Pag 182

Testability

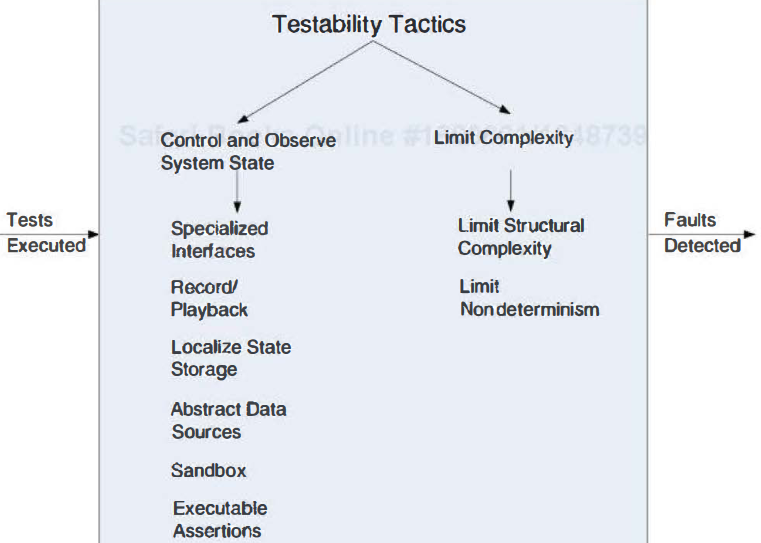
Entre 30 y 50% el costo del sistema es en pruebas.

Testability se refiere a la facilidad con que el software puede demostrar sus fallas.

Escenario

|  |  |
| --- | --- |
| Fuente de estimulo | Tester de unidad, de integración o sistemas, puede ser humano o una herramienta de testing. |
| Estimulo | Un conjunto de tests que puede ser de una clase, servicio, subsistema |
| Artefacto | Unidad de código, modulo, clase, subsistema |
| Entorno | El test puede ser en tiempo de desarrollo, en compilación, en ejecución |
| Respuesta | El sistema puede ser controlado para ejecutar un test especifico y el resultado del test puede ser observado |
| Medida de la respuesta | Se representa en que tan fácilmente un sistema “entrega” sus faults (fallas), puede incluir  -el esfuerzo para encontrar un fault,  -la dificultad para ejecutar el test  -duración del test |

De este modo, se definen categorías de tácticas



1. Control and Observe System State

Control y observación es la forma central para probar y los autores lo definen en este termino.

* 1. Specialized interfaces

Permite controlar o capturar valor de variables, ejemplos

Métodos get y set

Método que reporta

Método de reset para definir el estado interno

* 1. Record/playback

El estado que causa el fault a veces es difícil de recrear, ese estado se vuelve a “ejecutar” para observar y entender mejor la falla.

* 1. Localize state storage

Al empezar un sistema o subsistema o modulo en un estado arbitrario para aplicarle un test.

* 1. Abstract data sources

Controlar los datos de entrada para aplicar un test

* 1. Sandboxing

Separar una instancia del sistema a modo de experimento, este es fácil porque no causa consecuencias, se puede usar “escenario”, análisis, capacitación y simulación, una forma común de sandboxing es virtualizar recursos, ejemplos de virtualizar son: Stubs, mocks and dependency injection.

* 1. executable assertions

se inserta código en ciertas áreas y valida ciertas variables para indicar donde el programa está fallando, el resultado aumenta la observabilidad, básicamente es agregar verificaciones adicionales.

2 Limit Complexity

La complejidad es difícil de evaluar, esto es porque por definición el espacio o magnitud de la complejidad es muy amplia, es difícil recrear un estado exacto en un espacio muy grande.

2.1 Limit Structural complexity

Evitar o resolver dependencias cíclicas entre componentes, se aíslan o encapsulan dependencias del entorno externo y reduciendo dependencias en general.

En sistemas orientados a objetos, se puede simplificar la dependencia mediante herencia jerárquica. Limitar el número de clases de la clase derivada, limitar la profundidad del árbol de herencia, limitar polimorfismo y llamadas dinámicas.

También se recomienda alta cohesión, bajo acoplamiento y separación de preocupaciones.

Algunos estilos de arquitecturas ayudan con testability, por ejemplo la arquitectura por capas.

2.2 Limit nondeterminism

Se trata de la conducta compleja de los sistemas, el no determinismo es más difícil de Testear, se busca las fuentes de no determinismo como paralelismo sin restricciones.

contrastar contra las decisiones arquitecturales

|  |  |
| --- | --- |
| Localización de responsabilidades | Determinar que responsabilidades del sistema son criticas y requieren Testing, asegurar las siguientes actividades:  -ejecutar el test relacionado y capture resultados  -registre en bitácora (loggear) la actividad que resulte de un fault.  -controlar y observar estados relevantes relacionados con Testing |

|  |  |
| --- | --- |
| Modelo de coordinación | Asegurar que la coordinación del sistema y mecanismos de comunicación haga lo siguiente:  -soportar la ejecucion de un conjunto de tests y capturar los resultados en un sistema o entre sistemas.  -soportar la captura de actividad que resultó en un fault en un sistema o entre sistemas.  -soportar la inyección y monitoreo de un estado en los canales de comunicación para uso de testing en un sistema o entre sistemas.  -no introducir “no determinismo” a menos que sea necesario. |

|  |  |
| --- | --- |
| Modelo de datos | Determinar los datos sensibles y abstracciones de datos que requieran ser testeados para asegurar la correcta operación del sistema, y:  -asegurar que sea posible capturar los valores de instancias de las abstracciones de datos.  - asegurar que sea posible capturar los valores de instancias de las abstracciones de datos cuando se inserte un estado para recrear una falla.  -asegurar que la creación, inicialización, manipulación de las instancias sea ejecutada y capturada |

|  |  |
| --- | --- |
| Mapeo entre elementos arquitecturales | Determinar cómo probar los posibles mapeos de elementos arquitecturales (especialmente el mapeo de procesos a CPUs, hilos de procesos) de manera que el test capture el error esperado pueda.  -determinar si es posible testear la detección de mapeos de elementos arquitecturales ilegales. |

|  |  |
| --- | --- |
| Administrar recursos | -Asegurar que hay los recursos disponibles para ejecutar un conjunto de tests y capturar los resultados.  -asegurar que el entorno de pruebas sea el mas similar con el entorno real.  Asegurar que el sistema de soporte :  -testear los limites de los recursos  -capturar la mayor cantidad de evidencia en caso de falla.  -inyectar nuevos recursos y limites con propósitos de testing.  -proveer recursos virtuales para testing. |

|  |  |
| --- | --- |
| Binding time | -asegurar que los componentes que se unen en tiempo posterior puedan ser testeados en ese mismo entorno en tiempo posterior.  -asegurar que las uniones tardías puedan ser capturadas en la situación de una falla, de modo que se pueda recrear la falla.  -asegurar el rango mas grande de posibilidades de uniones (bindings) |

|  |  |
| --- | --- |
| Selección de tecnologia | Determinar que la tecnología disponible ayude a lograr los escenarios de Testing, inyección de fault, Testing regresivo, almacenamiento y así sucesivamente.  -Determinar en cuanto son sujetas de testing las tecnologías que han seleccionado.  Asegurar que las tecnologías seleccionadas soportan el nivel de testing adecuado al sistema.  Asegurar que el testing de inyectar estado de fault sea posible con las tecnologías seleccionadas. |