Aseguramiento de Calidad de Software

Carrera: Ingenieria en Software

Año: 2022

Profesores:

- **Julieta Barrionuevo**
- **Juan Manuel Ojeda**

Contenido TP3

Unidad 5

Cobertura de Testing

Introducction:

En esta catedra nos referimos a cobertura te testing a lo que tambien se conoce como test coverage. Esto es la cobertura de prueba del software para el que hacemos QA.

Esta medida es una de las medidas de la calidad que nos dice cuánto código y funcionalidades se han probado.

Muchass veces, la cobertura nos ayuda a definir entidades de sistema para asi poder cubrirlas con distinto tipos de pruebas.

¿Qué es, y que entendemos por cobertura?

La cobertura es la cantidad o porcion de código (medida porcentualmente) que se cubre actualmente por distintas pruebas .

Basicamente el calulo de covertura deviene de la ejecucion de pruebas de mi aplicación y de verificar si existe alguna línea de código que nunca fue ejecutada en el contexto de las pruebas. En este casos podemos entonces decir que “dicha línea no está cubierta.

Si en cambio mi código consta de 100 líneas y solo 25 líneas están siendo ejecutadas al correr las pruebas, entonces mi cobertura es del25% (Muy mala cobertura).

La preguntas que podemos hacer aqui son:

Topico	Descripción
Qué beneficios tiene tener una alta cobertura?	Este tema puede volverse polemico ya que entramos en el espectro de “ que tan bueno debe ser el soft” - por cuestiones de simplicidad diremos que: Tener alta cobertura, significa que gran parte el código está siendo probado y por consiguiente podría tener cierta certeza sobre el correcto funcionamiento de mi aplicación -
¿Qué porcentaje de cobertura es suficiente?	Dependiendo del tipo de proyecto y caracteristicas externas al Area podemos definir (no esta escrito en piedra) que lo ideal es de al menos 90% y en algunos proyectos podemos buscar el 80%. Una cobertura del 100% solo nos dice que todo nuestro código está siendo cubierto por pruebas, pero puede que las pruebas no esten contemplando algunas situaciones edge como las vistas en clase.
¿Qué necesito para poder medir la cobertura?	1- Escribir pruebas 2- Contar con alguna herramienta que permita medir la cobertura. muchos de loss lenguajes más populares cuentan con herramientas para medir la cobertura. ● Si trabajamos con C# y escribimos nuestras pruebas con MSTest ● Visual Studio nos ofrece la posibilidad de habilitar la medición de cobertura. ● Ruby, nos ofrece RSpec para escribir nuestras pruebas y SimpleCov para medir la cobertura. ● Java nos provee de la a herramienta code coverage.
Algunas herramientas?	Casos de Uso Diagramas O.O (Clase , ER , Secuencia) Wireframes SQL – Bases de Datos Documentación del sistema bajo prueba

	Además de las mencionadas existe una herramienta que hoy por hoy es la herramienta por excelencia utilizada por persona que se dedica a la calidad de software. Esta herramienta se llama Caso de prueba (Test Case)

Tipos de Cobertura

Utilizaremos esta web para realizar la practica y entender los distintos tipos de cobertura

- **Statement coverage**
  - La cobertura de **statements** garantiza que todas las declaraciones en el código fuente se hayan probado al menos una vez. Esta test de cobertura nos proporciona detalles de los bloques de código ejecutados y fallidos del total de bloques de código sin importar cuantos bucles, sentencias, lineas contenga - NO es barato ya que crece exponencialmente el costo a medida que crece el N° de statements.

- **Decision/Branch coverage (Branch)**
  - La cobertura de ramas se calcula encontrando el número mínimo de caminos que aseguren que todos los bordes han sido cubiertos. En el ejemplo que se meustra en la web, no hay una sola ruta que asegure la cobertura total de la totalidad de aristas en una sola vez.

- **Path Coverage**
  - Path coverage es un método de prueba estructural que implica el uso del código fuente de un programa para encontrar todas las rutas ejecutables posibles. Requirere un “pre work” para poder ser implementada ya que para encontrar este “path” es necesario tener acceso al codigo
  - La cobertura de “path” garantiza la cobertura de todas las rutas de principio a fin y reduce la repeticion de paths anteriormente testeados

- **Condition Coverage**
    - El “condition” testing comprueba si se han ejercido ambos resultados ("verdadero" o falso) de cada condición. Basicamente intenta reducir la ejecucion mediante la evaluacion de las condiciones y asegurando de que cada una de las condiciones sea evaluada en sus valores “verdadero” y “falso”
    - Este tipo de testeo requiere dos casos de prueba por condición para dos resultados.

- **Boundary Value Coverage**

  Esta métrica de cobertura es muy útil para aquellas aplicaciones en las que se produjo un error debido a los números de entrada o valores ingresados por el usuario o que fueron resultado de otro feature del mismo sistema.

  Cuando estos errores occurren en valores limites, es cuando el “boundary coverage” resulta util. En la cobertura de valores límite (boundary), los casos de prueba se seleccionan en los extremos de las clases de equivalencia.

    - Como revisamos en clases, tenemos que tener cuidado en no confundir ya que no es lo mismo **Boundary** que **Edge Case** ya que en el primero se refiere a valores LIMITE y en el 2do caso nos referimos a escenarios o casos poco probables de suceder que terminan sucediendo.

Niveles de Staging

Introduccion:

Para este tema utilizaremos esta pagina como referencia ya que provee un approach similar al utilizado en esta caterda y servira de apoyo en caso de que existan dudas o haya que validar algun conocimiento.

Que son los “Stagging environments” ?

Los Staging environemnts are simplemente **entornos de prueba** (Stages) , esto es, ambientes que soon una réplica casi exacta de un entorno de producción para pruebas de software.

Los entornos de prueba deben estar diseñados para probar codigo, funcionalidades, compilaciones y actualizaciones para garantizar la calidad en un entorno similar al de producción antes de la implementación de la aplicación en el mundo real.

El entorno de prueba requiere similaridad casi exacta de las mismas configuraciones de hardware, servidores, bases de datos y cache.

En un mundo ideal, todo en un entorno de prueba debe ser una copia lo más parecida posible al entorno de producción para garantizar que el software funcione correctamente.

Ejecutando pruebas en “staging”

Los Smoke tests y las pruebas de aceptación del usuario (UAT) se pueden realizar en un entorno de staging - es decir, un entorno de prueba.

Los Smoke tes verifican las funcionalidades esenciales del servicio (que no se haya roto nada que anteriormente andaba) y las pruebas UAT se realizan desde la perspectiva de un usuario final.

Por ejemplo, si se desarrolla un nuevo build (version), un smoke test puede rapidamente confirmar que las funciones principales aún funcionan correctamente y uqe nada de lo que funcionaba dejo de hacerlo, en cambio una prueba UAT puede y debe garantizar (en gran medida) la calidad desde la perspectiva del usuario. Las pruebas se realizan en el entorno de prueba porque bajo ningun aspecto queremos poner en riesgo los datos REALES de usuarios REALES en el entorno productivo.

OKRs - Objective and Key Results

La metodología de OKR es un framework donde se plantean desafíos más allá de los que son impuestos a nivel “medible” (metricas y KPIs) y pone un especial foco en aspectos no tan obvios a primera vista.

La idea es mantener en foco temas prioritarios mediante el uso objetivos desafiantes (aspiracionales) y resultados medibles.

Las siglas representan la traduccion del ingles de las siglas de OKRS “objectives and key results” (objetivos y resultados clave). Las mismas representan una metodología para definir objetivos que ayuda a los equipos a establecer metas claras que a suvez puedan ser medibles.

Los OKR’s ofrecen una alternativa simple, flexible y eficaz para que la organización mantenga foco en los temas prioritarios y se plantee objetivos desafiantes que permita alcanzar resultados más allá de esperado.

Los OKR’s se pueden explicar como la fragmentación de los objetivos estratégicos en períodos cortos e inmediatos (usualmente trimestrales) y la bajada de estos a resultados clave o tareas concretas, medibles y desafiantes.

EL uso y adopcion de OKRs Implica y requirere un cambio en la forma de pensar y en la cultura de la organización ya que una vez seteados estos objetivos y sus Krs, estos regiran nuestras iniciativas a lo largo del periodo para el cual fueron definidas.