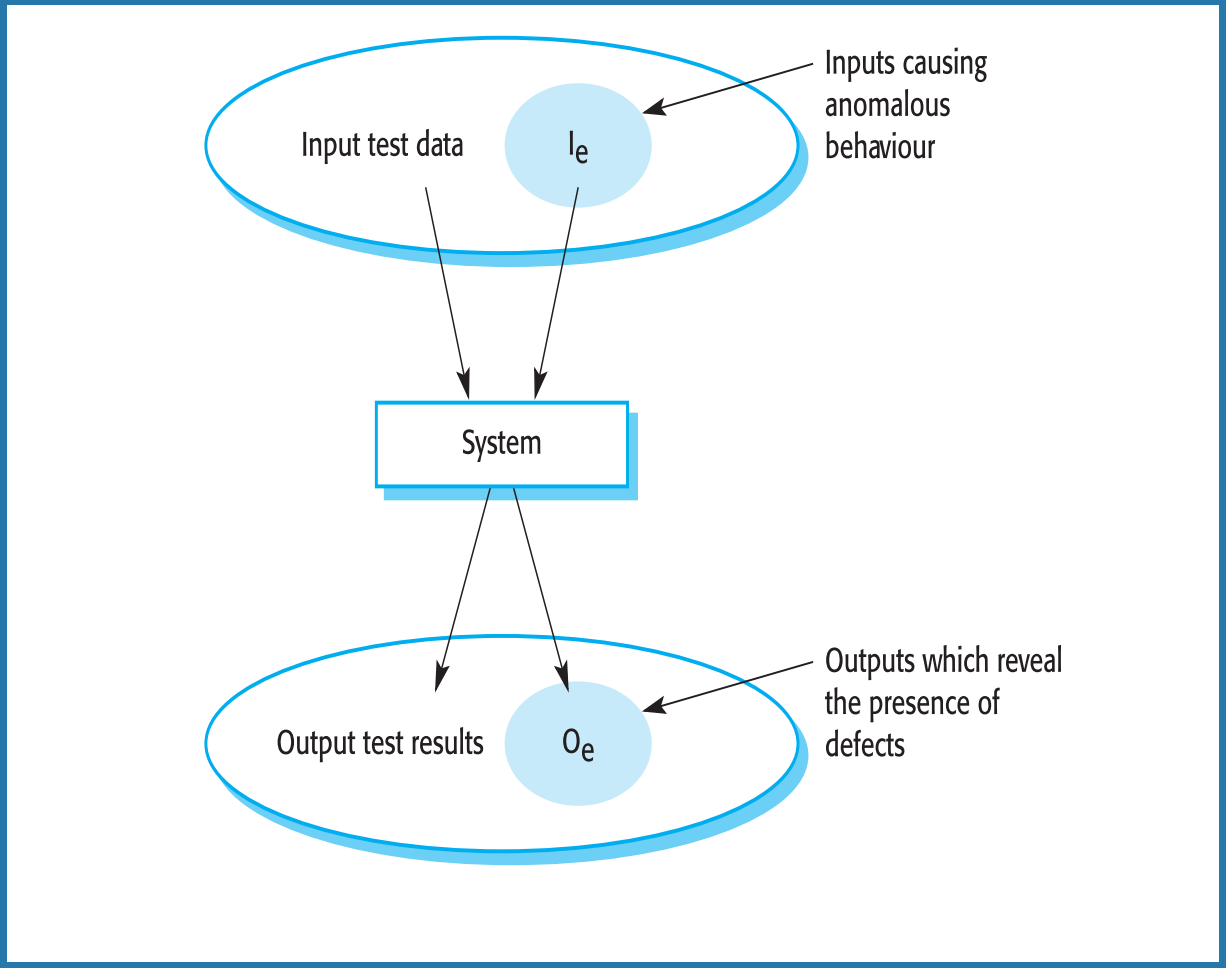
Enfoques de las Pruebas

- Validación de la arquitectura
  - La integración top-down es mejor para descubrir errores en una arquitectura de software
- Demostración del sistema
  - La integración top-down permite una demostración temprana en los inicios de la fase de desarrollo
- Test de implementación
  - La integración bottom-up es útil

Entregas de Pruebas

- El proceso de entregas de Pruebas de un sistema debe ser distribuido a los clientes
- El principal objetivo de lo anterior es entregar confianza al cliente de que el sistema cumple los requisitos establecidos
- Las entregas de Pruebas son usualmente cajas negras o funcionalidades
  - Basadas en la especificación del sistema
  - Los testers no saben lo que contiene la implementación del sistema

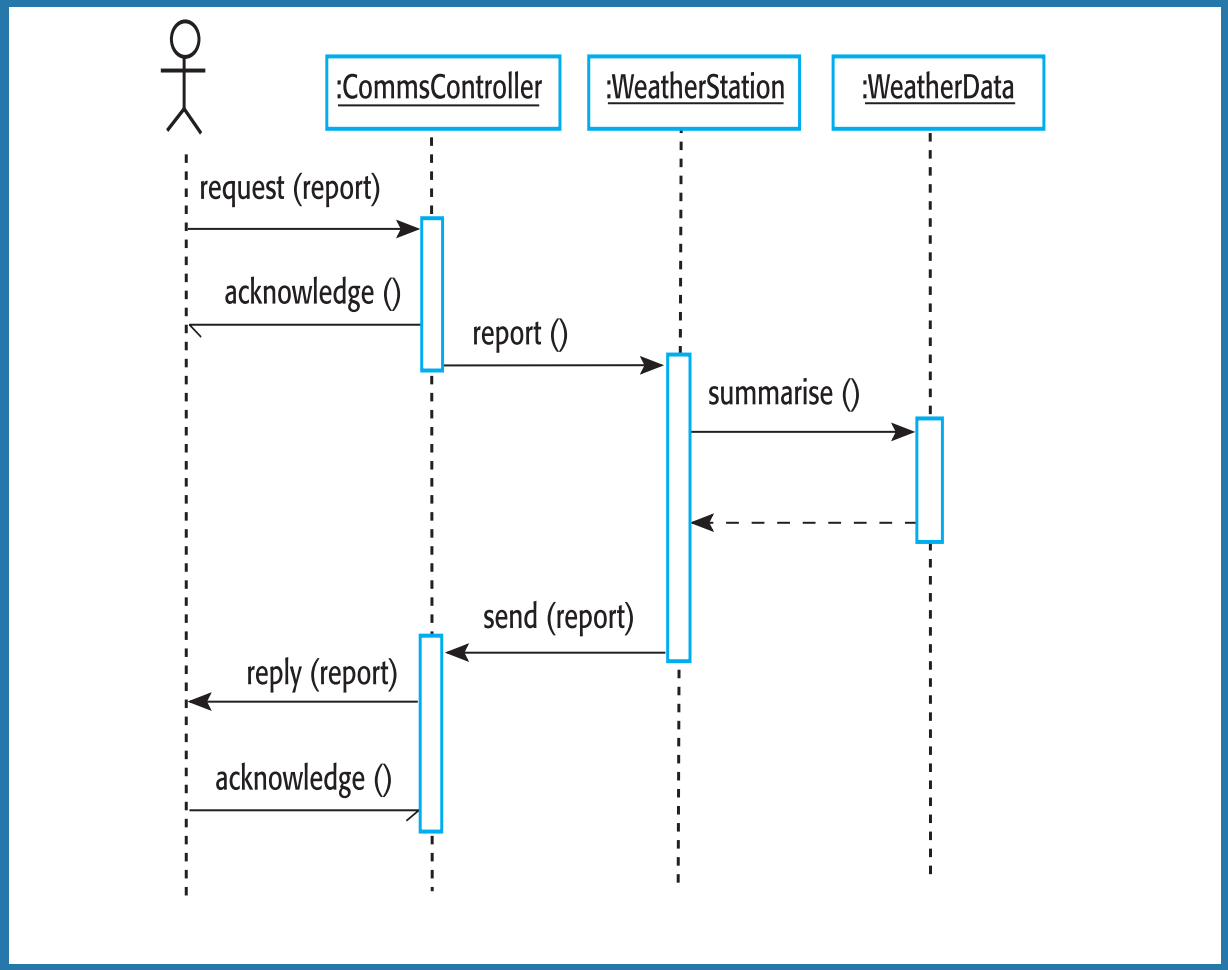
Black-box Testing



Casos de Uso [1]

- Los Casos de Uso pueden ser la base para derivar pruebas en el sistema de software
- Éstos ayudan a identificar operaciones que pueden ser testeadas y ayudan al diseño de los casos de prueba
- Además, con la ayuda de los diagramas de secuencias, se pueden realizar entradas y salidas para realizar pruebas de testing

Casos de Uso [2]



Pruebas de Regresión [1]

- Las Pruebas de Regresión se definen como un tipo de prueba de software para confirmar que un cambio reciente de programa o código no ha afectado negativamente las características existentes
- La Prueba de Regresión no es más que una selección completa o parcial de casos de prueba ya ejecutados que se vuelven a ejecutar para asegurar que las funcionalidades existentes del software funcionen bien.
- Esta prueba se realiza para asegurarse de que los nuevos cambios en el código no tengan efectos secundarios en las funcionalidades existentes. Se asegura de que el código antiguo sigua funcionando una vez que los nuevos cambios de código se realicen.

Pruebas de Regresión [2]

- Algunas herramientas que ayudan a realizar Pruebas de Regresión:
  - Selenium (http://www.seleniumhq.org)
  - Quick Test Proffesional (QTP) (https://www.tutorialspoint.com/qtp/)
  - Rational Functional Tester (RFT)->Herramienta de IBM basada en Java

Guías para Testing [1]

- Las guías que se describirán a continuación son propuesta por Sommerville [Sommerville, 2004] las cuales describen consejos para elegir la mejor prueba para encontrar defectos
  - Elegir las entradas que fuercen al sistema a generar un error
  - Diseñar entradas que cause saturación del buffer
  - Repetir los puntos anteriores varias veces
  - Forzar salidas inválidas del sistema
  - Probar dichas salidas con grandes y pequeños volúmenes de datos

Guías para Testing [2]

- QA Framework: Test Guidelines (http://www.w3.org/TR/2004/WD-qaframe-test-20040225/)
- Según Software Testing Mentor [http://www.softwaretestingmentor.com/articles/guidelines-for-software-testing/, 2016], algunas guías prácticas para Testing son:
  - La Prueba debe descubrir defectos y mejorar la calidad del software
  - La Prueba debe ser realizado a través del Ciclo de Vida del Software
  - La Prueba se debe hacer mediante la caja negra y blanca
    - White-box Testing consiste en pruebas estructurales del sistema de software, las cuales prueban la lógica interna del sistema
  - La Prueba debe ser ejecutado bajo la lógica del negocio
  - La Prueba debe ser ejecutado eficientemente con el objetivo de reducir el riesgo

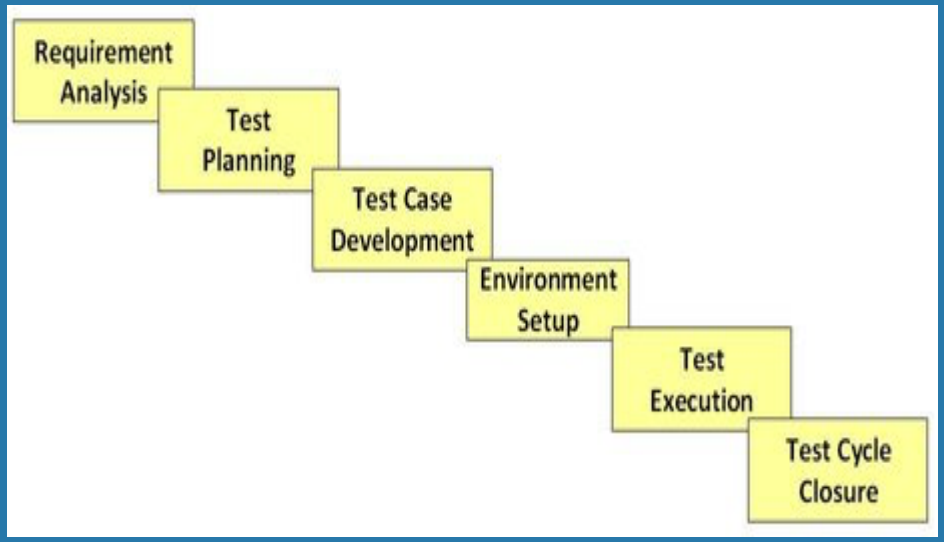
El costo de la calidad

Development Artifact	Preparation Time	Meeting Time
Requirement Document	25 pages per hour	12 pages per hour
Functional specification	45 pages per hour	15 pager per hour
Logic specification	50 pages per hour	20 pages per hour
Source code	150 lines of code per hour	75 lines of code per hour
User documents	35 pages per hour	20 pages per hour

“Software Testing Life Cycle” (STLC)

STLC

- Hasta el momento, se ha visto que Testing es una serie de actividades
- Pero, son actividades metodológicas



Análisis de Requisitos

- Durante esta fase se estudian los requisitos desde el punto de vista del testing con el objetivo de identificar requisitos testeables.
- **Actividades**
  - Identificar tipos de prueba
  - Recopilar datos sobre el establecimiento de prioridades y el enfoque
  - Identificar ambientes de prueba en donde se detalle qué se va a probar
- **Entregables**
  - Requirement Tracebility Matrix (RTM, http://www.guru99.com/traceability-matrix.html)
- **Necesidad**
  - Curiosidad

Plan de Prueba

- En esta fase, se discute el plan estratégico de las pruebas
- Típicamente, en esta etapa el analista QA determina el esfuerzo y costos estimados del proyecto y cómo se prepara y termina el plan
- Actividades
  - Preparación del plan
  - Selección de herramienta
  - Test de esfuerzo y estimación
  - La planificación de recursos y la determinación de funciones y responsabilidades
  - Entrenamiento
- Entregables
  - Plan de pruebas (http://www.guru99.com/what-everybody-ought-to-know-about-test-planing.html)
  - Estimación del esfuerzo (http://www.guru99.com/an-expert-view-on-test-estimation.html)
- Necesidad
  - Visión holística

Desarrollo de Casos de Prueba

- Esta fase involucra la creación, verificación y trabajo de los casos de prueba
- Se identifican los datos de prueba
- Actividades
  - Creación de casos de prueba
  - Revisar las bases de los casos de prueba
  - Creación de datos de prueba (si es necesario)
- Entregables
  - Casos/scripts de prueba
  - Datos de prueba
- Necesidad
  - Creatividad

Configuración del ambiente de Prueba

- La configuración del ambiente de prueba decide las condiciones del hadware o software donde el sistema será probado
- Actividades
  - Entender la arquitectura, configuración del ambiente y preparar hadware y software por cada requerimiento que será probado
  - Realizar pequeñas pruebas del sistema (para ver si todo está bien con el ambiente de prueba)
- Entregables
  - Ambiente listo con los datos de prueba establecidos
  - Resultados de las pequeñas pruebas
- Necesidad
  - Conocer la tolerancia del sistema de software

Ejecución de la prueba

- Durante esta fase, el equipo se preocupa del Plan de Prueba y los casos de prueba
- Cualquier detalle encontrado en las pruebas, se debe reportar al equipo de testing
- Actividades
  - Ejecutar pruebas acordes al plan
  - Documentar los resultados
  - Mapear defectos a casos de pruebas establecidos en RTM
  - Seguir los defectos
- Entregables
  - RTM completa
  - Casos de pruebas actualizados
  - Reporte de defectos
- Necesidad
  - Paciencia

Cierre del ciclo

- El equipo de testing conoce, discute y analiza los artefactos del testing con el objetivo de identificar estrategias que se pueden implementar a futuro
- Actividades
  - Evaluar el ciclo completo basado en atributos como: tiempo, costo, software, objetivos del negocio, otros
  - Preparar métricas de prueba
  - Documentar lo aprendido en el proyecto
- Entregables
  - Reporte de cierre de pruebas
  - Métricas de prueba
- Necesidad
  - Diplomacia

Criterios de completación de testing

- ¿Cómo saber cuándo dejar de hacer testing?
- Normalmente, se abandona al agotarse los recursos
- En la práctica el testing nunca acaba, solo cambia de tester ... del profesional al usuario
- Idealmente, utilizar modelos de fallas de software como función del tiempo de ejecución, basados en modelación estadística y teorías de confiabilidad para poder hacer afirmaciones como la siguiente:
  - ... hemos hecho suficiente testing para afirmar con un 95% de confianza que la probabilidad de 1000 horas de CPU de operación libre de fallas en un ambiente probabilísticamente definido es por lo menos 0.995 ....

Lectura recomendada

- Software Testing Tutorial
  - http://www.guru99.com/software-testing.html
- Herramientas de Pruebas de Software
  - http://www.softwaretestinghelp.com/15-best-test-management-tools-for-software-testers/

UNIVERSIDAD TECNICA
FEDERICO SANTA MARIA

Departamento de Informática
Universidad Técnica Federico Santa María

Pruebas de Software

Ingeniería de Software

Hernán Astudillo & Gastón Márquez
Departamento de Informática
Universidad Técnica Federico Santa María