

# Control de Calidad de Software

Ing. Jorge Montaña Párraga

A decorative graphic consisting of several horizontal lines of varying lengths and colors (teal, white, and light blue) extending from the right side of the slide towards the center.

# Agenda

*Fundamentos de Control de Calidad*

*Herramientas de apoyo a SQA*

*Técnicas de Diseño de Test*

*Administración de Test*

*Tipos de Test*

*Automatización en el Control de Calidad del Software*



# FUNDAMENTOS DE CONTROL DE CALIDAD

# Contenido

- Porque es necesario controlar la calidad?
- Que es testear?
- 7 Principios de Control de Calidad
- Proceso Fundamental de SQA



# Porque es necesario controlar la calidad?

## Transbordador espacial Challenger

- Falla de un sistema de emergencia de aborto de despegue cuando ocurren descompresiones o anomalías.

## Error del milenio Y2K (2000)

- Programadores adoptaron la convención de representar el año con dos dígitos.

## Sobregiro del Bank of New York (1985)

- En noviembre de 1985, tuvo accidentalmente un sobregiro de 32000 millones de dólares. Esto fue causado por un contador de 16 bits que se activo provocando un overflow que nunca fue verificado.

## Fracaso de Mariner I ( 1962)

- Fracaso por un carácter incorrecto (” ~ “) en la especificación del programa de control para el cohete de propulsión, lo cuál causó que finalmente se saliera de curso.

# Que es testear?

Tester o Testing tiene los siguientes objetivos:

- Encontrar defectos
- Adquirir confianza acerca del nivel de calidad
- Proveer información para la toma de decisiones
- Prevenir errores

Hay diferentes puntos de vista según el área. Ej.

- En el desarrollo de pruebas (pruebas de integración, componentes) objetivo principal es causar muchos errores con el fin de identificar defectos y arreglarlos.
- En pruebas de aceptación el objetivo es confirmar que el sistema trabaja como es esperado con el fin de adquirir confianza de que cumple los requerimientos

# 7 Principios de Control de Calidad

- Hacer pruebas muestra la presencia de defectos
- Hacer pruebas exhaustivas es imposible
- Hacer pruebas anticipadamente
- Agrupación de errores
- Paradoja del pesticida
- Hacer pruebas depende del contexto
- Falacia de ausencia de errores

# Proceso Fundamental de SQA

La parte visible de hacer pruebas es su ejecución. Para ser efectivo y eficiente, los planes de pruebas deben incluir tiempo estimado en planificación de pruebas, diseño de pruebas, preparación de la ejecución y evaluación de resultados.

## Proceso Fundamental de SQA (cont.)

Incluye las siguientes etapas:

- Planificación y Control
- Análisis y Diseño
- Implementación y ejecución de pruebas
- Evaluación del criterio de salida y Reportes
- Pruebas finales

# Planificación y control de pruebas

Planificación de pruebas es la actividad de definir los objetivos de pruebas y la especificación de actividades de pruebas

Control de pruebas es la actividad de comparar el progreso actual dentro lo planificado, y reportar estados,

La planificación de pruebas toma en cuenta la retroalimentación del monitoreo y control de actividades.

# Análisis y diseño de pruebas

El análisis y diseño de pruebas incluye las siguiente tareas:

- Revisar la base de pruebas (requerimientos, nivel de integridad del software, arquitectura, diseño, especificaciones de interfaces)
- Habilidad de realizar pruebas sobre la base de pruebas y objetos en prueba
- Identificar y priorizar condiciones de prueba basado en análisis de elementos de prueba, la especificación, comportamiento y estructura del software.
- Diseño y priorización de casos de prueba de alto nivel.

# Implementación y ejecución de pruebas

La implementación y ejecución incluye las siguientes tareas:

- Finalizar, implementar y priorizar casos de prueba (incluyendo la identificación de datos de prueba)
- Desarrollar y priorizar procedimientos de prueba, opcionalmente escribir scripts de casos de prueba automatizados
- Crear suites de casos prueba para su ejecución
- Verificar que el ambiente de pruebas ha sido correctamente configurado
- Guardar el resultado de ejecución de pruebas y guardar las versiones de software.
- Comparar resultados actuales con resultados esperados

# Evaluar el criterio de salida y reportes

Evaluar el criterio de salida es la actividad donde se compara la ejecución de pruebas contra la definición de objetivos. Esto debe realizarse en cada nivel de pruebas

- Incluye tareas como:
  - Verificar los logs de pruebas contra los criterios especificados en el plan de pruebas
  - Escribir resúmenes de pruebas para clientes

# Cierre de actividades de pruebas

Generalmente se realizan en hitos del proyecto como cuando un software es liberado, un proyecto de prueba es completado o cancelado, o un hito ha sido alcanzado.

- Incluye las siguientes tareas:
  - Verificar que entregables planificados han sido liberados
  - Cierre de reportes de incidentes
  - Documentar la aceptación del sistema
  - Analizar lecciones aprendidas para determinar cambios necesarios en futuras versiones



# TESTEO A TRAVES DEL CICLO DE VIDA DEL SOFTWARE

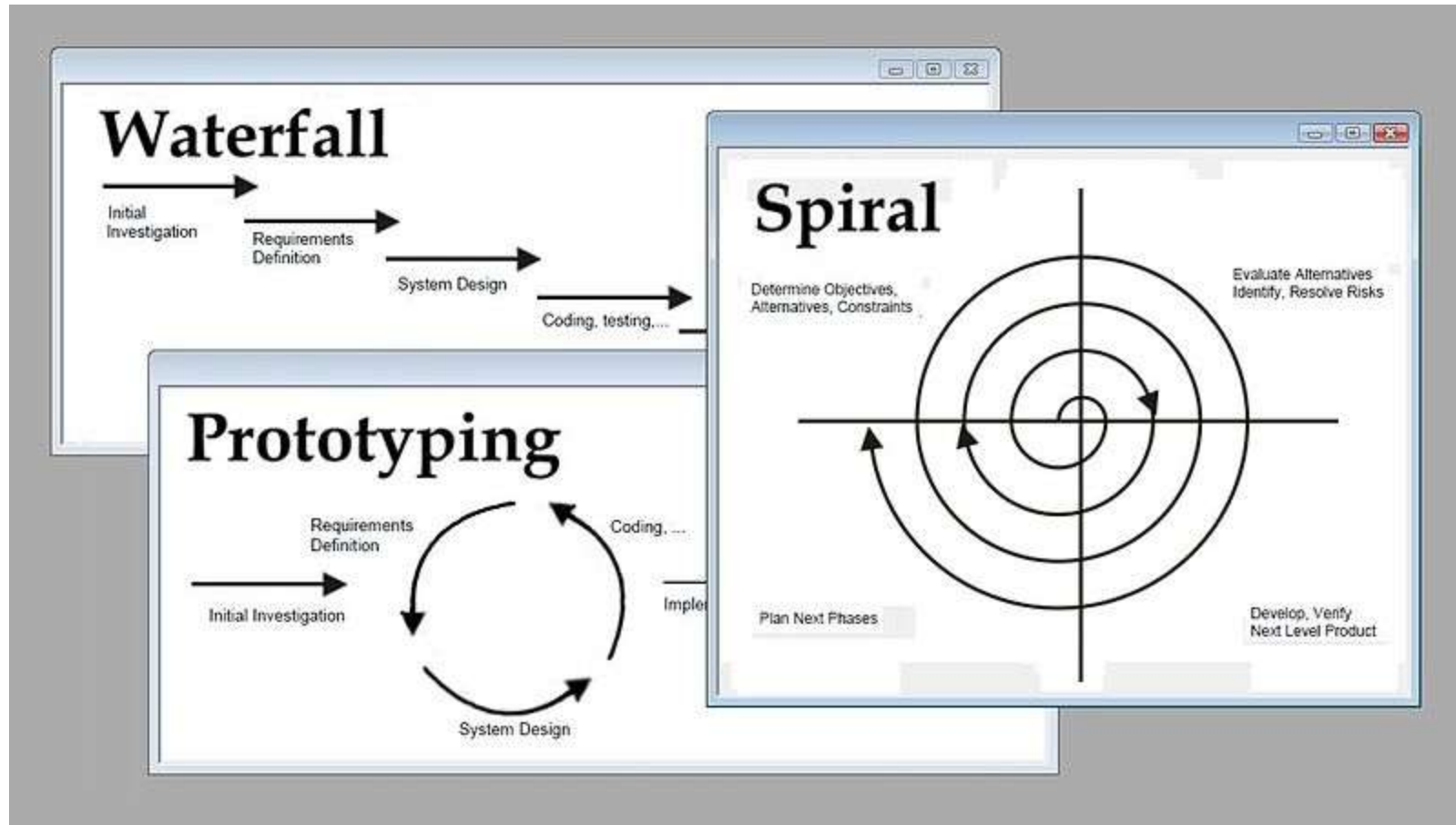
# Contenido

- Modelos de desarrollo de software
- Niveles de test
- Tipos de test
- Pruebas de mantenimiento

Testeo a través del ciclo de vida del software

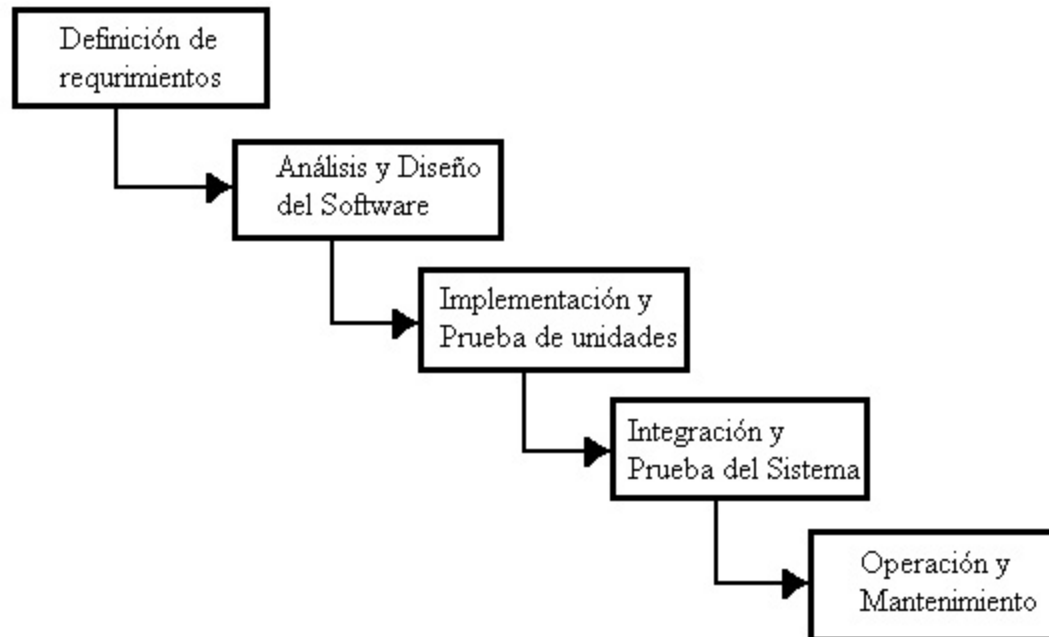
# **MODELOS DE DESARROLLO DE SOFTWARE**

# Modelos de desarrollo de software

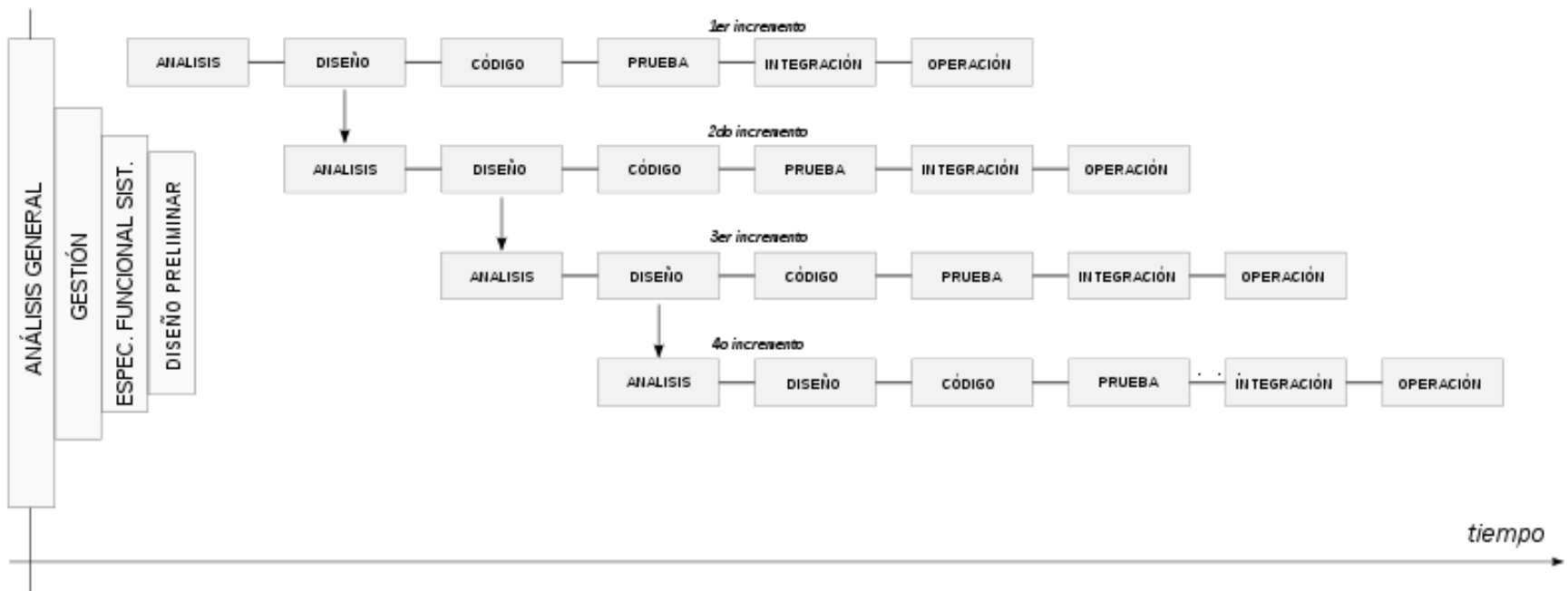


# Modelos de desarrollo de software (cont.)

- Modelo en cascada



# Modelo Iterativo Incremental



# Testeo dentro el ciclo de vida del software

Cada actividad de desarrollo debe tener una actividad de prueba

Cada prueba debe tener un objetivo específico en cada nivel

El análisis y diseño de una prueba debe empezar junto con la actividad de desarrollo

Testers deben involucrarse en la revisión de documentos tan pronto como se creen borradores dentro del ciclo de vida del software

Testeo a través del ciclo de vida del software

## **NIVELES DE TEST**

# Niveles de test

Para cada nivel de test, se debe identificar lo siguiente:

- Objetivos genéricos
- Que va a ser testeado
- Defectos y problemas a ser encontrados
- Enfoque específico y responsabilidades

# Pruebas de componentes

También conocido como (unit test) pruebas unitarias, pruebas de modulo o de programa.

## Base del test

- Pruebas de componentes
- Diseño detallado
- Código

## Objetos a ser testeados

- Componentes
- Programas
- Conversión de datos

# Pruebas de Integración

## Base del test

- Diseño de software
- Arquitectura
- Workflows
- Casos de uso

## Objetos a ser testeados

- Sub-sistemas de implementación de base de datos
- Infraestructura
- Interfaces

# Pruebas de Sistema

## Base del test

- Especificación de requerimientos de software
- Casos de uso
- Especificaciones funcionales
- Reportes de análisis de riesgos

## Objetos a ser testeados

- Sistema, usuarios y manuales de operación
- Configuración del sistema

# Pruebas de aceptación

## Base del test

- Requerimientos de usuario
- Requerimientos del sistema
- Casos de uso
- Proceso del negocio
- Reportes de análisis de riesgos

## Objetos a ser testeados

- Procesos de negocio en sistemas integrados
- Procesos operacionales
- Procedimientos de usuario
- Formularios
- Reportes

# Niveles de test

## Pruebas de aceptación de usuario

- Verifica que el sistema pueda ser usado correctamente por usuarios

## Pruebas de aceptación operacionales

- La aceptación de sistema por administradores del sistema, incluyendo
  - Pruebas de copias de respaldo/recuperación
  - Recuperación de desastres
  - Administración de usuarios
  - Tareas de mantenimiento
  - Tareas de migración y carga de datos
  - Verificación periódica de vulnerabilidades de seguridad

# Niveles de test (cont.)

Analistas de mercado, a menudo quieren retroalimentación de clientes potenciales en su mercado antes de que el producto salga a la venta

## Pruebas alpha y beta

- Pruebas alpha son realizadas en el lugar donde se desarrolla el producto pero no por el equipo de desarrollo
- Pruebas beta o pruebas de campo son realizadas por clientes o potenciales clientes en sus ubicaciones propias.

Testeo a través del ciclo de vida del software

## **TIPOS DE TEST**

# Tipos de test

Un tipo de test puede ser enfocado a un objetivo de test particular, que puede ser cualquiera de los siguientes

- Una función a ser realizada por el software
- Una característica no funcional de calidad, como la usabilidad o confiabilidad
- La estructura o arquitectura del sistema

# Prueba Funcional

Las funcionalidades de un sistema, subsistema o componente debe ser descritas como una especificación de requerimientos, casos de uso, o especificaciones funcionales. Las funcionalidades son lo que «hace» el sistema.

Las pruebas funcionales están basadas en funciones y características

# Pruebas No Funcionales

Pruebas no funcionales incluyen pero no esta limitado a:

Pruebas de  
performance

Pruebas de  
carga

Pruebas de  
stress

Pruebas de  
usabilidad

Pruebas de  
portabilidad

Es la prueba de «como» funciona el sistema

Pruebas no funcionales pueden ser realizadas en todos los niveles de testeo.

# Pruebas Estructurales (Caja blanca)

Pruebas de estructura (caja blanca) pueden ser realizadas en todos los niveles de testeo.

Coverage o cobertura es el grado que una estructura ha sido ejercitada por un suite de pruebas (test suite), expresado como el porcentaje de elementos cubiertos

# Pruebas de Regresión

Test de regresión es la repetición de testeos en un programa que ya ha sido testeado, después de una modificación, a fin de descubrir cualquier defecto introducido o no cubierto por los cambios realizados

Las pruebas deben ser repetibles, deben especificar bien sus pasos.

Las pruebas de regresión pueden ser realizadas en todos los niveles de testeos, incluye funcional, no funciona y pruebas de caja blanca.

Las pruebas de regresión son ejecutadas muchas veces y generalmente evolucionan lentamente

# Testeo de mantenimiento

Una vez que el sistema ha sido liberado, generalmente esta en servicio por años o décadas. Durante este tiempo su configuración, su ambiente es corregido, cambiado o extendido.

La planificación de versiones anticipadamente es crucial para las pruebas de mantenimiento.

Se debe distinguir entre versiones planificadas y hot fixes. Las pruebas de mantenimiento se realizan en sistemas operacionales, y son ocasionadas por las modificaciones, migraciones o retiro del software.



# PROCESO DE DESARROLLO DE PRUEBAS

# Contenido

- Técnicas de diseño de pruebas
- Categorías de técnicas de diseño de casos de prueba
- Técnicas de caja negra
- Técnicas de caja blanca
- Técnicas basadas en la experiencia
- Como elegir una técnica de prueba

# Técnicas de diseño de casos de prueba

Un caso de prueba consiste en un conjunto de valores de entrada, prerequisites de ejecución, resultados esperados, y condiciones de post-ejecución desarrollados para cubrir determinados objetivos de prueba

El estándar para documentación de pruebas de software (IEEE STD 829-1998) describe el contenido de especificación de diseño de pruebas

# Categorías de técnicas de diseño de casos de prueba

El propósito de una técnica de diseño es identificar condiciones de prueba, casos de prueba y datos de prueba

Hay una distinción clásica que denota las técnicas de prueba como pruebas de caja blanca o caja negra

# Técnicas de caja negra

Por definición no usan información relacionada a la estructura interna del componente o sistema

# Técnicas de caja blanca

Pruebas de caja blanca también llamadas técnicas estructurales o basadas en estructuras, están basadas en el análisis de la estructura de un componente o sistema

# Técnicas basadas en la experiencia

Pruebas basadas en la experiencia se refiere a la intuición, habilidad del tester y su experiencia con aplicaciones y tecnologías similares

Una técnica basada en la experiencia comúnmente usada es adivinar donde está el error. Generalmente el tester anticipa defectos basado en su experiencia. Una técnica es listar los posibles defectos y diseñar pruebas para atacar esos defectos.

# Como elegir una técnica de prueba

Elegir que técnicas usar depende de varios factores, como ser: tipo de sistema, estándares, requerimientos del cliente, niveles de riesgo, objetivos de pruebas, documentación disponible, etc.

Algunas técnicas son mas aplicables a determinadas situaciones y niveles de pruebas, otras son aplicables a todos los niveles.



# ADMINISTRACION DE TESTS

# Contenido

- Organización de pruebas
- Plan de testeo, estimados
- Control y monitoreo de progreso de test
- Administración de incidentes

# Organización de pruebas

La efectividad de encontrar defectos haciendo SQA puede ser mejorada teniendo testers independientes.

Para proyectos grandes, críticos y/o complejos es mejor tener múltiples niveles de pruebas, con algunos o todos los niveles realizados por testers independientes.

Equipo de desarrollo puede participar en las pruebas a bajo nivel, pero su falta de objetividad usualmente limita su efectividad.

Los testers independientes deben tener la autoridad requerida para definir procesos de pruebas y reglas.

# Organización de pruebas (cont.)

## Beneficios de independencia son:

- Testers independientes ven otros y diferentes defectos y son imparciales
- Un tester independiente puede verificar supuestos hechos durante la especificación e implementación del sistema

## Desventajas:

- Isolacion del equipo de desarrollo (si son totalmente independientes)
- Desarrolladores pueden perder el sentido de responsabilidad por la calidad
- Testers independientes pueden verse como cuellos de botella o ser culpados por retrasos en la liberación del producto.

# Plan de testeo y estimados

Planificación de pruebas puede ser documentada en un plan de pruebas maestro o en planes separados para los niveles de pruebas como ser pruebas de sistema y pruebas de aceptación (IEEE Std 829-1998)

Planificación de pruebas se ven influenciadas por las políticas de pruebas de la organización, el alcance de pruebas, objetivos, riesgos, restricciones.

Las estimaciones pueden tener dos enfoques:

- Basados en métricas
- Basados en experto

Una vez que el esfuerzo es estimado, recursos pueden ser identificados y programados.

# Control y monitoreo de progreso de test

El objetivo del monitoreo es prever retroalimentación y visibilidad de las actividades de pruebas. Esta información puede ser colectada manualmente o automáticamente y puede ser usada para verificar el criterio de salida y cobertura.

# Administración de incidentes

Como uno de los objetivos de realizar pruebas es encontrar defectos, las discrepancias entre el resultado actual y el esperado debe ser documentado como incidentes o defectos.

Los incidentes tienen el siguiente objetivo:

- Proveer a los desarrolladores y otras personas involucradas retroalimentación sobre el problema, como isolar y corregir si es necesario.
- Proveer ideas sobre como mejorar el proceso de pruebas

# Detalles de un incidente

## Detalles de un incidente puede incluir:

- Fecha del error, autor
- Resultados actuales y esperados
- Ambiente
- Descripción del incidente para facilitar su reproducción y resolución, incluyendo logs, capturas de pantalla
- Severidad del impacto en el sistema
- Prioridad para ser arreglado
- Estado del incidente (abierto, duplicado, arreglado, en espera de ser arreglado, cerrado)
- Conclusiones, recomendaciones y aprobaciones.

Para mas información IEE Std 829-1998



# **HERRAMIENTAS DE APOYO AL CONTROL DE CALIDAD DE SOFTWARE**

# Contenido

- Propósito de las herramientas de apoyo de SQA
- Tipos de herramientas de apoyo al Control de Calidad

# Propósito de las herramientas de apoyo de SQA

Las herramientas pueden ser usadas para una o mas actividades relacionadas a SQA, incluye

- Herramientas usadas directamente en pruebas como ser: Administración de pruebas, resultados de pruebas, datos, requerimientos, incidentes, etc., también reportes y monitores de ejecución de pruebas.
- Herramientas que son usadas en reconocimiento (exploración), Ej. Herramientas que monitorean actividades de ficheros de una aplicación.
- Cualquier herramienta que ayude a hacer pruebas(hoja de calculo)

## Propósito de las herramientas de apoyo de SQA (cont.)

El termino test frameworks es usado frecuentemente en la industria y tiene por lo menos dos significados:

- Librerías de pruebas reusable y extensibles que pueden ser usadas para construir herramientas de pruebas.
- Un tipo de diseño de pruebas automáticas (ej. Dirigida por datos, dirigidas por palabras claves)

# Tipos de herramientas de apoyo al Control de Calidad

Herramientas de apoyo a la especificación de pruebas

Herramientas de apoyo a la ejecución de pruebas y creación de bitácoras

Herramientas de apoyo al performance y monitoreo



# **AUTOMATIZACION EN EL CONTROL DE CALIDAD DEL SOFTWARE**

# Contenido

- Herramientas de automatización de software
- DEMO

# Herramientas de automatización de software

SilkTest

AutoIT

TestComplete

Selenium

Coded UI  
Test

# SilkTest

**SilkTest** es una herramienta de automatización para realizar pruebas de funcionalidad en aplicaciones empresariales

Es propiedad de Segue Software que fue adquirido por Borland en el año 2006.

Borland fue adquirida por Micro Focus International en el 2009.

## SilkTest (cont.)

### SilkTest tiene varios clientes:

- SilkTest Classic usa el lenguaje 4Test para realizar script automatizados. 4Test es un lenguaje orientado a objetos similar a C++.
- Silk4J permite realizar automatización en Eclipse usando Java como lenguaje de programación
- Silk4Net permite lo mismo en Visual Studio usando VB or C#.



Silk4NET

**DEMO**

# Donde continuar?

- [http://en.wikipedia.org/wiki/IEEE\\_829](http://en.wikipedia.org/wiki/IEEE_829)
- <http://standards.ieee.org/findstds/standard/829-1998.html>
- <http://www.sqa.org/>
- <http://istqb.org/display/ISTQB/Home>

¿Dudas o comentarios?