**DISPONIBILIDAD**

Habilidad del sistema de enmascarar o reparar fallas de forma que el periodo acumulado de interrupción no exceda un valor requerido sobre un intervalo de tiempo especificado

Un fallo puede dar por defectos internos o externos. Un defecto puede ser un fallo si no se corrige o enmascara. Un fallo es observable por el usuario, un defecto no. Refiere a la tolerancia a fallas

ESCENARIO GENERAL

* Fuente de estimulo

Fallo o defecto

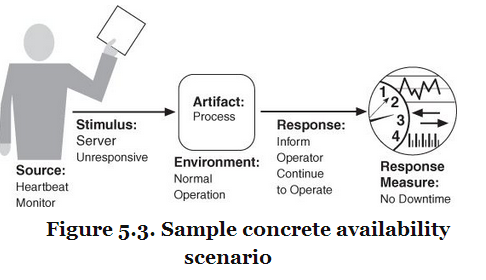
* Estimulo
* De omision: un componente que no responde
* De choque: un componente repetidamente no responde
* De sincronización: un componente responde antes o después de lo esperado
* De respuesta: Da un valor incorrecto
* Artefacto

Recurso que se desea esté disponible

* Medio ambiente

Sistema operando bajo restricciones por ocurrencia de defectos/fallos anteriores, donde se puede dar la baja. Sistema funcionando full donde se puede bajar el rendimiento por fallos/defectos

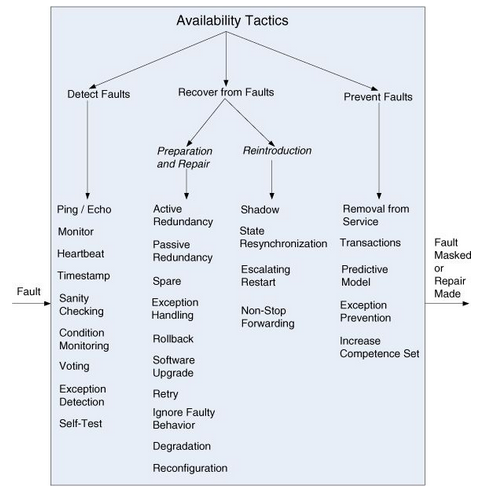
* Respuesta
* Registrar la falla
* Notificar a usuario u otros sistemas
* Cambiar el modo de operación a menor capacidad o funcionalidades
* Apagar sistemas externos
* Continuar operando normalmente
* Medida de respuesta
* % de disponibilidad
* Tiempo de reparacion
* Horas que debe estar disponible
* Horas en las que debe estar disponible



**TACTICAS**

Una falla ocurre cuando el sistema no entrega un servicio consistentemente con su especificación

Un defecto o una combinación de defectos es una potencial causa de fallos, la recuperación y reparación del sistema son aspectos fundamentales de la disponibilidad



**TACTICAS DE DETECCION DE DEFECTOS**

*PING/ECHO*

*Un componente emite un “ping” esperando un “echo” (eco). Puede realizarse mutuamente entre componentes responsables de una tarea o a servidores para comprobar si están operando con la eficiencia esperada*

*MONITOR*

Es un componente que chequea el estado de otras partes del sistema, puede detectar fallos o congestión de la red o de otros recursos compartidos producto de ataques de negación de servicios. Utiliza otras tácticas para realizar el monitoreo.

*HEARTBEAT – LATIDO*

Un componente emite un “latido” periódicamente y otro componente lo escucha. Si un latido falla se asume que el componente emisor fallo y se notifica. El latido a su vez puede enviar datos.

*TIME STAMP*

Es utilizado para detector la secuencia incorrecta de eventos, principalmente distribución de mensajes. Utiliza un reloj cuyas marcas de tiempo no son lo relevante

*SANITY CHECKING*

Chequea la validez o sensatez de una operación especifica o salidas de un componente

*CONDITION MONITORING*

Chequea las condiciones de un proceso o dispositivo, o valida supuestos hechos durante el diseño.

*VOTING*

Se envía el input a varios componentes, se elige mediante un algoritmo la mejor respuesta, las respuestas consideradas fallos pueden ser reportadas.

Lo más complejo es la implementación de la lógica del votador, la cual puede decidir por similitud en las respuestas (la cantidad que coinciden), un promedio, un componente preferido, entre otras.

Puede basarse en:

-Replicación: se tienen componentes idénticos, no asegura detectar errores de implementación

-Redundancia funcional: se tienen diferentes implementaciones

-Redundancia analítica: Intenta tolerar errores de especificación, dadas las respuestas , se fija en que entorno trabaja el componente que la envió, para saber cuál es el grado de error que puede aceptar. Es más inteligente

*EXCEPTION DETECTION*

Las excepciones se desencadenan cuando se reconoce una falla y se manejan en el mismo proceso en el que fallo

*SELF TEST*

**TACTICAS PARA RECUPERACION DE FALLAS**

**PREPARACION Y REPARACION**

*ACTIVE REDUNDANCE* – REDUNDANCIA ACTIVA

Se envían los input a varios componentes, la respuesta que se toma es la primera que se recibe, pero el resto de los componentes permanecen actualizados

*PASIVE REDUNDANCE* – REDUNDANCIA PASIVA

El o los componentes backup permanecen “inactivos” no apagados. Solo opera el principal pero este envia actualizaciones al resto de los componentes redundantes, de manera que si el primero deja de funcionar, un backup pueda activarse rápidamente teniendo el ultimo estado ok.

*SPARE*

Las redundancias están fuera de servicio, se ponen en operación cuando el componente operando falla, con lo cual lleva un tiempo de activación para ponerse en funcionamiento y suplir al fallido.

*ROLLBACK*

Volver a atrás utilizando la redundancia, viene de la mano también con el manejo de transacciones.

*SOFTWARE UPGRADE* – ACTUALIZACION DE SOFTWARE

Refactoreo y parches. Su objetivo es lograr mejoras de código sin afectas el servicio. Esto puede ser realizado como un parche de una función o clase (arreglo de errores) o partiendo de la redundancia, sin dar de baja el servicio, realizar cambios aportando nuevas características y capacidades

*RETRY* – VOLVER A INTENTAR

Asume que la falla que causo el fracaso es transitoria y que un nuevo intento puede conducir al éxito. Debe haber un límite en el número de reintentos que se intenten antes de declarar un fallo permanente

*IGNORE FAULTY BEHAVIOR* – IGNORAR EL COMPORTAMIENTO DEFECTUOSO

Esta táctica requiere ignorar los mensajes enviados desde una fuente en particular cuando se determina que los mensajes son falsos. Por ejemplo, ignorar los mensajes d un componente externo que intenta lanzar un ataque de negación de servicio

*DEGRADATION* – DEGRADACION DEL SERVICIO

Mantiene las funciones más críticas del Sistema en la presencia de fallas de componentes, dejando caer las funciones menos críticas. Se reducen las funcionalidades en lugar de causar un fallo del sistema completo.

*RECONFIGURATION* – RECONFIGURACION DE COMPONENTES

Se reasignan las responsabilidades a los recursos que dejaron de funcionar (potencialmente restringidos), mientras los restantes se encargan de las funcionalidades como sea posible

**REINTRODUCTION**

*SHADOW* – SOMBRA

Después de una falla, el componente trabaja en modo de prueba para asegurarse de que la corrección fue exitosa y que el componente está trabajando nuevamente de forma correcta

*STATE RESYNCHRONIZATION* – RESINCRONIZACION

Se utiliza en combinación con la redundancia. Cuando se utiliza redundancia activa la sincronización del estado se produce de manera orgánica, debido a que los componentes activos y de reserva reciben y procesan las entradas idénticas en paralelo y además se comparan periódicamente para asegurar la sincronización.

Cuando se utiliza junto con la táctica redundancia pasiva, la re sincronización del estado se basa únicamente en la información de estado periódica transmitida desde el componente activo para el componente de reserva, normalmente a través de los puntos de control.

*ESCALATING RESTART*

Reinicio de componentes en diferentes granularidades, afectando al minimo el nivel de servicio. Por ejemplo:

Nivel 0: se reinician todos los temas secundarios del componente

Nivel 1: se libera y reinicia la memoria sin protección

Nivel 2: se libera y reinicia toda la memoria, lo que obliga a recargar todas las aplicaciones

Nivel 3: implica el reinicio de todo el sistema

Se usa para una degradación de servicio elegante

*NON-STOP FORWARDING* – desvío

Consta de 2 partes, la gestión de la conectividad y el enrutamiento. Por ejemplo cuando se detecta una falla en un router, se desvía el envió de paquetes a un router vecino, mientras el anterior se recupera y valida. Se vuelve a generar el plano de enrutamiento, re asignando al router fallido de nuevo a sus tareas

**PREVENCION DE DEFECTOS**

*REMOVAL FROM SERVICES* – SEPARACION DE SERVICIOS

Colocación temporal de un componente del sistema en un estado fuera de servicio con el fin de mitigar los posibles fallos del sistema. Se quita el componente y se lo reinicia antes de que acumule defectos y provoque el fallo del sistema

*TRANSACTIONS* – TRANSACCIONES

Se realizan transacciones semánticas para asegurar que los mensajes asincrónicos intercambiados entre componentes son atómicos, consistentes, aislados y durables. Se realiza el bloqueo actualización y comiteo para no provocar problemas de consistencia entre los componentes para acceder a los recursos.

*PREDICTIVE MODEL* – MODELO PREDICTIVO

Se combina con un monitor, se emplea para controlar el estado de salud de un proceso del sistema para asegurar que el sistema está operando dentro de los parámetros de funcionamiento esperado, y para tomar medidas correctivas cuando se detectan condiciones que en futuro pueden originar fallas.

*EXCEPTION PREVENTION* – PREVENCION DE EXCEPCIONES

Técnicas empleadas para el propósito de la prevención de excepciones del sistema que se produzcan. El uso de clases de excepción, que permite que un sistema se recupere de forma transparente de las excepciones del mismo, se discutió previamente.

*INCREASE SET* – AUMENTAR CONJUNTO DE COMPETENCIAS

El conjunto de competencias es el conjunto de estados en los que se considera que un programa es competente para operar. Caso ejemplo de un estado de no competencia es el denominador cero en programas que realizan divisiones. Cuando un componente genera una excepción está indicando que encontró un estado en no está dentro de sus competencias, no sabe qué hacer en ese caso.