TEMA 3. Garantía de funcionamiento

(Duración aproximada: 3 horas)

1. Conceptos básicos y definiciones.

1.1. Introducción.

Todos los sistemas son susceptibles de averiarse. Tanto los sistemas biológicos –enfermedades, accidentes– como los eléctricos, mecánicos y, por supuesto, los informáticos. Por mucho cuidado que se tenga en el diseño e implementación de un sistema, la probabilidad de que se averíe nunca es cero. Los circuitos integrados por ejemplo, y en particular las memorias, son susceptibles de sufrir fallos por el bombardeo con partículas alfa a que están sometidos todos los cuerpos. Cuanto menor es el tamaño de los transistores, y menor la energía de los estados de los bits, la probabilidad de error aumenta. Si el sistema tiene una función crítica según criterios económicos –el sistema informático de un satélite, por ejemplo, que no puede ser reparado y que compromete el funcionamiento de una inversión de muy elevado conste– ecológicos o de seguridad humana –el sistema de control de una planta petroquímica o de una central nuclear– además de seguir un cuidadoso mantenimiento preventivo, se deben emplear otras técnicas para garantizar el funcionamiento correcto del sistema informático con mucha más seguridad que en el caso convencional.

Este tema trata de los conceptos y técnicas relacionados con los sistemas que deben ofrecer una elevada garantía de funcionamiento.

1.2. Definiciones.

La terminología empleada al discutir acerca de la garantía de funcionamiento de los sistemas utiliza muy a menudo palabras de significado similar en el lenguaje coloquial para definir de forma precisa conceptos claramente diferenciados en su ámbito. Es conveniente por ello definir precisamente todos los términos que se van a emplear, y los conceptos asociados que definen.

De este modo, definimos la **garantía de funcionamiento** (*dependability*) de un sistema como la propiedad que permite a sus usuarios depositar una confianza justificada en el servicio que les entrega. Para estudiar la garantía de funcionamiento y cómo obtener sistemas en los que se pueda confiar, debemos conocer los impedimentos, atributos y medios relacionados con esta propiedad.

1.2.1. Impedimentos de la garantía de funcionamiento.

Los imedimentos de la garantía de funcionamiento son circunstancias no deseadas –pero no necesariamente inesperadas– que causan la interrupción del servicio –en un sentido amplio– que debía entregar el sistema. El impedimento que percibe directamente el usuario se denomina avería. El proceso real que genera una avería cualquiera en un sistema recorre tres etapas desde la aparición de su causa primera hasta que ésta se manifiesta como un funcionamiento incorrecto de aquél. Cada una de estas etapas viene definida por un término preciso:

- **Fallo** (*fault*): Es cualquier defecto o imperfección, físico o de diseño, en algún elemento del sistema. Un fallo sería una línea abierta o cortocircuitada en un circuito, reflexiones indeseadas en la transmisión de alguna señal a lo largo de un bus o un desbordamiento no previsto al realizar alguna operación aritmética mediante software.

- **Error** (*error*): Es la consecuencia inmediata o manifestación en el sistema de un fallo. Normalmente será un estado erróneo del sistema, o un estado al cual el sistema no debería haber llegado según la misma secuencia de acontecimientos libre de fallos. Si bien un error es siempre consecuencia de un fallo, no todos los fallos producen errores, quedando a veces desapercibidos o enmascarados.

- **Avería** (*failure*): Una avería ocurre cuando el sistema no se comporta de acuerdo con la especificación. Según lo visto anteriormente, una avería es la consecuencia de un error, aunque no todos los errores dan lugar a averías. En la caracterización o detección de las averías es fundamental la existencia de una especificación clara del servicio a prestar por el sistema, incluyendo aspectos fundamentales como el tiempo de respuesta, y probablemente otros circunstanciales como ruidos generados, temperatura disipada, consumo...

De acuerdo con estas tres definiciones, un fallo puede causar, en primer lugar, un error que a su vez puede ocasionar una avería en el sistema. Además, estas tres definiciones, aplicadas sobre el sistema objeto de estudio, cambian al variar el nivel de detalle con el que se esté estudiando el sistema, de forma jerárquica: una avería en un subsistema podrá ser un fallo o un error en el sistema que lo engloba, y ser consecuencia de un fallo o error en un subsistema contenido en aquél. Este proceso recursivo debe llegar exclusivamente hasta los componentes más elementales susceptibles de ser mantenidos. No tendría sentido proceder hasta los componentes de las tarjetas de circuito impreso de un sistema, sus transistores o las moléculas de silicio dopado que los forman. Los subsistemas más allá de los cuales no se debe descender se denominan componentes atómicos.

Clasificación de los fallos.

Los fallos y sus fuentes son sumamente diversos. Su estudio es particularmente importante pues, cómo se ha visto, son la causa originaria de toda avería en los sistemas. Por ello se han establecido múltiples clasificaciones de los mismos, atendiendo a muy diversos criterios. Vamos a repasar las clasificaciones más importantes:

- Atendiendo a la relación del fallo con las *fronteras físicas del sistema*, tenemos:
  - **fallos internos** que son partes propias del sistema o de su estado que acabarán produciendo un error,
  - **fallos externos** que son el resultado de la interacción del sistema con su entorno físico o humano.
- Atendiendo *fase de creación del fallo* con respecto a la vida del sistema, tenemos:
  - **fallos de diseño** resultantes de las imperfecciones originadas bien durante el desarrollo del sistema, durante las modificaciones posteriores o durante el establecimiento de los procedimientos de explotación y mantenimiento.
  - **fallos de operación** que se originan durante la explotación del sistema.
- Atendiendo a *persistencia temporal de los fallos*, tenemos:
  - **fallos permanentes** cuya presencia no está ligada a condiciones puntuales, sean internas –actividad del sistema– o externas –entorno.
  - **fallos temporales** cuya presencia está ligada a aquellas condiciones y, por lo tanto, están presentes un tiempo limitado. Los fallos *temporales externos* se denominan **transitorios** pues aparecen esporádicamente y no premanecen en el sistema. Los *temporales internos* se denominan **intermitentes** pues aparecen de forma recurrente al estar su causa en el propio sistema.

1.2.2. Atributos y parámetros.

Según las aplicaciones de los sistemas se pone el énfasis en diferentes facetas de la garantía de funcionamiento, lo que quiere decir que puede ser vista desde diferentes propiedades, que permiten la definición de sus atributos:

- Fiabilidad (*reliability*): desde el punto de vista de la *continuidad en el servicio*. Un sistema será fiable si es capaz de funcionar largo tiempo sin averías. Es propio de los sistemas con difícil mantenimiento, como los equipos informáticos embarcados en misiones espaciales no tripuladas.
- Disponibilidad (*availability*): desde el punto de vista de la *preparación para el servicio*. Un sistema con alta disponibilidad estará en funcionamiento la mayor parte del tiempo. No sólo es importante la ausencia de averías sino también la reparación rápida de estas. Es propio de servidores de bases de datos en sistemas transaccionales con elevada carga –reservas de vuelos, sistemas bancarios.
- Seguridad-inocuidad (*safety*): desde el punto de vista de la *no ocurrencia de averías catastróficas*. El sistema puede dejar de funcionar, pero de forma controlada y segura. Es propio de sistemas críticos de controls –centrales nucleares, petroquímicas, etcétera.
- Seguridad-confidencialidad (*security*): desde el punto de vista de la *prevención del acceso y/o de la manipulación no autorizada de la información*. El sistema debe garantizar la privacidad y seguridad de la información.

Para poder cuantificar estos atributos se han definido los siguientes parámetros, que se expresan normalmente en horas:

- **Tiempo medio hasta la avería, MTTF** (*Mean Time To Failure*). Es el tiempo promedio desde que el sistema entra en funcionamiento hasta que se produce una avería.
- **Tiempo medio de reparación, MTTR** (*Mean Time To Repair*). Es el tiempo promedio que transcurre desde que un sistema se avería hasta que vuelve a estar en funcionamiento.
- **Tiempo medio entre averías, MTBF** (*Mean Time Between Failures*). Es la suma de los dos anteriores, y viene a significar el promedio de averías de los equipos.

Un sistema con una elevada fiabilidad tendrá un MTTF muy alto. Un sistema con una alta disponibilidad puede permitir un MTTF más bajo si el MTTR es también muy bajo.

1.2.3. Medios para garantizar el funcionamiento de los sistemas.

Los sistemas informáticos son cada vez más complejos, por ello la probabilidad de que se produzcan fallos no es despreciable. Para obtener sistemas con una elevada garantía de funcionamiento se debe disponer de ciertas técnicas que permitan este objetivo. Estas técnicas, cuya fase de apliación aparece en la figura 1, son los medios de la garantía del funcionamiento.

- **Prevención de fallos:** Las técnicas de prevención de fallos se deben aplicar antes de que el sistema entre en su fase de explotación. Estas técnicas están estrechamente ligadas con el uso de metodologías y prácticas de diseño, depuración e instalación adecuadas.
  - **Evitación de fallos:** Todavía se está muy lejos de poder aplicar verificaciones formales a los diseños de los sistemas informáticos reales, ya sea al componente matarial como al lógico. De esta forma, la evitación de fallos debe conseguirse mediante el uso de metodologías adecuadas.
  - **Eliminación de fallos:** Una vez el sistema ha sido diseñado, y antes de su explotación, se deeben ralizar procesos de verificación y prueba de su funcionamiento. Los fallos detectados durante estas pruebas serán eliminados, y el diseño original corregido adecuadamente.
- **Tolerancia a fallos:** Una vez el sistema está en explotación, sólo se puede mejorar su garantía de funcionamiento si se han incluido en su diseño técnicas de tolerancia a fallos. Durante el funcionamiento del sistema, estas técnicas sirven para evitar que los fallos se propaguen y acaben dando lugar a las averías. En los apartados siguientes se trata específicamente este tema.

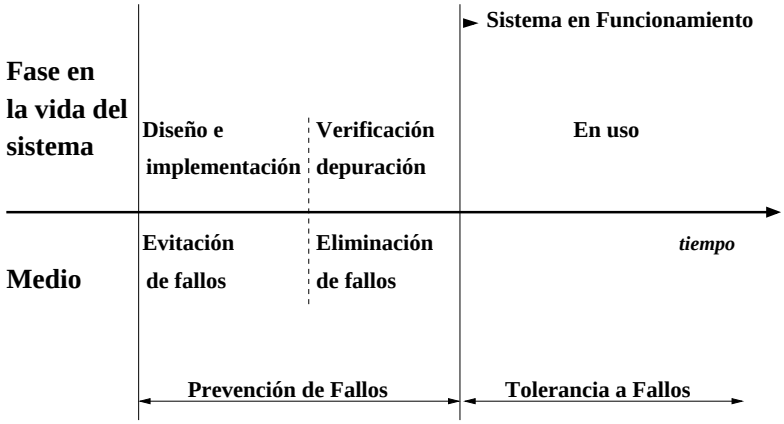

Figura 1: Aplicación de los medios para mejorar la garantía de funcionamiento.

1.3. Metodología de diseño.

La finalidad de las técnicas de tolerancia a fallos es obtener sistemas con una elevada garntía de funcionamiento, partiendo de componentes idénticos a los que se usan en el diseño de sistemas normales y que, como es sabido, son susceptibles de error. Los sistemas tolerantes a fallos, además, participan de la estructura general común a cualquier sistema computacional, a la que se incorporan, de una u otra forma, características para mejorar la garantía de funcionamiento. Estas afirmaciones plantean una cierta paradoja de la tolerancia a fallos: partiendo de los mismos componentes que cualquier otro sistema, se obtiene un sistema más complejo que, por contra, ofrece una mayor grantía de funcionamiento.

La base fundamental de las técnicas de tolerancia a fallos es la redundancia. Así pues, un sistema que la incorpore contendrá más componentes que otro sin replicación. Consecuentemente, si la tasa de fallos de los componentes es la misma para ambos sistemas, como así suele suceder, la probabilidad de que ocurra un fallo en un sistema tolerante será siempre mayor. El diseño de sistemas tolerantes a fallos debe mantener el equilibrio preciso entre el aumento de complejidad de los sistemas, con el consiguiente incremento en la tasa de fallos, y la detección y/o eliminación de éstos para que no lleguen a producir averías, y se consiga una garantía de funcionamiento suficientemente elevada.

A fin de mantener el compromiso antes planteado entre complejidad y eficacia, se hace necesaria la adopción de una buena metodología de diseño, aplicable de forma sencilla y efectiva. Partiendo de las características modulares de los sistemas informáticos su define una metodología basada en las relaciones jerárquicas entre sistemas y subsistemas componentes que se describe a continuación.

El funcionamiento básico de un sistema, conocido a través de su interfaz con otros de su nivel o con el de nivel superior que lo incluye, consiste en admitir peticiones para realizar servicios de su competencia y enviar las subsiguientes respuestas. Para satisfacer las peticiones recibidas, y siempre de acuerdo con su propio diseño, el sistema enviará a su vez peticiones a los susbsistemas que lo forman y obtendrá de ellos las respuestas que le permitan completar las peticiones externas recibidas. Desde un punto de vista suficientemente abstracto, este proceso describe el funcionamiento normal de cualquier sistema, incluyendo las relaciones hacia arriba de su interfaz y hacia abajo, al nivel inferior de la jerarquía.

En la figura 2 se esquematiza el funcionamiento de un sistema idealizado que incorpore tolerancia a fallos. La idea de redundancia aparece en la división en dos mitades del cuerpo del sistema, una de ellas dedicada al funcionamiento normal y la otra responsable de la gestión de errores. Siguiendo la literatura tradicional, a cualquier actividad relacionada con la presencia de fallos se la considera funcionamiento excepcional por contraste con lo que sería funcionamiento normal del sistema, empleando el término excepción para designar tanto al objeto de la gestión como a los mensajes de entrada o salida que involucra. Dado que un sistema es conocido únicamente por su comportamiento en la interfaz, las respuestas normales constituyen el funcionamiento esperado, libre de errores del sistema.

Las fuentes de errores con que debe tratar el sistema son tres, y su comportamiento ante alguna de ellas puede diferir según el éxito de los mecanismos de gestión de errores que incorpore. En primer lugar, el sistema puede recibir peticiones incorrectas, o que no está diseñado (o preparado) para servir. En este caso se generaría una excepción de la interfaz, indicando al nivel superior de la jerarquía la imposibilidad de satisfacer la petición. No tiene sentido pretender que el sistema gestione este error por sí mismo, dado que la petición es responsabilidad

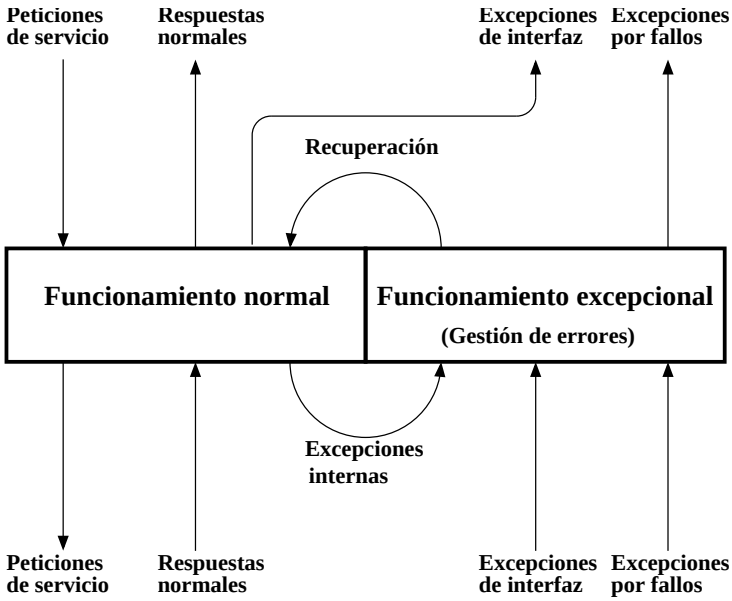

Figura 2: Metodología de diseño de los sistemas tolerantes a fallos.

de un nivel superior.

En segundo lugar el sistema puede detectar un error interno, y generar una excepción interna por tal causa. Si la excepción se puede tratar en el propio sistema, se produciría una recuperación, volviendo a la actividad normal. En caso contrario el sistema comunicaría mediante una excepción por fallo al nivel superior la presencia de una excepción interna que no ha podido ser corregida, delegando en aquél la responsabilidad de lograr la recuperación y vuelta a la actividad normal.

En tercer lugar, el sistema puede recibir de sus subsistemas componentes los dos tipos de excepciones que hemos visto es capaz de transmitir a un nivel superior: excepciones de interfaz por peticiones que no pueden ser satisfechas y excepciones por fallos en los componentes. Como en el caso anterior, el sistema puede bien recuperarse gestionando convenientemente estas excepciones, bien propagarlas al nivel superior de la jerarquía como excepciones por fallos.

La estructuración de un sistema completo en niveles o capas ordenados jerárquicamente es, como se ha visto, la base de la metodología de diseño expuesta. Cuando un subsistema no es capaz de gestionar un determinado error, genera una excepción que será tratada por algún nivel superior en la jerarquía. Siguiendo esta filosofía un diseño debe garantizar que todos los errores posibles sean detectados y tratados en algún nivel (no tiene por qué ser el mismo nivel el que detecte un error y el que lleve a cabo su tratamiento) evitando que exista algún punto único de avería (single point of failure) que comprometa el buen funcionamiento del sistema. Se emplea el término **cobertura** de un sistema para indicar el porcentaje de errores que es capaz de detectar.

1.3.1. Escenarios de aplicación de las técnicas.

Para la aplicación de la metodología se definen una serie de etapas ordenadas o escenarios en que aplicar las técnicas de tolerancia a fallos para que su actuación sea eficiente. Estas técnicas son aplicables ya sea completamente en diferentes capas del sistema, ya sea, siguiendo el caso más común, distribuyéndose a lo largo de los distintos niveles. Estas etapas serían:

- **Enmascaramiento o detección de errores:** Previamente a la realización de cualquier actividad encaminada a gestionar los errores en un sistema, éste debe tener evidencia de la existencia de aquéllos. El sistema debe, primeramente, detectar los errores. Una alternativa es el enmascaramiento. El sistema (a un cierto nivel) no detecta los errores sino que los enmascara en caso de que se produzcan. Un ejemplo de enmascaramiento lo constituiría un sistema de memoria con código de protección y corrección automática de errores.
- **Confinamiento y diagnóstico de errores:** Antes de emprender cualquier actividad correctora, el sistema debe evitar en lo posible la propagación del error detectado. La forma más normal de lograrlo es aplicar

rápidamente las medidas correctoras necesarias, disminuyendo así la latencia (tiempo transcurrido desde la aparición hasta la corrección) del error. Otras técnicas sin embargo incluyen el aislamiento efectivo del subsistema averiado evitando así la extensión del error fuera de aquél. Estas últimas técnicas proceden posteriormente a determinar la causa y/o extensión del error en el sistema aislado, mediante el proceso de diagnóstico. Según el resultado de éste se deciden las medidas correctoras a llevar a cabo.

- **Recuperación del sistema:** Una vez el error ha sido detectado, confinado y diagnosticado (en el caso en que todas estas etapas sean necesarias) el sistema debe recuperarse de las consecuencias de aquél. Si el error es permanente, el sistema deberá entrar en una fase de reconfiguración, reduciendo su garantía ed funcionaiento y en algunos casos funcionalidad o prestaciones (degradación del sistema). Si el error ha sido transitorio (o intermitente en ciertos casos) el sistema deberá garantizar la correcta ejecución de la actividad que estaba llevando a cabo. En algunos casos, tras errores permanentes, el sistema deberá realizar ambos procesos de reconfiguración y eliminación de las consecuencias de los errores sobre los cálculos.
- **Reparación y mantenimiento:** Un sistema con una elevada garantía de funcionamiento debe seguir una buena política de mantenimiento (en el caso en que sea posible) para evitar la vulnerabilidad que supone el desgaste y envejecimiento de los componentes. Este mantenimiento debe incluir, por supuesto, reparación de componentes averiados, revisión de los funcionales y estadísticas al respecto de tasas dinámicas de fallos cuando así sea posible. De este modo se pueden detectar errores intermitentes (por ejemplo, aquellos componentes cuya tasa de errores aparentemente transitorios supere un cierto umbral pueden marcarse como averiados por presencia de errores intermitentes) y, en cualquier caso, reemplazar los componentes menos fiables del sistema.

1.3.2. Cobertura de errores.

Como se ha visto, la **cobertura** de fallos es un parámetro fundamental en los sistemas tolerantes a fallos. En el caso ideal, con una cobertura del 100 %, el sistema sería completamente inmune. Como este porcentaje nunca se puede alcanzar en la práctica, el diseño de cualquier sistema tolerante a fallos debe comenzar con un estudio de los fallos que se quieren soportar, lo que se llama previsión o anticipación de fallos. Las técnicas de tolerancia a fallos se diseñarán luego para tolerar el 100 % de los fallos anticipados –o aquéllos que se quiera tolerar, no preocupándose específicamente por los modos de fallos cuya cobertura no es responsabilidad del sistema.

2. Técnicas de tolerancia a fallos.

La base fundamental de las técnicas de tolerancia a fallos es la redundancia, de la que se identifican hasta cuatro formas distintas de aplicación en los sistemas tolerantes a fallos:

- **Redundancia física:** se provee al sistema con más dispositivos físicos que los estrictamente necesarios para realizar su función en ausencia de fallos. Estos dispositivos adicionales servirán para la detección o enmascaramiento de aquéllos.
- **Redundancia lógica:** se incluyen varios módulos de código que realizan de forma diferente el mismo cómputo, para conseguir tolerancia frente a errores en su diseño.
- **Redundancia de Información:** la representación de los datos con que trabaja el sistema se extiende para dar cabida a información adicional que permita verificar su validez y, en algunos casos, corregir errores. El ejemplo más representativo es el de los códigos detectores y correctores de errores.
- **Redundancia Temporal:** los procesos a llevar a cabo por el sistema dedican más tiempo del estrictamente necesario según su función, con el fin de realizar acciones encaminadas a la detección y corrección de errores.

Estos tipos de redundancia pueden darse combinados (de hecho esto es lo más común) y es, en algunos casos, difícil distinguir a qué tipo pertenecen algunas técnicas. Los partados sigientes van a presentar las más comúnmente utilizadas.

2.1. Técnicas en el soporte físico.

El hardware de los sistemas digitales constituye la parte física de los mismos, donde se generan y propagan las señales eléctricas que permiten su funcionamiento. Por ello es la parte de los sistemas susceptible a fallos físicos, ya sean transitorios por causas esporádicas o permanentes o intermitentes por envejecimiento o desgaste de los componentes. Las técnicas hardware están orientadas principalmente a tolerar fallos de operación de esta índole, suficientemente estudiados y tipificados de manera que los modos de fallo son previsibles, por lo que las técnicas se pueden aplicar con efectividad. Los fallos de diseño, implementación o instalación deberían tratarse y eliminarse mediante verificación suficiente de los sistemas y mantenimiento de los mismos, siendo en general imprevisibles y, por lo tanto, difíciles de evitar aplicando algunas de las técnicas que se describen.

Estas técnicas aplicadas en el hardware para obtener tolerancia a fallos se han clasificado, atendiendo al comportamiento frente a los fallos del subsistema que las incorpora, en tres grupos:

- **Redundancia pasiva** (*passive redundancy*): El sistema proporciona tolerancia a fallos sin ninguna acción de su parte frente a su presencia. Los fallos simplemente se enmascaran, evitando la propagación de estados erróneos fuera del subsistema tolerante. Suelen ser sistemas de altas prestaciones y elevado coste.

- **Redundancia activa** (*active redundancy*): El sistema detecta los fallos y se reconfigura para seguir ofreciendo la misma funcionalidad mediante el aislamiento lógico y/o eléctrico de los componentes averiados. Este tipo de redundancia, también llamada dinámica, recorre las tres primeras etapas de tratamiento de fallos vistas con anterioridad, por lo que en general requiere un cierto tiempo para reanudar su funcionamiento normal tras la recuperación. Como contrapartida, permite elegir un compromiso entre tiempo de reconfiguración y coste o complejidad del sistema, según el tipo de aplicación.

- **Redundancia híbrida** (*hybrid redundancy*): Combina las características de las anteriores, para ofrecer unas mejores características. La problemática de la redundancia pasiva es la progresiva pérdida de fiabilidad a medida que los componentes van quedando averiados. Por otra parte, el enmascaramiento de fallos no requiere de reconfiguración, por lo que en general se obtienen módulos con mejores prestaciones. La redundancia híbrida proporciona enmascaramiento de fallos, al tiempo que aislamiento de los componentes averiados sin incurrir en retardos por reconfiguración significativos, manteniendo la fiabilidad y las prestaciones del sistema a costa de una mayor provisión de componentes y, por consiguiente, de un coste más elevado.

2.2. Técnicas en el soporte lógico.

La característica fundamental del software, por contraposición a lo dicho en la introducción a las técnicas hardware, es su inmunidad ante fallos de operación. Para ser más estrictos, teniendo el software únicamente existencia lógica (aunque necesite de medios físicos para ser almacenado, transportado y ejecutado) no es susceptible de más errores que los de diseño. Por lo tanto, las técnicas de replicación tal y como se han visto antes carecen de aplicabilidad en el ámbito del software, debiendo encontrarse soluciones completamente distintas. Las técnicas deben tratar con errores de diseño, con manifestaciones difíciles de prever. En la detección de estos errores se emplean fundamentalmente dos mecanismos: la realización de pruebas de validez para los resultados obtenidos mediante una determinada computación, que presupone un cierto conocimiento a priori acerca de los mismos, y la comparación de los resultados obtenidos al realizar el cómputo empleando una serie de programas distintos.

El segundo mecanismo es la base de la técnica llamada **programación de N versiones** (*N-Version Programming*) en la que se comparan los resultados obtenidos por N programas distintos que realizan los mismos cálculos, eligiendo como válida la respuesta dada por la mayoría. Los objetos de estudio fundamentales que tratan de optimizar esta técnica tienen en cuenta la forma de obtener N versiones diferentes para realizar los mismos cálculos, cómo realizar la votación incluyendo ciertos márgenes de discrepancia (no siempre son aplicables las meras comparaciones de igualdad) y la protección del mecanismo de votación en sí.

Otra técnica empleada habitualmente es la utilización de bloques de recuperación (recovery blocks). El programa se divide en bloques de ejecución atómica, al final de cada uno de los cuales se lleva a cabo una prueba de validez de los resultados. Si éstos se consideran satisfactorios, el programa progresa hacia el siguiente bloque. En caso contrario, se vuelve al estado previo a la ejecución del último bloque que falló, y se repiten los cálculos utilizando una versión distinta del código. Como en el caso anterior, cuantas más versiones se tengan del software, mayor será el número de reintentos que se podrán llevar a cabo antes de que un programa fracase totalmente debido a un error de diseño. Esta técnica ofrece mejores prestaciones, pues no es necesario repetir los

cómputos si el programa se ha ejecutado correctamente, pero presenta el inconveniente de la cobertura de las pruebas de validez, siempre menor que la comparación de resultados que deberían ser idénticos. Otro aspecto muy importante de esta técnica es la incorporación del concepto de vuelta atrás (*rollback*) a un estado previo en la historia de la ejecución de un cierto programa.

2.3. Técnicas a nivel de sistema.

Como se ha explicado anteriormente, la metodología de diseño de sistemas tolerantes a fallos distingue típicamente diferentes fases o escenarios en que el sistema debe actuar de forma específica para evitar y prevenir su mal funcionamiento o avería. Una de estas fases es la de recuperación de errores, es decir, la vuelta del sistema a un estado correcto tras la aparición de algún error durante su funcionamiento normal. En teoría esta recuperación puede llevarse a cabo hacia adelante (*forward error recovery*) o hacia atrás (*backward error recovery*), siendo este último procedimiento (figura 3) el que se aplica en la mayor parte de los casos.

El principio de funcionamiento de la recuperación de errores hacía atrás es simple: en caso de detectarse un error el sistema vuelve a un estado anterior, libre de errores, retomando su evolución desde este estado que llamaremos consistente. La aplicación de esta técnica de recuperación, sin embargo, requiere del sistema una serie de características que le añaden un mayor coste y complejidad:

- Debe estar provisto de los mecanismos necesarios de detección de errores.
- Debe ser capaz de reconocer estados libres de error.
- Debe estar dotado de capacidad para volver a uno de estos estados tras la detección de un error, y continuar el funcionamiento normal desde él.

El primer requisito se puede conseguir con una buena parte de las técnicas hardware vistas anteriormente, y es fundamental en todo sistema tolerante a fallos. El segundo se satisface como consecuencia directa del primero: se puede considerar un estado libre de errores cualquiera al que el sistema haya llegado sin haber detectado ningún error. La garantía de ausencia de errores en el estado alcanzado depende de la cobertura de los mecanismos de detección, pero es el único criterio que se puede aplicar al respecto para un sistema dado.

Por último, si bien existen diversas formas para conseguir que un sistema sea capaz de volver a un estado anterior, la más empleada es el establecimiento de puntos de recuperación (*checkpoints*): el sistema, en un instante determinado, almacena la información suficiente sobre su estado actual. En caso de detectarse un error, esta información es utilizada para llevar el sistema al estado que tenía cuando la guardó. El sistema continúa su evolución desde el estado recuperado. Aunque es teóricamente posible disponer de un numero arbitrario de puntos de recuperación, correspondientes a sucesivos estados por los que ha transcurrido el sistema, en la mayor parte de los casos cada punto de recuperación anula el anterior, por lo que en cada momento existe en el sistema un único punto de recuperación correspondiente al último estado libre de errores que se guardó. Así pues, en caso de detectarse algún error, el sistema debe estar dotado de los mecanismos que permitan llevar a cabo una vuelta atrás, es decir, utilizar la información almacenada en el último punto de recuperación establecido (también llamada frecuentemente copia de seguridad del estado del sistema) para retomar el funcionamiento del sistema desde aquel punto, ignorando los posibles errores producidos desde el instante recuperado.

Una alternativa al establecimiento de puntos de recuperación consiste en mantener un **registro** (*log, journal*) en el que se van anotando las acciones que va realizando el sistema, de forma que en caso de desencadenarse una vuelta atrás estas se puedan deshacer en orden inverso al que se han llevado a cabo. Esta técnica se puede combinar con la anterior para establecer puntos de recuperación con menor frecuencia, manteniendo un registro entre ellos.

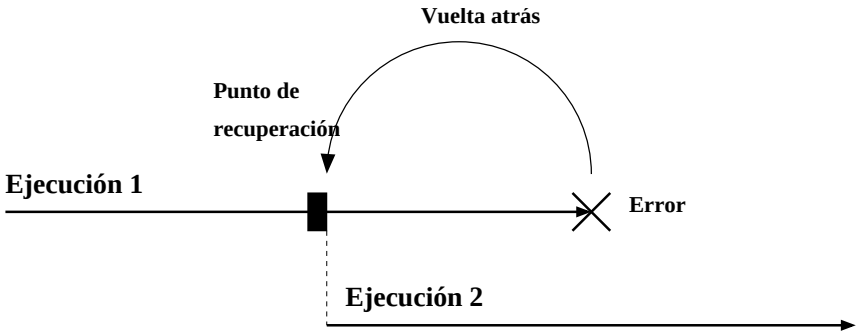

Figura 3: Recuperación mediante vuelta atrás.