TOLERANCIA
A
FALLAS

Conceptos básicos

- La **fiabilidad** (*reliability*) de un sistema es una medida de su conformidad con una especificación autorizada de su comportamiento, es la probabilidad de que ese sistema funcione o desarrolle una cierta función, bajo condiciones fijadas y durante un período determinado.
- Una **avería** (*failure*) es una desviación del comportamiento de un sistema respecto de su especificación.
- Las averías se manifiestan en el comportamiento externo del sistema, pero son resultado de **errores** (*errors*) internos
- Las causas mecánicas o algorítmicas de los errores se llaman **fallos** (*faults*)

- Avería
  - No realización de alguna acción esperada
- Error
  - Manifestación de una falla
- Falla
  - Defecto dentro de un componente de hardware o software

Fallos encadenados

Fallas de funcionamiento

- Los fallas de funcionamiento de un sistema pueden tener su origen en:
  - Una especificación inadecuada
  - **Errores de diseño del software**
  - Averías en el hardware
  - Interferencias transitorias o permanentes en las comunicaciones

Tipos de fallos

- **Fallos transitorios**
  - desaparecen solos al cabo de un tiempo
  - ejemplo: interferencias en comunicaciones
- **Fallos permanentes**
  - permanecen hasta que se reparan
  - ejemplo: daños de hardware, errores de software
- **Fallos intermitentes**
  - fallos transitorios que ocurren de vez en cuando
  - ejemplo: calentamiento de un componente de hardware

Otros

TIPO DE FALLO	DESCRIPCION
FALLO CRASH	El servidor se detiene, pero estaba operando correctamente.
Fallo por Omision Omision de recibido Omision de envio	Un servidor falla en responder a las peticiones El servidor falla en recibir mensajes El servidor falla en mandar mensajes
Fallo de tiempo	La respuesta del servidor no esta en el intervalo de tiempo especificado.
Fallo de respuesta Fallo de valor Fallo de estado de transicion	La respuesta del servidor es incorrecta El valor de la respuesta es incorrecto El servidor se desvia del flujo de control
Fallo arbitriario	El servidor produce fallos arbitrarios en tiempo indefinidos.

Tipos de avería

Prevención y tolerancia de fallas

- Hay dos formas de aumentar la fiabilidad de un sistema:
  - **Prevención de fallos**
    - Se trata de evitar que se introduzcan fallos en el sistema antes de que entre en funcionamiento
  - **Tolerancia de fallas**
    - Se trata de conseguir que el sistema continúe funcionando aunque se produzcan fallos
- En ambos casos el objetivo es desarrollar sistemas con modos de fallo bien definidos

Prevención de fallos

- Se realiza en dos etapas:
- **Evitación de fallos**
  - Se trata de impedir que se introduzcan fallos durante la construcción del sistema
- **Eliminación de fallos**
  - Consiste en encontrar y eliminar los fallos que se producen en el sistema una vez construido

Técnicas de evitación de fallos

- **Hardware**
  - Utilización de componentes fiables
  - Técnicas rigurosas de montaje de subsistemas
  - Apantallamiento de hardware
- **Software**
  - Especificación rigurosa o formal de requisitos
  - Métodos de diseño comprobados
  - Lenguajes con abstracción de datos y modularidad
  - Utilización de entornos de desarrollo con computador (CASE) adecuados para gestionar los componentes

Técnicas de eliminación de fallos

- **Comprobaciones**
  - Revisiones de diseño
  - Verificación de programas
  - Inspección de código

- **Pruebas** (*tests*)
  - Son necesarias, pero tienen problemas:
    - no pueden ser nunca exhaustivas
    - sólo sirven para mostrar que hay errores, no que no los hay
    - a menudo es imposible reproducir las condiciones reales
    - los errores de especificación no se detectan

Limitaciones de la prevención de fallos

- Los componentes de hardware fallan, a pesar de las técnicas de prevención
- La prevención es insuficiente si
  - la frecuencia o la duración de las reparaciones es inaceptable
  - no se puede detener el sistema para efectuar operaciones de mantenimiento
- La alternativa es utilizar técnicas de **tolerancia de fallos**

Tolerancia a fallas

- Es la propiedad que permite a un sistema continuar operando adecuadamente en caso de una falla en alguno de sus componentes.
- La tolerancia de fallas es muy importante en aquellos sistemas que deben funcionar todo el tiempo.
- Una forma de lograr tolerancia de fallas, es duplicar cada componente del sistema.

Grados de tolerancia de fallos

- **Tolerancia completa** (*fail operational*).
  - El sistema sigue funcionando, al menos durante un tiempo, sin perder funcionalidad ni prestaciones
- **Degradación aceptable** (*failsoft*).
  - El sistema sigue funcionando con una pérdida parcial de funcionalidad o prestaciones hasta la reparación del fallo
- **Parada segura** (*failsafe*).
  - El sistema se detiene en un estado que asegura la integridad del entorno hasta que se repare el fallo

El grado de tolerancia de fallos necesario depende de la aplicación

Redundancia

- La tolerancia de fallos se basan en la **redundancia**
- Se utilizan componentes adicionales para **detectar** los fallos y **recuperar** el comportamiento correcto
- Esto aumenta la complejidad del sistema y puede introducir fallos adicionales
- Es mejor separar los componentes tolerantes del resto del sistema

Redundancia

- de Información
  - replicación de datos, códigos de corrección de errores.
- de Recursos
  - se agrega equipo adicional para tolerar la pérdida o mal funcionamiento de ciertos componentes
- de Tiempo
  - se realiza una acción, y de ser necesario, se vuelve a realizar

Redundancia en hardware

- **Redundancia estática**
  - Los componentes redundantes están siempre activos
  - Se utilizan para ***enmascarar*** los fallos
  - Ejemplo:
    - Redundancia modular triple (*ó N*)
- **Redundancia dinámica**
  - Los componentes redundantes se activan cuando se detecta un fallo
  - Se basa en la ***detección*** y posterior ***recuperación*** de los fallos
  - Ejemplos:
    - sumas de comprobación
    - bits de paridad

Tolerancia de fallos de software

- Técnicas para detectar y corregir errores de diseño

- **Redundancia estática**
  - Programación con N versiones

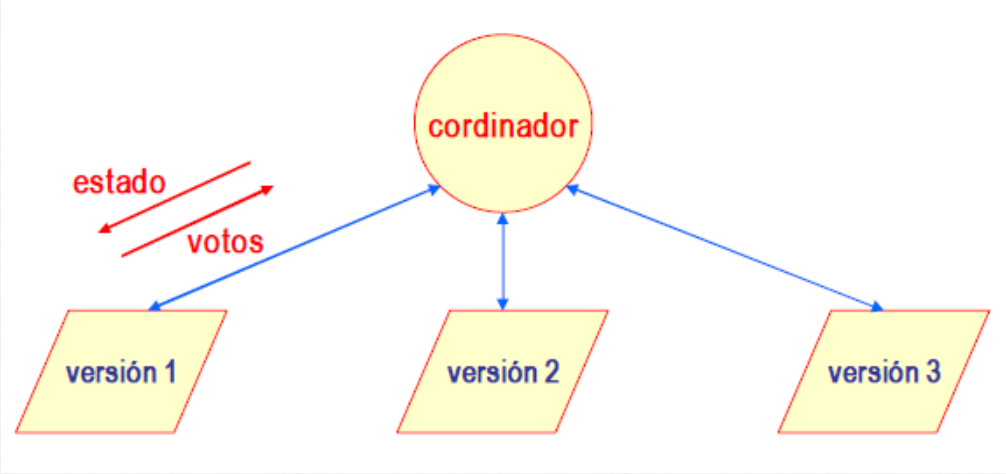
- **Redundancia dinámica**
  - Dos etapas: detección y recuperación de fallos
  - **Bloques de recuperación**
    - Proporcionan recuperación hacia atrás
  - **Excepciones**
    - Proporcionan recuperación hacia adelante

Programación con N versiones

- **Diversidad de diseño**
  - N programas desarrollados independientemente con la misma especificación
  - Sin interacciones entre los equipos de desarrollo

- **Ejecución concurrente**
  - Proceso coordinador *(driver)*
    - » intercambia datos con los procesos que ejecutan las versiones
  - Todos los programas tienen las mismas entradas
  - Las salidas se comparan
  - Si hay discrepancia se realiza una **votación**

Programación con N versiones



- Puntos de comparación: puntos dentro de la versión donde deben comunicar sus votos al proceso coordinador.
- Granularidad de las votaciones:
  - Fina => semejanza en los detalles de estructura de los programas, reduce la independencia entre versiones.
  - Gruesa =>Gran divergencia en los resultados obtenidos aunque menos sobrecarga por el sistema de votación.

Problemas de la programación con N versiones

- La correcta aplicación de este método depende de:

- **Especificación inicial.**
  - Un error de especificación aparece en todas las versiones.
- **Desarrollo independiente.**
  - No debe haber interacción entre los equipos.
  - No está claro que distintos programadores cometan errores independientes.
- **Presupuesto suficiente.**
  - Los costes de desarrollo se multiplican.
  - El mantenimiento es también más costoso.

- Se ha utilizado en sistemas de aviónica críticos.

Redundancia dinámica en software

- Los componentes redundantes sólo se ejecutan cuando se detecta un error

- Se distinguen cuatro etapas:
  1. **Detección de errores**
  2. **Evaluación y confinamiento de los daños**
  3. **Recuperación de errores**
     - Se trata de llevar el sistema a un estado correcto, desde el que pueda seguir funcionando
  4. **Reparación de fallos**
     - Aunque el sistema funcione, el fallo puede persistir y hay que repararlo

Detección de errores

- Por el entorno de ejecución
  - hardware (p.ej.. instrucción ilegal)
  - núcleo o sistema operativo (p.ej. puntero nulo)

- Por el software de aplicación
  - Duplicación (redundancia con dos versiones)
  - Comprobaciones de tiempo
  - Inversión de funciones
  - Códigos detectores de error
  - Validación de estado
  - Validación estructural

Evaluación y confinamiento de daños

- Es importante *confinar* los daños causados por un fallo a una parte limitada del sistema
- Se trata de estructurar el sistema de forma que se minimice el daño causado por los componentes defectuosos (*comportamiento estancos, firewalls*)
- Técnicas
  - Descomposición modular: confinamiento **estático**
  - Acciones atómicas: confinamiento **dinámico**

Recuperación de errores

- Es la etapa más importante
- Se trata de situar el sistema en un estado correcto desde el que pueda seguir funcionando
- Hay dos formas de llevarla a cabo:
  - **Recuperación directa** (hacia adelante)
    - Se avanza desde un estado erróneo haciendo correcciones sobre partes del estado
  - **Recuperación inversa** (hacia atrás)
    - Se retrocede a un estado anterior correcto que se ha guardado previamente

Recuperación directa

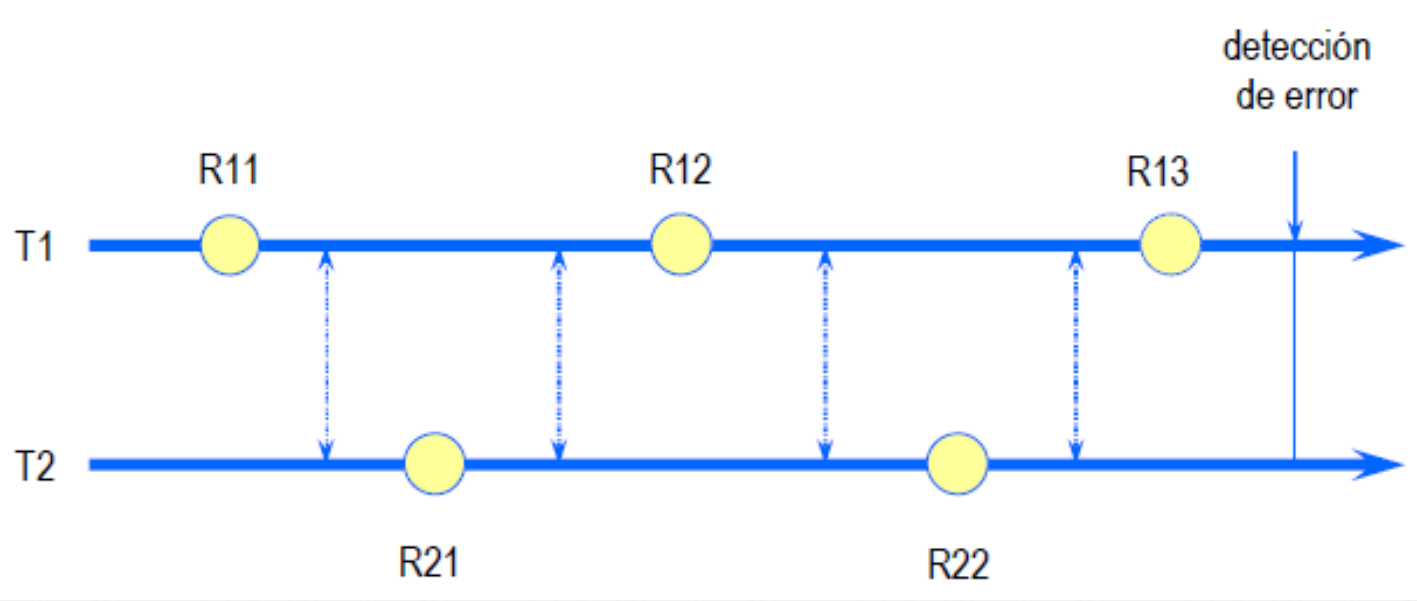
- La forma de hacerla es específica para cada sistema
- Depende de una predicción correcta de los posibles fallos y de su situación
- Hay que dejar también en un estado seguro el sistema controlado
- Ejemplos
  - punteros redundantes en estructuras de datos
  - códigos autocorrectores

Recuperación inversa

- Consiste en retroceder a un estado anterior correcto y ejecutar un segmento de programa alternativo (con otro algoritmo)
  - El punto al que se retrocede se llama **punto de recuperación**
- No es necesario averiguar la causa ni la situación del fallo
- Sirve para fallos imprevistos
- ¡Pero no puede deshacer los errores que aparecen en el sistema controlado!

Efecto dominó

- Cuando hay tareas concurrentes la recuperación se complica



Solución: establecer puntos de recuperación globales
Conjunto de puntos de recuperación globales: línea de recuperación, líneas de recuperación consistentes para todas las tareas

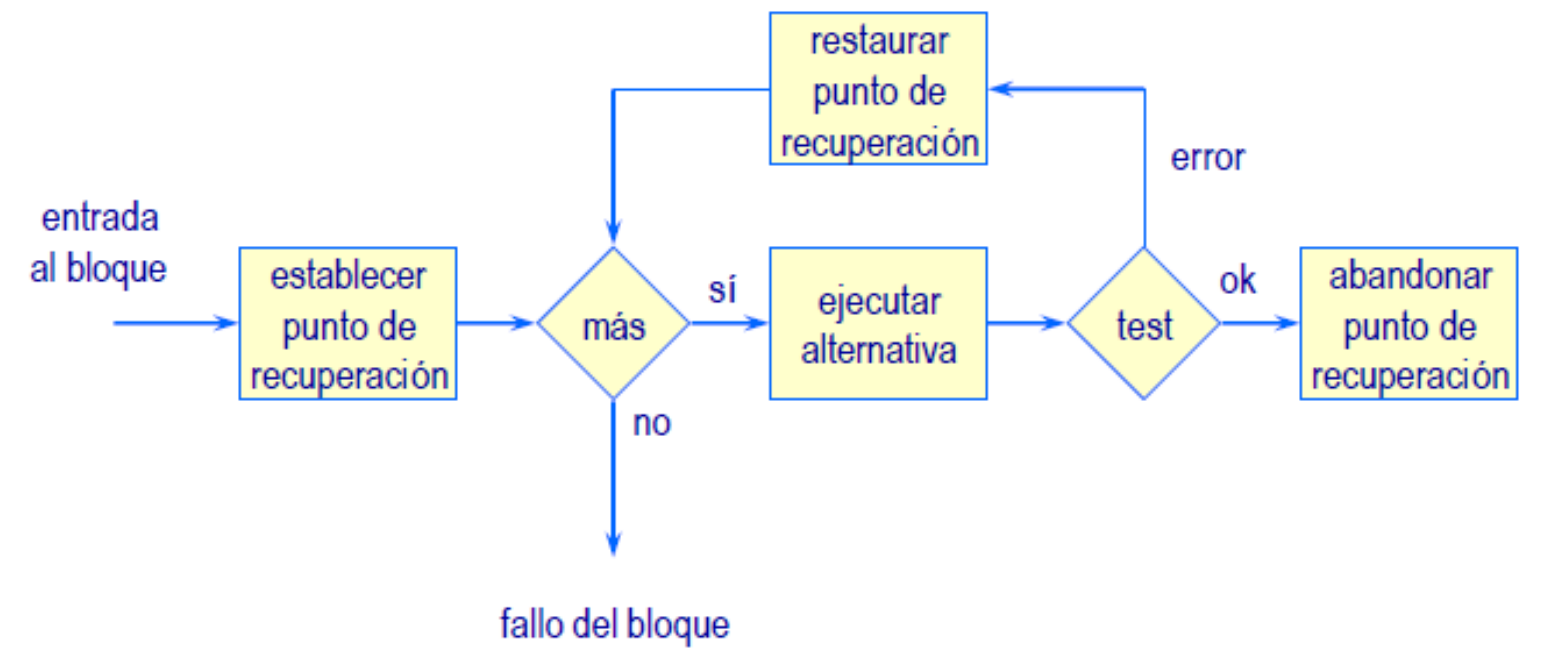
Reparación de fallos

- La reparación automática es difícil y depende del sistema concreto
- Hay dos etapas
  - **Localización del fallo**
    - Se pueden utilizar técnicas de detección de errores
  - **Reparación del sistema**
    - Los componentes de hardware se pueden cambiar
    - Los componentes de software se reparan haciendo una nueva versión
    - En algunos casos puede ser necesario reemplazar el componente defectuoso sin detener el sistema

Bloques de recuperación

- Es una técnica de recuperación inversa integrada en el lenguaje de programación
- Un **bloque de recuperación** es un bloque tal que:
  - su entrada es un *punto de recuperación*
  - a su salida se efectúa una *prueba de aceptación*
    - sirve para comprobar si el **módulo primario** del bloque termina en un estado correcto
  - si la prueba de aceptación falla,
    - se restaura el estado inicial en el punto de recuperación
    - se ejecuta un **módulo alternativo** del mismo bloque
  - si vuelve a fallar, se siguen intentando alternativas
  - cuando no quedan más, el bloque falla y hay que intentar la recuperación en un nivel más alto

Esquema de recuperación



- Puede haber bloques anidados
  - si falla el bloque interior, se restaura el punto de recuperación del bloque exterior

Prueba de aceptación

- Es fundamental para el buen funcionamiento de los bloques de recuperación
- Hay que buscar un compromiso entre detección exhaustiva de fallos y eficiencia de ejecución
- Se trata de asegurar que el resultado es ***aceptable,*** no forzosamente ***correcto***
- Pero hay que tener cuidado de que no queden errores residuales sin detectar

Comparación

- **N versiones**
  - Redundancia estática
  - Diseño
    - algoritmos alternativos
    - proceso guía
  - Ejecución
    - múltiples recursos
  - Detección de errores
    - votación

Bloques de recuperación

- Redundancia dinámica
- Diseño
  - algoritmos alternativos
  - prueba de aceptación
- Ejecución
  - puntos de recuperación
- Detección de errores
  - prueba de aceptación

¡Ambos métodos son sensibles a los errores en los requisitos!

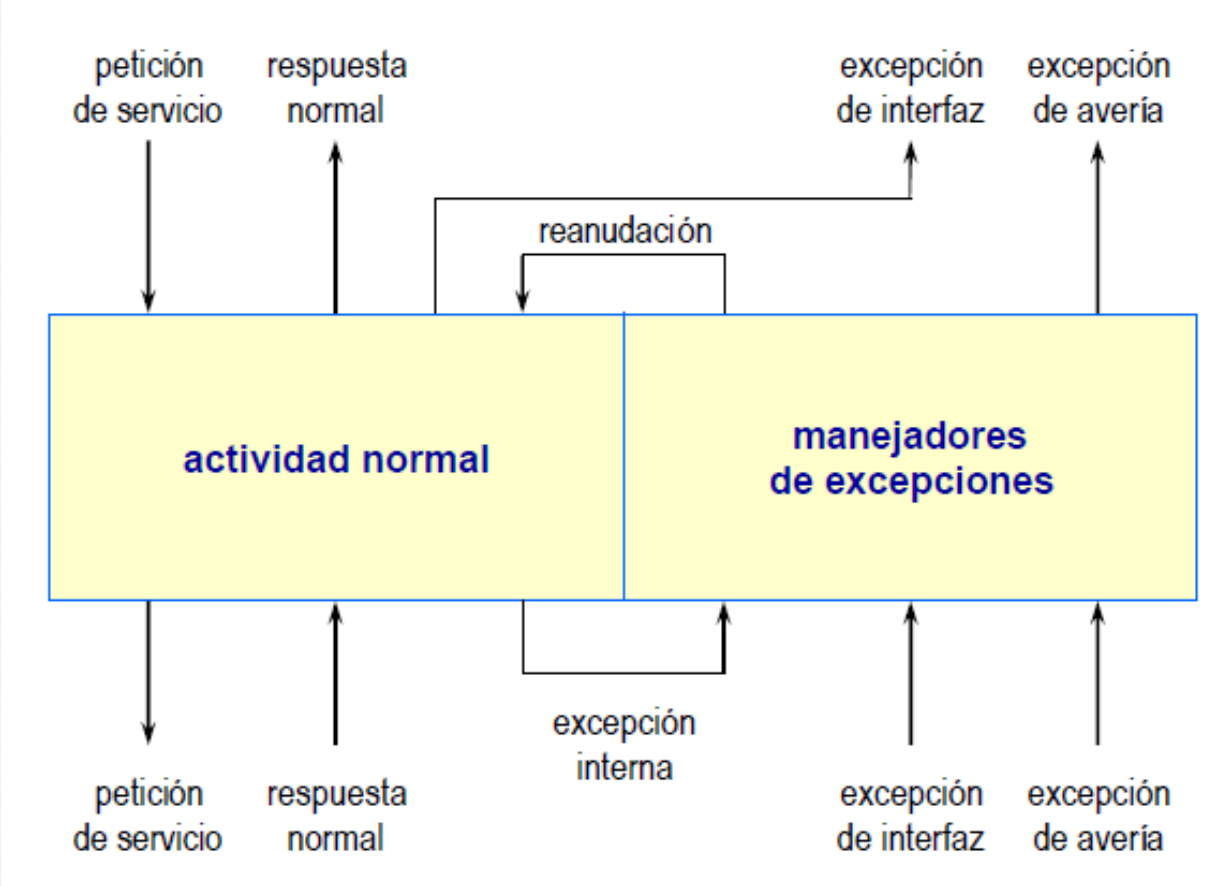
Excepciones

- Una **excepción** es una manifestación de un cierto tipo de error
- Cuando se produce un error, se ***eleva*** la excepción correspondiente en el contexto donde se ha invocado la actividad errónea
- Esto permite ***manejar*** la excepción en este contexto
- Se trata de un mecanismo de ***recuperación directa*** de errores (no hay vuelta atrás)
- Pero se puede utilizar para realizar recuperación inversa también

Aplicaciones de las excepciones

- Tratar situaciones anormales en el sistema controlado
- Tolerar fallos de diseño de software
- Facilitar un mecanismo generalizado de detección y corrección de errores

Componente ideal de un sistema tolerante a fallos



Seguridad y fiabilidad

- Un sistema es **seguro** si no se pueden producir situaciones que puedan causar muertes, heridas, enfermedades, ni daños en los equipos ni en el ambiente

  Un **accidente** (*mishap*) es un suceso imprevisto que puede producir daños inadmisibles

- Un sistema es **fiable** si cumple sus especificaciones

- Seguridad y fiabilidad pueden estar en conflicto

La seguridad es la probabilidad de que no se produzcan situaciones que puedan conducir a accidentes, independientemente de que se cumpla la especificación o no

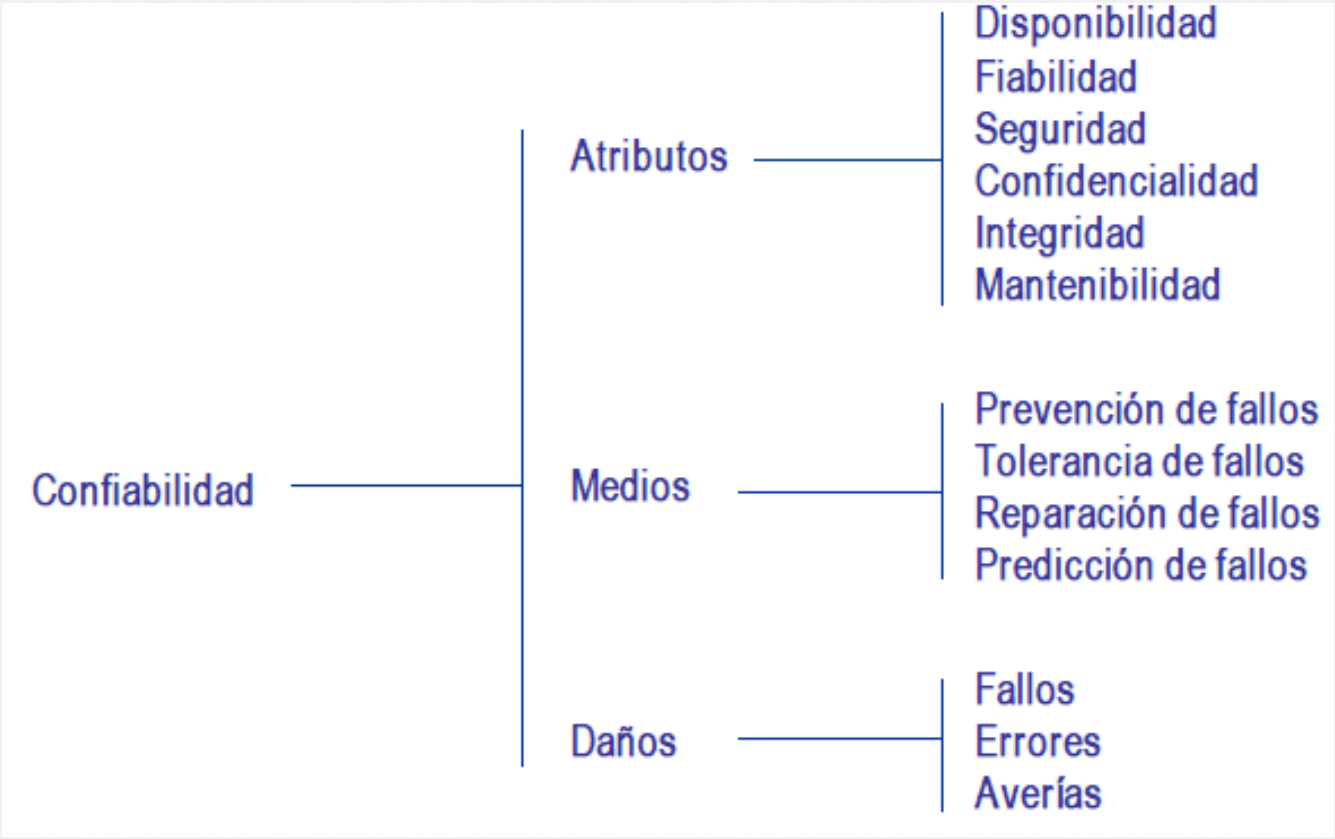
Confiabilidad

- Es la medida en la cual la confianza se puede poner justificadamente en el servicio que se obtiene del sistema (Laprie (1992).

- La tolerancia a fallas trata con la confiabilidad de un sistema, o sea como asegurar que el sistema corra correctamente.

Confiabilidad

- La confiabilidad es una propiedad de los sistemas que permite confiar justificadamente en el servicio que proporcionan.
- La confiabilidad de un sistema es una propiedad más amplia que la fiabilidad
- Tiene varios aspectos


Confiabilidad
Atributos
Disponibilidad
Fiabilidad
Seguridad
Confidencialidad
Integridad
Mantenibilidad
Medios
Prevención de fallos
Tolerancia de fallos
Reparación de fallos
Predicción de fallos
Daños
Fallos
Errores
Averías