LA FIABILIDAD EN LOS SISTEMAS DE TELECOMUNICACIONES

Antonio Moya Catena
Responsable de Calidad y Desarrollo Operativo
Centro I+D, Ericsson España, S.A.

ERICSSON

Global presence and customer relationships

- A unique position: 130 years – 140 countries
- Never left a market – never left a customer
- Innovation for customer growth and profitability

Dedicated to customer success

Definición de Confiabilidad

Confiabilidad.

- 1. f. Cualidad de confiable.
- 2. f. **fiabilidad** (probabilidad de buen funcionamiento de algo).

Fiabilidad de sistemas

- Extendiendo el significado de fiabilidad a sistemas, para fijar el contexto de esta ponencia, se puede decir que:

  "la fiabilidad de un sistema es la probabilidad de que ese sistema funcione o desarrolle una cierta función, bajo condiciones fijadas y durante un período de tiempo determinado" (definición según IEEE).

Conceptos relacionados con fiabilidad

Disponibilidad (Availability, uptime)

- Es una medida del tiempo durante el cual un sistema esta disponible y operativo ininterrumpidamente. Normalmente se usa como una medida de la fiabilidad y estabilidad de un sistema, y podría representar el tiempo durante el que un sistema se podría dejar desatendido sin que caiga/falle.
- Se suele hablar de disponibilidad en termino de “nueves”, siendo “cinco nueves” el grado de alta disponibilidad:

	Tiempo total de caida (Dowtime)(HH:MM:SS)		
Availability %	**por día**	**por mes**	**por año**
99%	00:14:23	07:18:17	87:39:29
99.9%	00:01:26	00:43:49	08:45:56
99.99%	00:00:08	00:04:22	00:52:35
99.999%	00:00:00.4	00:00:26	00:05:15

Tiempo de caída del sistema (Downtime planned/unplanned)

- El tiempo de caída (downtime o outage) se refiere al periodo de tiempo, o porcentaje del tiempo, durante el cual un sistema está no disponible o fuera de servicio. Es el tiempo durante el cual el sistema esta indisponible/falla debido a eventos no planificados (fallos) o planificados (rutinas de mantenimiento).
- Se aplica tanto a redes como a servidores.

Conceptos relacionados con fiabilidad

Carrier grade

- En telecomunicaciones, se dice que un sistema es “carrier grade” o “carrier class” cuando es extremadamente fiable, bien testeado y cuyas características han sido totalmente verificadas.

- Para que un sistema adquiera la categoría de “carrier grade” tiene que haber sido diseñado y testeado para cumplir las exigencias de los “cinco nueves” o de alta fiabilidad, además de proporcionar mecanismos de restablecimiento (recovery) muy rápidos en caso de fallos totales.

Restablecimiento (Recovery)

- El tiempo de restablecimiento esta íntimamente relacionado con la disponibilidad de un sistema, ya que es el tiempo total que requiere el sistema para restablecer las condiciones de disponibilidad después de una caída total planificada o no planificada.
- En ciertos tipos de eventos como fuego o inundación, este tiempo puede ser infinito si no hay unos mecanismos apropiados de restablecimiento del tipo ”business continuity”.


¿Cómo conseguir una fiabilidad
99,999%
en un entorno tan complejo?
WWW.CATAPULT.COM

Diseño de sistemas para alta fiabilidad

Los sistemas complejos, como la red y sistemas de telecomunicaciones, para conseguir alta fiabilidad (99,999%) utilizan diferentes técnicas y procesos:

Entre estas técnicas están:

- Redundancia de sistemas (en distintas localidades)
- Redundancia de componentes hardware del sistema y bases de datos
- Códigos de corrección de errores en protocolos
- Aseguramiento de la integridad de datos (
- Retransmisiones en protocolos
- Cheksum
- Actualización del sistema en caliente
- Propagación multi-trayecto de señales
- Backups
- Estándares

Procesos y procedimientos de desarrollo de sistemas basados en

- Modelos de Madurez y aseguramiento de calidad (CMMI, SPICE, ISO)
- Reglas de diseño (tanto en Análisis de sistemas, Diseño, Codificación)
- Aseguramiento de calidad basado en testing, estableciendo estrategias de verificación, como en diseño)
- Revisiones e Inspecciones
- Listas de chequeo
- Análisis de riesgo y causas de error
- Modelos de aseguramiento de Calidad y Prevención de defectos:

Entre estos modelos comentare uno: Software Quality Rank

Software Quality Ranks
SQR

Breve introducción

¿Como mejorar la calidad del SW?

- Mejorando la gestión de los requisitos de usuario
  - Construiremos el sistema correcto
- Mejorando el Análisis de Sistema
  - Crearemos una mejor arquitectura del sistema
- Mejorando el diseño
  - Especificaremos un sistema comprensible y verificable
- Mejorando la codificación
  - Implementaremos el sistema tal como fue diseñado
- Mejorando las pruebas
  - Verificaremos que el sistema funciona como debería

SQR: Principios

- Es una buena práctica para la gestión de la calidad del SW y su mejora continua
  - Mas funcionalidad, menos rehacer el trabajo (por errores/fallos)
  - Mejor calidad incluidos los aspectos no-funcionales
- Analistas de sistemas y diseñadores tiene que tomar responsabilidad sobre la calidad del SW
  - Tanto el código fuente, como las descripciones, pruebas y medidas deben contribuir a la obtención de un grado de calidad conocido
- La calidad del SW debe ser medible por el proyecto
  - Usando medidas de calidad de la organización y no de eficiencia
  - SQR debe tener impacto en la mejora de los resultados de calidad y por tanto en la reducción de tiempos de entrega y de eficiencia (menor trabajo de corrección de errores)
  - Mediación del retorno de la inversión en calidad
- La mejora de calidad debe hacerse donde mas beneficio produce
  - En una organización siempre hay diferentes proyectos, múltiples funcionalidades, distintos impactos en SW, diferentes objetivos de calidad
- SQR soporta la aplicación de diferentes métodos de desarrollo
  - Tradicionales y Agile: test driven development, re-factoring, etc.

SQR: rangos

- Rango 1:
  - Calidad básica, entregas pre-comerciales, demos
- Rango 2:
  - SW comercial no crítico
- Rango 3:
  - Telecom grade: SW no crítico
- Rango 4:
  - Telecom grade: SW crítico
- Rango 5:
  - Aplicaciones espaciales, SW de emergencias

SQR: Aspectos de calidad elegidos

- Mantenibilidad
  - Los aspectos de mantenibilidad reflejan cuanto de bien es conocido el sistema y cuanto de bien esta descrito.
- Estructura
  - Los aspectos estructurales tienen que ver con cuanto de bien ha sido compuesto el sistema.
- Funcionalidad
  - Los aspectos de funcionalidad describen que funcionalidad se espera que funcionen en un componente.
- Capacidad
  - Los aspectos de capacidad describen como se han tratado los diferentes requisitos no funcionales.

Aportación de la aplicación de SQR

- Capacidad para definir objetivos de calidad
- Capacidad para planificar las actividades de aseguramiento de calidad
- Capacidad para comparar diferentes partes de la base de diseño y nuevos componentes
- Medir y comparar resultados para apoyar la predicción de calidad
- Evaluar el retorno de la inversión
  - Análisis de Fault-Slip-Through
  - Análisis de causas raíz de fallos
  - Implementar mejoras

SQR beneficios

- Determinar el estado de cada componente por su rango
- El rango de un componente ilustra clara y precisamente su calidad
- Invertir en calidad donde es más rentable/apropiado
- El retorno de la inversión puede ser evaluado
  - Por ejemplo, un componente con rango superior debería tener menos defectos que uno de nivel mas bajo. El número de fallos que se pasan de una fase a otra (Fault Slip Through) debería ser menor también.
- La gestión de la calidad es tan importante como mejorarla, asignando el presupuesto de calidad donde se necesita

SQR: Despliegue

- Hacer una pre-evaluación
- Asegurar el compromiso de la organización
- Introducción y entrenamiento en SQR
- Organizar un proyecto SQR
- Adaptar el modelo SQR
- Establecer una línea de base de SQR
- Analizar el producto actual
- Seleccionar herramientas necesarias
- Definir como implementar y seguir SQR

	A	B	D	E	F	G	H	I	J	K
1				Component						
2		Project:		Component 1	Component 2	Component 3	Component 4	Component 5	Component 6	Component 7
3		Submitted by:								
4		WP name:								
5		Approved at BCB:								
6			Requested/Planned SQR level for each component							
7		Description and Behaviour (DB # = x.y there x = level of SQR and y = number of requirement for that level)								
8	SQR 1	DB 1.1	Existing code, design documents, test code, and test documentation are checked in and are under revision control.							
9		DB 1.2	All (new) states (i.ein the design/code) in the component are known by at least two designers.							
11	SQR 2	DB 2.1	All states (i.e state charts) in the component are documented.							
12		DB 2.2	New changes in the interfaces are avoided unless well motivated.							
13		DB 2.3	The control flow (or dataflow) is investigated and known by at least two designers.							
14		DB 2.4	Relevant Minimal and Maximal delimiters should be documented (e.g in the code header)							
16	SQR3	DB 3.1	The Component shall contain enough comments and have appropriate documentation, including design and test specification.			Not passed				
17		DB 3.2	Documented automated suite of tests.							
18		DB 3.3	A state-transition table or diagram shall be documented, to support creation of proper test cases, and make better design documentation and should be updated when changed.							
20		Dependencies and impacts (DI # = x.y there x = level of SQR and y = number of requirement for that level)								
21	SQR 1	DI 1.1	All new dependencies are known.							
23	SQR 2	DI 2.1	Relevant dependencies are documented.							
24		DI 2.2	Introduced impact on other system parts has proper fault handling, documentation and is tested.							
26	SQR3	DI 3.1	New dependencies shall be tested, documented and communicated.							
27		DI 3.2	Special effort should be made to test fault/error situations, fault handling mechanisms or special failure scenarios for dependencies that might impact others.							
29		Test and Functionality (TF # = x.y there x = level of SQR and y = number of requirement for that level)								
30	SQR 1	TF 1.1	In case of change, the functionality that worked in previous version is still working.							
31		TF 1.2	The normal cases of the functionality are working.							
32		TF 1.4	Component Test Record or equivalent documentation is written.							
33		TF 1.5	Number of functional test cases are written in the Component Test Record or equivalent document							
34										
36	SQR 2	TF 2.1	Basic error/ fault handling is working.							
37		TF 2.2	Interfaces are tested with minimal and maximal values and at least one value of each equivalent class (valid and invalid data)							
38		TF 2.3	Error checks and fault scenarios are tested.							
39		TF 2.4	All possible states are tested (been visited by a test case).							
40		TF 2.5	Failure scenarios are tested (functionally) where sequences of events are in focus (other than dependencies - see DI 3.2)							

¿Preguntas?