Relatório Técnico

Mário ALVES, Francisco VASQUES

CONFIANÇA NO FUNCIONAMENTO
CONCEITOS BÁSICOS E TERMINOLOGIA

1998

Tradução Comentada de

Jean-Claude LAPRIE (ed.)

DEPENDABILITY
BASIC CONCEPTS AND TERMINOLOGY

in English, French, German, Italian and Japanese

1992

Mário Alves

Departamento de Engenharia Electrotécnica

Instituto Superior de Engenharia do Porto

Rua de São Tomé

4200 Porto

Portugal

Tlf: +351.22.8340500

Fax: +351.22.8321159

malves@dee.isep.ipp.pt

http://www.hurray.isep.ipp.pt

Francisco Vasques

Departamento de Engenharia Mecânica e Gestão Industrial

Faculdade de Engenharia da Universidade do Porto

Rua Dr. Roberto Frias

4200 Porto

Portugal

Tlf: +351.22.5081702

Fax: +351.22.5081440

vasques@fe.up.pt

http://www.fe.up.pt/ ~ vasques

PREFÁCIO DA VERSÃO PORTUGUESA

Este documento apresenta uma tradução comentada de [Lap, 92], um livro que define as bases da área da *confiança no funcionamento*, tornando-se uma referência internacional dos conceitos e terminologias adoptados neste domínio científico. O livro está escrito em Inglês, Francês, Alemão, Italiano e Japonês, dispondo de um glossário de termos em cada uma das línguas e um índice multilingue (uma tabela com os termos nas cinco línguas). Justifica-se portanto a inclusão de um glossário de termos em Português e um índice bilingue (Português – Inglês).

Os comentários justificam-se nas situações em que se definem termos em Português que não são exactamente a tradução "à letra" do Inglês, bem como para tentar clarificar o leitor de certos conceitos, por vezes recorrendo a exemplos de aplicação.

Para enriquecer esta tradução, foi consultada a bibliografia indicada em 'Referências da Versão Portuguesa' e ainda [Lap, 90a]. É feita uma tentativa de comparar os conceitos defendidos neste texto com as ideias apresentadas pelo autor, ao longo dos últimos anos, antes ([Lap, 90a]) e depois ([Lap, 93], [Lap, 95a], [Lap, 95b], [Lap, 98]) de 1992, de modo a poder apresentar a evolução dos conceitos do próprio autor. A consulta de [Mag, 95] e [Ver, 89] foi fundamental, pois permitiu analisar as definições, conceitos e termos que já tinham sido definidos em Português. Sempre que relevante, são apresentadas as ideias de outros autores, alguns deles baseando-se no trabalho alvo de tradução para evoluir os seus próprios conceitos.

Entendeu-se importante o desenvolvimento deste trabalho pois a tendência actual é para que a área da *confiança no funcionamento* de sistemas se torne cada vez mais importante, senão fundamental, nomeadamente nos sistemas de controlo computadorizado. Há alguns anos a esta parte, apenas as aplicações de segurança crítica, isto é, cujas avarias podem ter consequências catastróficas para pessoas e bens (os sistemas de controlo de navegação, nos aviões e os sistemas de controlo da sinalização, no tráfego ferroviário são exemplos típicos dessas aplicações) eram alvo de concepções que contemplavam a *confiança no funcionamento*.

Isto está obviamente a mudar, dado que a nossa dependência em sistemas de controlo baseados em computadores é cada vez maior. Basta dizer que não só grande parte das infra-estruturas dos países (a produção, transporte e distribuição de energia eléctrica, os transportes, os sistemas de comunicação e os sistemas de informação bancária, por exemplo), como também pequenos sistemas, como também sistemas de menor dimensão/custo, muitas vezes utilizados por particulares (aplicações em domótica, máquinas e equipamentos industriais e os sistemas de segurança activa e passiva nos automóveis, por exemplo) dependem cada vez mais de sistemas computacionais. Mais ainda, a procura de sistemas "de confiança" é muitas vezes o resultado de um acidente ou conjunto de acidentes, que sublimam a importância da *confiança no funcionamento*. Como se costuma dizer: "casa roubada, trancas à porta".

Ora, se nós precisamos de depositar maior confiança nesses sistemas, eles devem ser concebidos tendo em conta a *confiança no funcionamento*. Estamos nitidamente a assistir a um fenómeno tipo "bola de neve", dado que se é um facto que depositamos cada vez mais confiança nos sistemas controlados por computador, começam a conceber-se sistemas tendo em conta a confiança no funcionamento, reduzindo o seu custo, o que tem como consequência a vulgarização da sua utilização, e por aí adiante.

Agradecemos a colaboração dos outros colegas do Grupo IPP HURRAY! (HUgging Real-time and Reliable Architectures for computing sYstems), do Instituto Superior de Engenharia do Porto.

PREFÁCIO DA VERSÃO ORIGINAL

Este volume é a compilação de um texto resultante de vários anos de trabalho, no âmbito do "Working Group 10.4 of IFIP[1], *Dependable Computing and Fault Tolerance*", juntamente com as suas versões Francesa, Alemã, Italiana e Japonesa. O objectivo é fornecer definições precisas que caracterizem a confiança no funcionamento dos sistemas computacionais[2]. A confiança no funcionamento é inicialmente introduzida como um conceito genérico e é fornecido um conjunto de definições básicas. Estas definições são então comentadas e acrescentadas nas secções subsequentes, abordando respectivamente os impedimentos à confiança no funcionamento (falhas, erros, avarias), os meios para a obtenção de confiança no funcionamento (prevenção de falhas, tolerância a falhas, supressão de falhas, previsão de falhas) e os atributos da confiança no funcionamento (fiabilidade, disponibilidade, segurança e inviolabilidade). As 116 definições fornecidas ao longo do texto são recapituladas em cinco glossários[3] e está também disponível uma tabela de termos nas cinco línguas referidas[4].

O que é apresentado não existiria sem as inúmeras discussões tidas com muitos colegas. O papel proeminente dos contribuintes listados na página de título é reconhecido, quer através das contribuições escritas e comentários, quer pela sua ajuda no desenvolvimento de ideias durante as discussões. É um prazer estender estes agradecimentos a todos os membros do WG 10.4, especialmente a David Morgan, pelo seu papel dinamizador, e a todos os membros do "*Dependable Computing and Fault Tolerance*" do LAAS[5], dos quaisJean Arlat, Christian Beounes, Yves Deswarte, Jean-Charles Fabre, David Powell e Pascale Thevenod merecem uma atenção especial. De particular importância é o trabalho desenvolvido pelos meus colegas que gentilmente aceitaram produzir as versões aqui reunidas: Yoshiaki Koga produziu a versão Japonesa, Luca Simoncini produziu a versão Italiana, Udo Vogues produziu a versão Alemã com a ajuda de Winfried Görke e Hermann Kopetz. Fui grandemente ajudado pelo Alain Costes na produção da versão Francesa, e a sua infindável disponibilidade para discutir este tópico ao longo dos anos merece uma atenção especial. Um agradecimento particular vai também para Joëlle Penavayre pela sua ajuda na tarefa de formatação.

Toulouse, Julho de 1991

Jean-Claude Laprie

---

[1] IFIP é o acrónimo de *International Federation for Information Processing*.

[2] Tradução de "*computing systems*".

[3] Neste documento existe apenas um glossário em Português.

[4] Neste documento a tabela apenas contempla termos em Português e em Inglês.

[5] LAAS é o acrónimo de *Laboratoire d'Analyse et d'Architecture des Systemes*.

ÍNDICE

INTRODUÇÃO

Este documento tem como objectivo fornecer definições informais mas precisas de modo a caracterizar os vários atributos da confiança no funcionamento dos sistemas computacionais. É uma contribuição para o trabalho desenvolvido no âmbito da comunidade científica e técnica dos "Sistemas Computacionais Fiáveis e Tolerantes a Falhas"[6] ([Avi, 67], [Jes, 77], [Mel, 77], [Avi, 78], [Ran, 78], [Car, 79], [And, 81], [FTC, 82], [Sie, 82], [Cri, 85a], [Lap, 85], [Avi, 86], [Lap, 89]) de modo a propor definições claras e aceites por muitos para alguns conceitos básicos.

A confiança no funcionamento é primeiro introduzida como um conceito global que engloba os atributos usuais de fiabilidade, disponibilidade, segurança e inviolabilidade. As definições básicas que surgem na primeira secção são então comentadas e suplementadas através de definições adicionais, nas secções subsequentes. O anexo inclui um glossário onde se recapitulam as definições dadas ao longo do documento. A apresentação foi estruturada de forma a evitar referenciação para a frente. Caracteres a negrito são utilizados quando da definição de um termo, sendo os caracteres itálicos um convite para captar a atenção do leitor. As linhas de orientação que regeram esta apresentação podem ser sumariadas como se segue:

- procura de um reduzido número de conceitos que permitam que os atributos da confiança no funcionamento possam ser expressos;
- utilização de termos que sejam idênticos - sempre que possível - ou tão próximos quanto possível daqueles normalmente usados;
    - dar ênfase à integração ([Gol, 82], [Ran, 86]) (em oposição a especialização) através da independência das definições dadas no que respeita às classes de falhas.

Este documento pode ser visto como tendo um consenso mínimo dentro da comunidade, de modo a facilitar interacções frutuosas; adicionalmente, espera-se que este documento seja adequado para: a) utilização por outros organismos (incluindo organizações de normalização) e b) fins pedagógicos. Deste ponto de vista, o esforço de terminologia associado não é um fim por si próprio: as palavras só têm interesse quando identificam univocamente conceitos e permitem a partilha de ideias e pontos de vista. Este documento não tem qualquer pretensão de ser um estado da arte ou as "Tábuas da Lei": os conceitos que são apresentados têm de evoluir com a tecnologia, bem como com o nosso avanço na compreensão e domínio da especificação, projecto, implementação e avaliação dos sistemas computacionais em cujo funcionamento se deposita confiança.

O que é exposto neste documento não existiria sem as inúmeras discussões tidas com muitos colegas, especialmente os membros do IFIP WG 10.4.

[6] Tradução de "*Reliable and Fault Tolerant Computing*".

1. DEFINIÇÕES DE BASE

A **confiança no funcionamento**[7] de um sistema computacional é a propriedade desse sistema que permite que um seu utilizador possa depositar uma confiança justificada no serviço que ele presta[8] ([Car, 82]). O **serviço** prestado por um sistema é o seu comportamento *tal como percebido* pelo seu utilizador(es). Um **utilizador** é um outro sistema (humano ou físico) que *interage* com o primeiro.

Dependendo da aplicação(ões) pretendida para o sistema, podem ser enfatizadas diferentes facetas da confiança no funcionamento, i.e., a confiança no funcionamento pode ser vista de acordo com propriedades diferentes mas complementares, que permitem definir os *atributos* da confiança no funcionamento:

- relativamente à capacidade para estar *pronto a utilizar*, confiança no funcionamento significa **disponibilidade**;
- no que respeita à *continuidade do serviço* prestado, confiança no funcionamento significa **fiabilidade**;
- no que respeita à *não ocorrência de avarias catastróficas* para o meio envolvente, confiança no funcionamento significa **segurança**[9];
- no que concerne à *prevenção de acesso ou manipulação de informação não autorizados*, confiança no funcionamento significa **inviolabilidade**[10].

A **avaria** de um sistema ocorre quando o serviço prestado deixa de estar conforme a **especificação**[11], que é uma descrição acordada da função e/ou

[7] Tradução de "*dependability*". A tradução segundo [Por, 85] é *confiança*, *segurança*. Segundo [Lon, 95], alguém ou algo "*dependable*" é alguém ou algo em que se pode confiar para fazer aquilo que se necessita ou espera que faça. Adopta-se portanto o termo *confiança de funcionamento*, tal como nas escolas de Coimbra ([Mag, 95]) e de Lisboa ([Ver, 89]) e a terminologia francesa "s*ûreté de fonctionnement*", adoptada no documento original ([Lap, 92]). Para "*dependable*", utilizar-se-à o termo "de confiança", pois "confiável", no Brasil, significa "fiável", o que poderia dar aso a confusões. É interessante focar o que ([Lap, 95a], [Lap, 98]) define como o objectivo último da confiança no funcionamento: "especificar, projectar, implementar e explorar sistemas onde a falha é natural, prevista e tolerável".

[8] Notar que um serviço, para ser bem sucedido, tem não só que produzir valores lógicos correctos (domínio do valor), mas também de os disponibilizar dentro de um determinado período de tempo (domínio do tempo).

[9] [Mag, 95] traduz "*safety*" para "segurança contra falhas acidentais". Segundo a definição contida neste texto, "*safety*" representa a confiança no funcionamento relativamente à não ocorrência de avarias catastróficas. Ora, uma avaria catastrófica pode ocorrer devido a uma falha intencional, pelo que a tradução de "*safety*" dada por [Mag, 95] não parece a mais correcta. Melhor é a tradução de [Ver, 89], que utiliza "segurança de funcionamento" para "*safety*". De forma análoga, tal como refere [Lap, 93], e bem, "*security*" não se pode limitar a evitar falhas intencionais, pois uma falha acidental pode causar uma fuga inesperada de informação. Isto quer dizer que a tradução "segurança contra falhas intencionais" para "*security*" defendida quer por [Mag, 95], quer por [Ver, 89], não parece muito feliz. Tendo em vista evitar a utilização de termos compostos e dada a riqueza da Língua Portuguesa, nesta tradução adoptaram-se os termos "segurança" para "*safety*" e "inviolabilidade" para "*security*".

[10] Em artigos posteriores ([Lap, 93], [Lap, 95a], [Lap, 95b], [Lap, 98]), o autor deixa de utilizar o termo *inviolabilidade* como um atributo simples de confiança no funcionamento, de acordo com as sua definição usual, passando a considerá-la como a combinação de *confidencialidade*, a prevenção de divulgação não autorizada de informação, de *integridade*, prevenção de modificação ou supressão não autorizados de informação e de disponibilidade, prevenção de retenção não autorizada de informação (no sentido de que um intruso pode impedir (de uma forma não autorizada) o acesso/manipulação de informação a um utilizador (autorizado)). Merece particular destaque o esquema feito em [Lap, 93] que representa a relação entre os atributos da confiança no funcionamento.

[11] O autor evoluiu o conceito de avaria ([Lap, 93], [Lap, 95a], [Lap, 95b], [Lap, 98]) referindo que ocorre uma avaria num sistema quando o serviço prestado se desvia do cumprimento da *função* do sistema, isto é, àquilo a que se destina o sistema, ou seja a ocorrência de uma avaria é definida *em relação à função de um sistema e não à*

serviço esperado do sistema. Um **erro** é aquela parte do estado de um sistema que pode levar a uma avaria: um erro afectando o serviço é uma indicação de que ocorreu ou está a ocorrer uma avaria. A causa hipotética ou julgada de um erro é uma **falha**.

A concepção de um sistema computacional em cujo funcionamento se pode confiar apela à utilização combinada de um conjunto de métodos, que podem ser classificados da seguinte forma:

- **Prevenção de falhas**: como evitar a ocorrência ou introdução de falhas;
- **Tolerância a falhas**: como fornecer um serviço de acordo com a especificação, apesar da ocorrência de falhas;
- **Supressão de falhas**[12]: como reduzir a presença (número, gravidade) de falhas;
- **Previsão de falhas**: como estimar o corrente número, a incidência futura e as consequências das falhas.

A prevenção de falhas e a tolerância a falhas podem ser vistas como **meios para a obtenção** de confiança no funcionamento: como *dotar* o sistema com a capacidade de prestar um serviço de acordo com a especificação; a supressão de falhas e a previsão de falhas podem ser vistas como meios para a **validação** da confiança no funcionamento: como *alcançar confiança* na capacidade do sistema para prestar um serviço de acordo com a especificação.

A confiança no serviço prestado por um sistema e a justificação para essa confiança são (ou devem ser) baseadas na *avaliação* do sistema, primariamente direccionada para os atributos da confiança no funcionamento que são pertinentes para os serviços prestados pelo sistema.

As noções introduzidas até agora podem ser agrupadas em três classes (*Figura 1*):

- os **impedimentos**[13] à confiança no funcionamento: falhas, erros, avarias; são situações indesejáveis – mas não imprevisíveis, em princípio – causadas ou resultantes de uma desconfiança no funcionamento (cuja definição deriva simplesmente da definição de confiança no funcionamento: não se pode, ou não se poderá, a partir de um dado momento, depositar confiança no serviço prestado);
- os **meios** para a confiança no funcionamento: prevenção de falhas, tolerância a falhas, supressão de falhas e previsão de falhas; estes são os métodos e técnicas que permitem a) fornecer a capacidade de prestar um serviço em que pode ser depositada confiança e b) atingir confiança nesta capacidade;
- os **atributos** da confiança no funcionamento: disponibilidade, fiabilidade, segurança, inviolabilidade; estes a) permitem expressar as propriedades[14] que se esperam do sistema e b) permitem avaliar a qualidade do sistema que resulta dos impedimentos à confiança no funcionamento e dos meios para os ultrapassar.

---

sua especificação. De facto, se bem que um comportamento inaceitável seja normalmente identificado como uma avaria devido a um desvio da conformidade com a especificação, é possível que esse comportamento satisfaça a especificação, mas seja inaceitável para os utilizadores do sistema, revelando-se assim uma falha de concepção (especificação). Esta evolução do conceito de avaria parece mais correcta que a dada neste texto.

[12] Preferir-se-ia o termo "eliminação de falhas", por ser mais vulgar, mas tem-se em conta que outros autores portugueses e brasileiros já utilizam "supressão de falhas".

[13] Preferir-se-iam os termos "obstáculos" ou "entraves", por serem mais vulgares, mas tem-se em conta que outros autores portugueses e brasileiros já utilizam o termo "impedimentos".

[14] Propriedades que, neste contexto, têm a ver com a confiança no funcionamento do sistema, tentando quantificar a confiança no funcionamento do sistema.

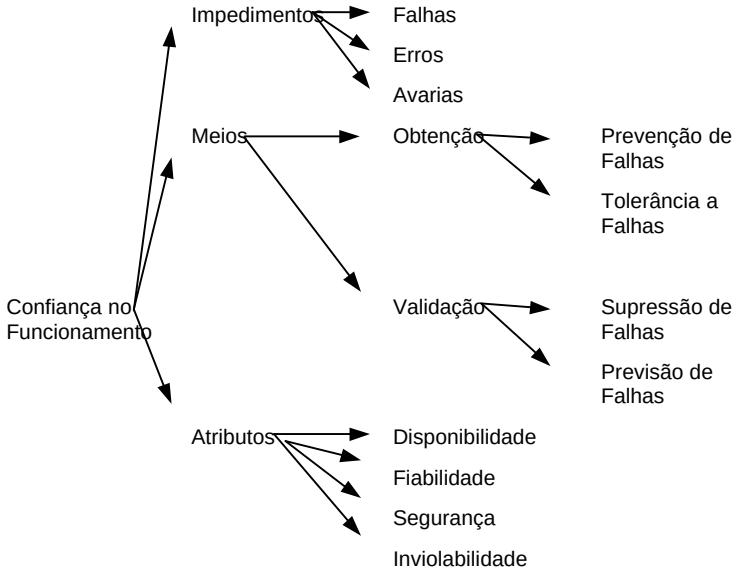

Figura 2 – A árvore da confiança no funcionamento

2. INTRODUZINDO A CONFIANÇA NO FUNCIONAMENTO COMO UM CONCEITO GENÉRICO

Uma tendência natural em qualquer domínio[15] técnico ou científico emergente é, num primeiro passo, restringir o seu campo de investigação de forma a fazer (rápidos) progressos na resolução dos problemas associados. Chega então uma altura em que as suas interacções com outros domínios deixam de poder ser ignoradas. Uma grande tentação é considerar estes outros domínios como "casos especiais" do domínio particular. Isto resulta normalmente em grandes debates, muitas vezes moderados pelos investigadores de cada domínio, cada um com a sua própria linguagem (palavras e expressões) técnica. Foi este caso da fiabilidade, disponibilidade, segurança e inviolabilidade dos sistemas computacionais. Inicialmente, a grande preocupação era dispor de sistemas computacionais que funcionassem: *fiabilidade*. À medida que estes sistemas se tornaram fiáveis, os seus serviços começaram a ser utilizados e necessários numa base regular, tornando a *disponibilidade* um factor essencial. A utilização de sistemas computacionais em aplicações críticas trouxe a preocupação de *segurança*. O sistema mais seguro é muitas vezes o sistema que não faz nada, o que não é muito útil; portanto, as pessoas que se preocupam com a segurança de sistemas tendem a considerar a fiabilidade como um subconjunto de segurança. O advento dos sistemas distribuídos exacerbou a *inviolabilidade*[16]. Novamente, um sistema inviolável é suposto cumprir funções; adicionalmente, as violações de um sistema podem ser catastróficas; portanto, as pessoas que se preocupam com a inviolabilidade tendem a considerar a fiabilidade e a segurança como subconjuntos da inviolabilidade.

---

[15] Tradução de "*discipline*".

[16] Tal como refere [Wei, 97], a inviolabilidade dos sistemas computacionais é cada vez mais importante, dado que grande parte das infra-estruturas dos países dependem de sistemas computacionais distribuídos, dependentes de software e frágeis. Exemplos de alvos potenciais do ataque de intrusos são a rede eléctrica, os transportes (principalmente os ferroviários e aéreos) e os sistemas bancários.

Contudo, as relações entre fiabilidade, segurança e inviolabilidade são mais complexas do que uma simples dependência. Consideremos o exemplo das chamadas "bombas de software"[17], i.e., falhas deliberadamente introduzidas num sistema computacional de modo a provocar, num momento escolhido pelo "terrorista" – e sob o seu controlo – uma avaria no sistema, de consequências preferivelmente sentidas pelo utilizador como não catastróficas (até que este se dê conta das causas da avaria). Este exemplo envolve claramente fiabilidade, segurança e inviolabilidade, de uma forma intrincada e variada, dependendo do ponto de vista considerado[18]. O que é certo é que o utilizador não pode, ou não deve, colocar confiança no serviço prestado por tal sistema, que não é de confiança.

A discussão anterior mostra claramente que *não* é intenção deste documento contribuir para a controvérsia no que respeita a fiabilidade ser um conceito mais vasto do que segurança ou vice-versa, e, de forma similar, quando se considera também a inviolabilidade. O que é essencial na inter-relação entre fiabilidade, segurança e inviolabilidade e a noção de confiança de funcionamento é que os três primeiros são *atributos* do último: como tal, espera-se um enriquecimento recíproco. Esta é a razão fundamental para a adição de um novo termo[19] a uma lista já extensa – fiabilidade, disponibilidade, segurança, inviolabilidade, etc. Uma razão adicional para a introdução de confiança no funcionamento como um conceito genérico é a vontade de aliviar a fiabilidade do seu significado etimológico global[20], de forma a concentrar-se no significado mundialmente aceite (e historicamente recente) de continuidade no serviço, no que respeita tanto ao seu significado genérico – sistema fiável – como também à sua definição probabilística – fiabilidade de um sistema[21].

Apesar de confiança no funcionamento[22] ser um sinónimo de fiabilidade, o primeiro tem uma conotação de dependência. À primeira vista pode sentir-se uma conotação negativa[23], quando confrontada com a noção positiva expressa pela fiabilidade, mas na realidade, releva a dependência da nossa sociedade em sistemas sofisticados, genericamente, e especialmente nos sistemas computacionais. Continuando com considerações etimológicas, "*to rely*" vem do Francês "*relier*", este derivado do Latim "*religare*" que significa unir, apertar. A palavra Francesa para fiabilidade, "*fiabilité*", estava já presente no século doze, no nome de "*fiableté*", cujo significado era "carácter digno de confiança"; a origem latina é o verbo "*fidare*", um verbo popular para "ter confiança". À luz destas considerações etimológicas, arrependemo-nos apenas que a definição de fiabilidade correntemente empregue em muitos campos da engenharia[24] tenha substituído a noção de "capacidade"[25] pela noção de "confiança", pelas duas razões seguintes (pelo menos):

---

[17] Tradução de "*softbombs*"

[18] (NA1) É também digno de nota que os eventos descritos na secção sobre "*Risk to the Public in Computer Systems*" da *ACM Software Engineering Notes* estão relacionados com fiabilidade, segurança e inviolabilidade.

[19] Refere-se a "*dependability*" (confiança no funcionamento).

[20] Alguém ou algo fiável ("*reliable*") é alguém ou algo em que se pode confiar.

[21] (NA2) É interessante notar que:

a) a maior parte dos livros que referem a palavra "fiabilidade" no seu título tratam, na realidade, de como avaliar, medir e prever a fiabilidade dos sistemas, não propriamente de como se constróem sistemas fiáveis;
b) olhando para a confiança no funcionamento como um conceito mais genérico que fiabilidade, disponibilidade, etc. e incorporando as suas noções foi já tentado no passado (ver [Hos, 60]); contudo, estas tentativas não foram feitas com a generalidade que são feitas aqui, já que então, o objectivo era definir uma medida que apenas incorporasse disponibilidade e fiabilidade, não considerando a inviolabilidade, por não ser de interesse na altura.

[22] "*Dependability*".

[23] Refere-se à conotação negativa da palavra "dependência".

[24] (NA3) Por exemplo, "Fiabilidade: a capacidade de um item executar a função pretendida para dadas condições, durante um dado intervalo de tempo" ([IEC, 85]).

a) do ponto de vista do utilizador do sistema, o que realmente tem interesse não é tanto a *capacidade* de fornecer funcionalidades, mas sim o *serviço* que é *realmente* prestado ao utilizador;

b) do ponto de vista de quem concebe o sistema, que está disposto a admitir a possibilidade da existência de falhas na sua concepção, a interpretação do termo "capacidade" deve ser questionável, apesar do facto de ter sido adoptado nos glossários de engenharia de *software* ([IEE, 82]).

[25] Tradução de "*ability*". Outros sinónimos ([Por, 85]) são "aptidão", "competência", "habilidade", "qualificação".

3. Acerca da Função, do Comportamento, da Estrutura, e da Especificação de um Sistema

Até este momento, um **sistema**[26] foi considerado, implicitamente, como um todo, dando ênfase ao seu comportamento percebido do exterior. A seguinte definição está de acordo com esta visão do sistema como uma "caixa preta": um sistema é uma entidade que interagiu ou interferiu, que está a interagir ou interferir, ou com probabilidade de interagir ou interferir com outras entidades, i.e., com outros sistemas. Estes outros sistemas constituíram, constituem ou vão constituir o **meio envolvente** do sistema considerado[27]. Um **utilizador** de um sistema é aquela parte do meio envolvente que *interage* com o sistema considerado: o utilizador fornece entradas ao sistema e/ou recebe as suas saídas; o que o distingue do resto do meio envolvente é o facto de *utilizar o serviço(s)* prestado pelo sistema[28].

A **função**[29] de um sistema é para o que *se destina* o sistema ([Kui, 85]). O **comportamento** de um sistema é o que o sistema *faz*. O que *lhe permite*

[26] É interessante analisar as definições de sistema dadas por [Kop, 93], [Mag, 95] e [Wei, 97].

[27] (NA4)

a) O facto de se utilizarem definições recursivas não é pelo simples prazer da recursividade. O objectivo é dar ênfase à relatividade no que respeita ao ponto de vista adoptado. É este o caso da noção de sistema: as fronteiras de um sistema poderão variar dependendo do se são vistas pelo seu projectista(s), pelo seu utilizador(s), pela sua equipa de manutenção, etc.

b) Os tempos verbais passado, presente e futuro são empregues de modo a sublinhar que o meio envolvente de um sistema pode variar no tempo, especialmente no que respeita às fases do seu ciclo de vida. Por exemplo, a noção de "ambiente de programação" enquadra-se na definição dada, bem como o ambiente físico com que um sistema é confrontado durante a sua vida operacional.

[28] Ver 43.

[29] É interessante ler a secção '2.1. Função e avaria' de [Lap, 93].

fazer o que faz é a sua **estrutura** ([Zie, 76]). Adoptando o espírito de [And, 81], um sistema, do ponto de vista estrutural (caixa de vidro), é um conjunto de componentes interligados com vista a interagir; um **componente** é outro sistema, etc. A recursividade pára quando um sistema é considerado como sendo **atómico**: qualquer outra estrutura interna não pode ser discernida, ou não se mostra relevante, podendo ser ignorada[30]. O termo "componente" tem de ser compreendido num sentido lato: as camadas de um sistema bem como as componentes entre camadas; adicionalmente, como um componente é, por si só, um sistema, ele engloba a inter-relação dos componentes que o constituem. Uma definição mais clássica da estrutura de um sistema é o que o sistema *é*. Tal definição enquadra-se na perfeição quando representamos um sistema sem ter explicitamente em consideração quaisquer impedimentos à confiança no funcionamento, e portanto no caso de onde a estrutura é considerada como *fixa*. Nós não queremos restringir-nos a sistemas cuja estrutura é considerada fixa. Em particular, necessitamos de permitir mudanças estruturais provocadas por ou resultantes de impedimentos à confiança no funcionamento. Uma estrutura poderá então ter estados[31]. Advém então uma definição de **estado**: um estado é uma situação[32] em que se está, no que respeita *a um conjunto de circunstâncias*. Esta noção aplica-se tanto ao comportamento de um sistema como à sua estrutura[33].

O serviço prestado por um sistema é, a partir da sua própria definição (o comportamento percebido pelo utilizador), uma *abstracção* do comportamento do sistema. Note-se que esta abstracção é altamente dependente da aplicação a que se destina o sistema computacional. Um exemplo desta dependência é o importante papel desempenhado pelo tempo: as granulosidades temporais[34] do sistema e do seu utilizador(es) são geralmente diferentes, variando de aplicação para aplicação. Adicionalmente, um serviço não se restringe apenas a saídas, mas engloba todas as interacções que são de interesse para o utilizador; por exemplo, ler

---

[30] [Wei, 97] refere que: "A observação de um sistema é recursiva por natureza. Um componente pode, por sua vez, ser consituído por componentes de *hardware* e *software*. Um exemplo simples disso é um microprocessador cujo conjunto de instruções é implementado em microcódigo. O microprocessador está sujeito a todos os tipos comuns de falhas de *hardware* (por exemplo, alteração do valor de um bit), bem como a todos os tipos usuais de falhas de *software* (por exemplo, falhas de concepção no microcódigo). O utilizador do microprocessador não sabe ou não lhe interessa que funcionalidades do componente são implementadas em *hardware* e *software*. O microprocessador parece uma peça de *hardware*, para a maior parte dos utilizadores. Para quem concebe o microcódigo, o microcódigo parece *software* a correr num bocado de hardware especializado (a máquina que executa o microcódigo). Quem concebe o microprocessador terá que lidar com *hardware* e *software* que, conjuntamente, constituem o microprocessador, e estar preparado para gerir uma má utilização deste componente por parte do utilizador."

[31] (NA5)

a) Podemos então dizer que uma "estrutura" tem também um "comportamento", especialmente no que respeita aos impedimentos à confiança de funcionamento, mesmo que as velocidades de evolução consideradas no que respeita i) ao pedido do utilizador(es), por um lado e ii) aos impedimentos, por outro lado, sejam - esperançosamente – diferentes.
b) A definição dada permite incorporar outros tipos de sistemas com estruturas variáveis, tais como os sistemas adaptáveis ou os sistemas baseados em conhecimento.

[32] Sinónimo de estado ([Por, 98] e [Pin, 48]), assim como circunstância e condição.

[33] (NA6) Esta definição pretende dar ênfase à relatividade da noção de estado, que depende directamente dos fenómenos e circunstâncias considerados; por exemplo, estados relativos às actividades de processamento de informação e estados relativos à ocorrência de avarias.

[34] *Granulosidade temporal* representa o passo ou intervalo temporal contado por um relógio (base de tempo global) de um sistema, normalmente distribuído. A *granulosidade temporal* é normalmente pequena, mas obviamente maior do que a incerteza cometida na contagem/medição do tempo ([Ver, 93]). Torna-se fundamental a definição da *granulosidade temporal* nos sistemas conduzidos pelo tempo ("*time-triggered*"), pois é necessário actualizar o estado do sistema a uma dada taxa, lendo as entradas e actualizando as saídas ([Kop, 93]).

sensores é claramente parte do serviço que se espera de um sistema de monitorização.

Utilizámos até este momento o singular para função e serviço. Um sistema cumpre normalmente mais do que uma função e presta mais do que um serviço. Pode então considerar-se que uma função e um serviço são constituídos, respectivamente, por um conjunto de funções e por um conjunto de serviços. Para simplificar, utilizaremos o plural simplesmente quando for relevante distinguir os vários elementos constituintes de uma função ou de um serviço.

As propriedades de um sistema no que respeita às suas restrições temporais são de especial interesse no que respeita à confiança no seu funcionamento. Uma **função ou serviço de tempo-real** é uma função/serviço que tem de ser prestado dentro de intervalos de tempo finitos, *ditados pelo meio envolvente*; um **sistema de tempo-real** é um sistema que cumpre pelo menos uma função de tempo-real ou presta pelo menos um serviço de tempo-real ([PDC, 90]).

A *especificação* de um sistema descreve o que dele se espera em termos a) da sua função e/ou do seu serviço e b) as condições em que – ou sob as quais – eles têm de ser prestados: ambientais, duração, desempenho, capacidade de observação, etc. A função e/ou o serviço começam por ser especificados, habitualmente, em termos do que *deverá* ser prestado, tendo em conta o objectivo(s) primário do sistema. Quando consideramos sistemas com premissas de segurança e inviolabilidade, esta especificação é normalmente completada com o que *não deve* acontecer (por exemplo, estados perigosos que poderão levar a catástrofes, ou à divulgação de informação confidencial). Tal especificação poderá levar à especificação de funções ou serviços adicionais que o sistema deverá prestar de modo a reduzir as possibilidades do que não deverá acontecer (por exemplo, verificando a identificação e os direitos de um utilizador).

Adicionalmente, estas diversas especificações poderão:

a) ser expressas segundo vários graus de pormenor: especificação dos requisitos, especificação do projecto, especificação da implementação, etc.

b) decompostas de acordo com a ausência ou a presença de falhas; o primeiro caso é relativo ao que normalmente se denomina o modo *nominal* de operação e o segundo poderá relacionar-se com o modo *degradado* de operação, se os recursos sobreviventes deixarem de ser suficientes para a prestação do serviço(s) nominal.

Como consequência, não existe normalmente uma especificação apenas, mas várias e, claramente, um sistema pode falhar no que respeita a algumas destas múltiplas especificações e, ao mesmo tempo, satisfazer as outras.

É essencial que uma especificação seja acordada por duas pessoas ou grupos de pessoas: o fornecedor do sistema (num sentido lato: o projectista, construtor, vendedor, etc.) e o seu utilizador(es) humano[35]. Este acordo é necessário de forma a que a especificação sirva de base para avaliar se o serviço prestado é aceitável ou não, ou, equivalentemente, se ocorreu ou não uma avaria. O que pode ser considerado como um serviço aceitável relativamente a uma especificação com um dado nível de pormenor, poderá não a satisfazer considerando um nível menos detalhado. Isto deve-se a enganos na elaboração da especificação, denominados de *falhas de especificação*, falhas estas que podem afectar qualquer uma das várias especificações. Em termos mais genéricos, não se pode considerar que uma especificação se torne imutável no momento em que é estabelecida. Isso seria ignorar simplesmente os factos da vida, que implicam *mudança*. Estas mudanças podem ser motivadas pela modificação dos requisitos do sistema: modificação da função e/ou do serviço esperado, ou a correcção de algumas

[35] (NA7) O acordo pode ser implícito, como quando adquirimos um sistema que vem com a sua especificação e manual de utilização., ou quando se usam sistemas "chave na mão" ("*off-the-shelf*").

falhas[36]. Mais uma vez, o que é importante é que a especificação seja acordada entre as partes.

A partir da definição anterior de estrutura de um sistema, as noções de função, de serviço e da sua especificação aplicam-se igualmente e naturalmente aos componentes. Isto torna-se particularmente interessante durante a concepção, quando são utilizados componentes "chave na mão" quer de software, quer de hardware; o que se torna mais interessante para o projectista é a função e/ou serviço que eles são capazes de fornecer, não tanto o seu comportamento (interno) detalhado.

[36] (NA8) Estamos portanto perante um problema circular: é necessária uma referência para determinar se um serviço prestado é aceitável ou não, mas a própria referência pode estar defeituosa. Desde há muito que se dedica uma atenção especial ao aperfeiçoamento das especificações, incluindo propostas de modelos de "ciclo de vida" destinados a este objectivo ([Boe, 88]).

4. IMPEDIMENTOS À CONFIANÇA NO FUNCIONAMENTO

4.1. FALHAS

As falhas[37] e as suas fontes são extremamente diversificadas. Elas podem ser classificadas de acordo com três pontos de vista principais: a sua natureza, a sua origem e a sua persistência[38].

A *natureza* das falhas leva-nos a distinguir:

- **falhas acidentais**, que aparecem ou são criadas fortuitamente;
- **falhas intencionais[39]**, que são criadas deliberadamente, presumidamente de má fé.

[37] A definição de falha dada neste texto, mais à frente, é "a causa julgada ou hipotética de um erro". Em '1. Definições de Base' refere-se que uma falha é uma *situação* que pode levar a que não se deposite confiança no serviço prestado por um sistema. [Nel, 90] refere que uma falha é "uma situação física anómala".

[38] Dois anos antes, ([Lap, 90a]), o autor apenas classificava as falhas de acordo com a sua independência (falhas relacionadas ou independentes) e a sua persistência (falhas ténues ou acentuadas). No mesmo ano, [Nel, 90] classificava as falhas de acordo com a sua duração (falhas transitórias, intermitentes ou permanentes), natureza e extensão (falhas locais ou globais). [Kop, 93] classifica as falhas segundo a sua: natureza (falhas acidentais ou intencionais), causa (falhas físicas ou de concepção), fronteira (falhas internas ou externas), origem (falhas de concepção ou de operação) e persistência (falhas temporárias ou permanentes). [Joh, 96], por sua vez, classifica as falhas segundo a sua: *causa* (enganos no projecto, enganos na implementação, defeitos nos componentes e perturbações externas) *natureza* (não especifica muito bem que tipos pode haver), *duração* (falhas permanentes, transitórias e intermitentes), *extensão* (falhas locais ou globais) e *valor* (falhas de valor determinado ou de valor indeterminado).

[39] Em ([Lap, 95a], [Lap, 98]), o autor especializa este conceito em *falhas intencionais maliciosas* e *falhas intencionais não maliciosas*.

A *origem* das falhas pode, por sua vez, ser decomposta em três pontos de vista[40]:

- as *causas fenomenológicas*, que levam a distinguir ([Avi, 78]):
  - **falhas físicas**[41], que se devem a fenómenos físicos adversos;
  - **falhas humanas**, que resultam de imperfeições humanas[42];
- as *fronteiras do sistema*[43], que levam a distinguir:
  - **falhas internas**, que são aquelas partes do estado de um sistema que, quando invocadas pela actividade computacional, irão produzir um erro;
  - **falhas externas**, que resultam de interferência ou de interacção com o seu meio envolvente físico (perturbações electromagnéticas, radiação, temperatura, vibração, etc.) ou humano;
- a *fase de criação*, no que respeita à vida do sistema, que leva a distinguir:
  - **falhas de concepção**[44], que resultam de imperfeições a) durante o desenvolvimento do sistema (da especificação dos requisitos até à implementação) ou durante modificações subsequentes, ou b) durante o estabelecimento dos procedimentos para operar ou manter o sistema;

[40] [Joh, 96] defende que as *causas* das falhas podem ser:
- *enganos no projecto*: incluem algoritmos, arquitecturas e projecto de software e hardware incorrectos;
- *enganos na implementação*: projecto pobre, escolha de componentes de fraca qualidade, montagem inadequada ou enganos nos programas;
- *defeitos dos componentes*: imperfeições de fabrico, envelhecimento dos componentes;
- *perturbações externas*: radiação, interferência electromagnética, erros do operador, condições ambientais adversas.

[41] Exemplos de falhas físicas ([Nel, 90]) são componentes defeituosos devido quer a fadiga de material ou outro tipo de deterioração, quer a perturbações externas, tais como condições adversas do meio envolvente, interferência electromagnética, etc..

[42] É muito importante ter em conta que a classificação de uma falha como uma falha humana é muito discutível e complexa. Por exemplo, até que ponto se pode culpar o piloto de um avião de um determinado acidente, decretando que foi *falha humana*, quando a instrumentação foi concebida com uma pobre ergonomia (instrumento digital quando deveria ser analógico, indicação demasiado pequena, luz de alarme fracamente perceptível, etc.). A comunidade científica e industrial que se dedica à investigação e desenvolvimento de sistemas de segurança crítica está a começa agora a considerar de extrema importância áreas científicas como a ciência do conhecimento, a psicologia e a ergonomia, a par das tradicionais engenharias de hardware e software, numa perspectiva holística ([OLO, 98]). Esta análise integrada é particularmente importante na análise, projecto e implementação de Sistemas Homem-Máquina.

[43] O conceito de falha interna ou externa depende inteiramente das fronteiras consideradas para o sistema. De facto, muitas vezes temos de considerar o utilizador humano de um sistema como parte integrante deste, pois o sistema foi concebido para interagir com o seu utilizador. Este ponto de vista alarga obviamente os limites das falhas internas, pois falhas originadas pelo operador/utilizador humano passam a ser consideradas falhas internas.

[44] Tradução de "*design faults*", tal como em [Mag, 95]. Opta-se por traduzir "*design*" como "concepção" e não como "projecto", pois neste contexto, "*design*" significa todo o processo de concepção até o sistema ficar pronto a utilizar (tal como em [Lap, 90a]). A implementação pode ser vista como a transformação da especificação (projecto) de hardware e software no software e hardware "reais" ([Joh, 96]). Em engenharia, principalmente na engenharia de computadores, normalmente utilizam-se separadamente os termos projecto e implementação, podendo englobar-se os dois no termo concepção (mais abrangente). Como o autor percebe "*design faults*" como as falhas existente em todo o processo de concepção do sistema (análise, projecto e implementação), opta-se pelo termo "falhas de concepção".

– **falhas de operação**, que aparecem durante a exploração do sistema;

Pode também ser feita uma distinção no que concerne à *persistência*[45] temporal das falhas, levando a:

- **falhas permanentes**, quando a sua presença não está relacionada com condições pontuais, internas (actividade computacional) ou externas (meio envolvente);
- **falhas temporárias**, quando a sua presença se deve a tais condições e, consequentemente, estão presentes durante um tempo limitado[46].

Os problemas de inviolabilidade são dominados por falhas intencionais, que são claramente falhas humanas, mas não só. As falhas intencionais podem ser internas ou externas; exemplos típicos são:

- no que respeita a falhas internas, a inclusão de **lógica maliciosa** (os chamados "Cavalos de Tróia"), que é uma falha intencional *de concepção*;
- relativamente a falhas externas, uma **intrusão** que é uma falha *externa de operação* e intencional.

De modo a serem "bem sucedidas", as falhas intencionais podem tirar partido de falhas acidentais, como por exemplo uma intrusão que aproveita uma brecha na inviolabilidade devido a uma falha acidental de concepção; existem similaridades interessantes e óbvias entre este exemplo e uma falha externa, temporária e acidental que "explore" uma falta de blindagem[47].

Poderia discutir-se que a introdução das causas fenomenológicas no critério de classificação das falhas poderia levar-nos, recursivamente, "muito para trás", por exemplo: por que é que os programadores cometem erros? porque é que os circuitos integrados avariam? A própria noção de falha é *arbitrária*, sendo de facto um meio disponibilizado para parar a recursividade. Daqui resulta a definição de falha: causa *julgada* ou *hipotética* de um erro. Esta causa poderá depender do ponto de vista adoptado: mecanismos de tolerância a falhas, engenheiros de manutenção, oficina de reparação, projectista, físico de semicondutores, etc. A nosso ver, a recursividade pára *na causa que se pretende ser prevenida ou tolerada*. Este ponto de vista torna consistente a distinção entre falha humana e falha física: sendo um sistema computacional uma criação do homem, qualquer falha nele existente ou que o afecte é, em última análise, provocada por pessoas. Isto deve-se à incapacidade humana para gerir todos os fenómenos que regem o comportamento de um sistema. Em termos absolutos, a distinção entre falhas físicas e humanas (especialmente falhas de concepção) pode ser considerada desnecessária. Contudo, esta distinção torna-se relevante quando consideramos os métodos e técnicas (actuais) para a obtenção e validação da confiança no funcionamento. Se não pararmos a recursividade acima referida, então *uma falha mais não é do que a consequência de uma avaria de um outro sistema (incluindo o projectista) que prestou ou está a prestar um serviço ao sistema em consideração.*

Seguem-se alguns exemplos da discussão anterior:

---

[45] [Joh, 96] utiliza o termo "*duration*", definindo-o como o intervalo de tempo em que uma falha está activa, podendo ser uma falha permanente, transitória ou intermitente (não agrupa transitória e intermitente em temporária, como aqui).

[46] As falhas temporárias são reconhecidas como a grande maioria das falhas de *hardware* ([Lap, 93]), facto agravado pela crescente integração dos circuitos electrónicos. Contudo, as falhas temporárias também se aplicam ao *software*. [Lap, 93] refere que a maior parte das falhas de *software* que aparecem durante a vida operacional de um sistema são falhas temporárias. Isto é um pouco estranho, pois apesar de não se poder discutir que as falhas de *software* estarão presentes enquanto não forem "tratadas", deve reconhecer-se que a maioria das falhas de software que se manifestam em programas grandes e complexos, são suficientemente subtis, de forma a que as suas condições de activação dependem de combinações igualmente subtis do estado interno e de solicitações externas, implicando uma quase impossibilidade da sua reprodução. Dito de outra forma, o domínio das avarias das falhas temporárias de *software* pode variar com as condições de execução do *software*, podendo ser um conjunto vazio na maioria das condições de operação.

[47] Tradução de "*lack of shielding*"

- uma falha de concepção resulta de uma avaria de quem concebeu o sistema[48];
- uma falha física interna é devida a uma avaria num componente de hardware, que, por sua vez, é a consequência de um ou mais erros ao nível eléctrico ou electrónico (a comunidade da "fiabilidade física" raramente caracteriza as avarias como "súbitas e imprevisíveis"). Este erros têm como origem desordens físico-químicas, novamente resultantes da produção do hardware, ou mesmo o limite do nosso conhecimento de física dos semicondutores. As falhas físicas internas podem ser vistas como falhas *recorrentes*, desde que a sua reparação consista na substituição de um componente avariado por um outro, idêntico, não avariado. O novo componente poderá avariar no futuro, de forma similar, a não ser que a causa da avaria tenha sido identificada na concepção e produção e removida, conduzido a um componente modificado com menor probabilidade de avariar;
- uma falha externa física ou humana é, de facto, uma falha de concepção: a incapacidade de prever todas as situações com que o sistema se vai deparar durante a sua vida operacional, ou a recusa de considerar algumas delas (por razões económicas, por exemplo), por exemplo:
  - no caso de uma interferência electromagnética: será uma falha externa ou uma falha de concepção, isto é, a ausência de uma blindagem adequada?
  - no caso de uma avaria causada por um operador que tecla um caracter inapropriado: será uma falha de interacção[49] ou uma falha de concepção, isto é, a ausência de confirmação por parte do sistema ([Nor, 83])?

[48] *Avaria* de quem concebeu o sistema no sentido em que essa pessoa em alguma altura deixou de prestar o serviço que dela se esperava.

[49] Interacção com o meio envolvente (humano), faz parte das falhas externas (ver 43).

Quanto à persistência temporal, tecem-se ainda os seguintes comentários:

1) as falhas externas temporárias originadas pelo meio envolvente físico são muitas vezes denominadas de **falhas transitórias**[50].
2) As falhas internas temporárias são normalmente chamadas de **falhas intermitentes**[51]: estas falhas resultam da ocorrência (muito rara) de combinações de condições; exemplos são: a) falhas sensíveis a determinados padrões nas memórias semicondutoras, alterações nos parâmetros dos componentes de hardware (efeitos da variação de temperatura, atrasos na temporização devido a capacidades parasitas, etc.), ou b) situações – afectando tanto hardware como software – que ocorrem quando a carga do sistema ultrapassa um determinado nível, tais como temporizações ou sincronizações marginais. A própria noção de falha intermitente é, última instância, arbitrário: estas falhas são nada mais nada menos do que falhas permanentes cujas condições de activação não podem ser reproduzidas ou que ocorrem (suficientemente) raramente.

[50] [Mag, 95] diz que as falhas temporárias podem ser classificadas como falhas transitórias e falhas intermitentes. Transitórias quando se manifestam casualmente e com uma curta duração.

[51] No seguimento da nota anterior, as falhas temporárias são intermitentes quando se manifestam repetidamente. [Joh, 96] quanto classifica as falhas quanto à sua *duração*, especializa logo em:, *falhas transitórias* e *falhas intermitentes*.

- *falhas permanentes*: permanecem indefinidamente até que seja tomada uma acção de correcção;
- *falhas transitórias*: podem aparecer e desaparecer após um pequeno período de tempo;
- *falhas intermitentes*: aparecem, desaparecem e voltam a aparecer repetidamente.

Esta classificação, similar à de [Mag, 95] parece-nos mais correcta e simples de entender que a deste documento, que continua a ser defendida pelo autor em [Lap, 95a], [Lap, 95b] e [Lap, 98].

Contudo, tal como já se considerou na distinção entre falhas físicas e falhas de concepção, a sua consideração é útil.

Da discussão precedente, parece que *qualquer falha pode ser vista como uma falha permanente de concepção*. Isto é realmente verdade em termos absolutos, mas não se torna muito útil para as pessoas que concebem e avaliam sistemas.

A Figura 2 sumaria as várias classes de falhas[52] que se consideraram atrás, classificadas segundo os pontos de vista identificados. Se fossem possíveis todas as combinações de classes de falhas, de acordo com os 5 pontos de vista da Figura 2, haveria 32 classes de falhas diferentes. De facto, o número de combinações verosímeis é menor: 11 combinações são indicadas pelas linhas da Figura 3[53], que também fornece os nomes normalmente a elas atribuído – *não a sua definição*. Estes nomes são usualmente utilizados de modo a exprimir de forma condensada o resultado da combinação de vários pontos de vista.

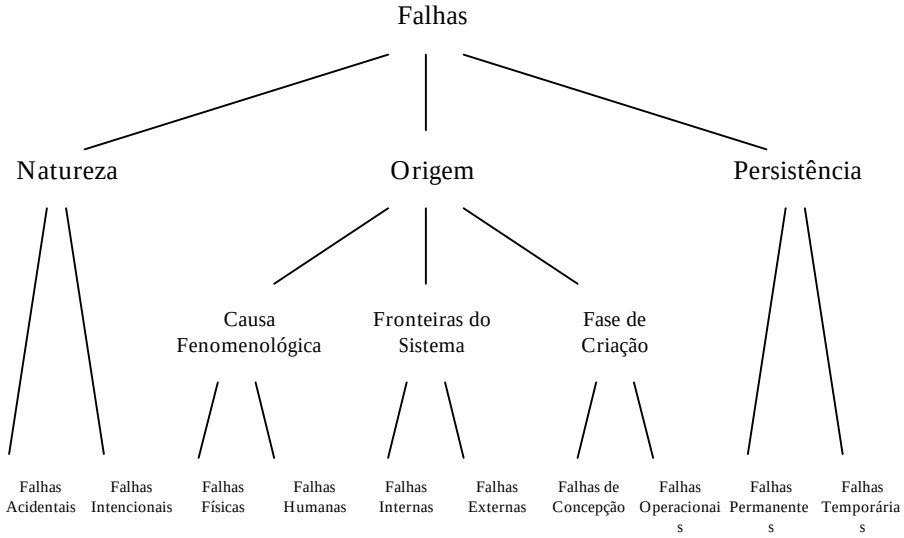

Figura 2 – As classes de falhas segundo vários pontos de vista

[52] Podem considerar-se como classes de falhas elementares ([Lap, 95a], [Lap, 95b], [Lap, 98]).

[53] Merece ser consultada a actualização desta tabela em ([Lap, 95a], [Lap, 95b], [Lap, 98]), apresentando 15 combinações de falhas com probabilidade de ocorrer. Em [Lap, 95a] e [Lap, 98] é também dedicada uma atenção particular às classes de falhas combinadas que derivam de **falhas humanas**:

- *Falhas de concepção*, que são as falhas de concepção acidentais ou intencionais, maliciosas ou não;
- *Falhas de interacção*, que são as falhas externas, acidentais ou intencionais não maliciosas;
- *Lógica maligna*, que são as falhas internas intencionais maliciosas;
- *Intrusões*, que são as falhas de operação externas, intencionais maliciosas.

São também interessantes os comentários tecidos acerca das classes de falhas humanas, na mesma referência.

Natureza		Origem						Persistência		Nome comum
		Causa fenomen.		Fronteiras sist.		Fase de criação				
Acident	Intenci	Físicas	Human	Interna	Externa	Concep	Operaç	Perman	Tempor	
✓		✓		✓			✓	✓		Físicas
✓		✓			✓		✓	✓		
✓		✓			✓		✓		✓	Transit
✓		✓		✓			✓		✓	Intermitentes
✓			✓	✓		✓			✓	
✓			✓	✓		✓		✓		Concep
✓			✓		✓		✓		✓	Interac
	✓		✓	✓		✓		✓		Lógica maliciosa
	✓		✓	✓		✓			✓	
	✓		✓		✓		✓	✓		Intrusões
	✓		✓		✓		✓		✓	

Figura 3 – As classes de falhas resultando das combinações de acordo com os vários pontos de vista da Figura 2.

4.2. ERROS

Um erro[54] foi definido como sendo susceptível de provocar uma avaria. Um erro conduz ou não a uma avaria dependendo de três factores principais:

1) A composição do sistema, e particularmente do tipo de redundância[55] existente:

[54] Curiosamente, este documento não dá uma definição de erro, por si só. Apenas refere (1. Definições de Base) que é uma *situação* que pode levar a que não se deposite confiança no serviço prestado por um sistema. [Lev, 95] define erro como "uma imperfeição na concepção ou o desvio de um estado desejado ou esperado. Enquanto que uma avaria é definida como um acontecimento (comportamento), um erro é uma situação estática (estado)". [Nel, 90] refere que "um erro é a manifestação de uma falha num sistema, em que o estado lógico de um elemento difere do valor pretendido. Um erro ocorre quando uma falha é "sentida"; por outras palavras, para um estado particular de um sistema e um dado estímulo de entrada, resulta um estado e/ou saída erróneo".

[55] O autor não define o que é redundância, por si só. Apenas define o que é *redundância temporal*: comportamento redundante obtido por repetição. Contudo é altamente relevante, especialmente neste contexto, dar uma definição de redundância. [Kir, 87] define *redundância* como "os elementos adicionais necessários devido a possíveis avarias e ausentes em concepções simplificadas". Essa referência distingue também entre dois tipos de redundância: *redundância para verificação* (redundância introduzida com o único propósito de detecção de erros) e *redundância funcional* (fornece a mesma função que a parte funcional). [Kop, 93] define redundância como os recursos extra num sistema que não seriam necessários num mundo perfeito, sendo necessária para detectar erros e para dissimular falhas no mundo real. Esta referência considera três possíveis tipos de redundância: *redundância de recursos físicos* (replicação de recursos físicos), *redundância temporal* (repetição de processamento ou comunicação) e *redundância de informação* (técnicas de codificação). Distingue ainda entre *redundância passiva* ("*passive, cold or standby redundancy*") e *redundância activa* ("*active, hot or workby redundancy*"). [Mag, 95] define *redundância protectora* de um sistema computacional como os recursos que foram acrescentados aos estritamente necessários ao seu funcionamento. Divide apenas em *redundância estrutural* e *redundância temporal*. Para [Joh, 96], o conceito de redundância implica a adição de informação, recursos ou tempo para além do

- redundância *intencional* (introduzida para proporcionar tolerância a falhas), que se destina explicitamente a evitar que um erro conduza a uma falha;
- redundância *involuntária* (na prática, é difícil, senão impossível, de construir um sistema sem qualquer forma de redundância[56]) que pode ter o mesmo resultado que a redundância intencional (mas inesperado).

2) A actividade do sistema: um erro pode ser apagado antes de produzir estragos[57].

3) A definição de uma avaria do ponto de vista do utilizador: o que poderá ser uma avaria para um dado utilizador, pode ser uma insignificância perfeitamente suportável para outro. Exemplos são: a) a tomada em consideração da granulosidade temporal de um utilizador: um erro que "atravessa" a interface sistema-utilizador(es) pode ou não ser visto como uma avaria, dependendo da granulosidade temporal do utilizador; b) a noção de "taxa de erros aceitável" – implicitamente antes de considerar que uma avaria ocorreu – na transmissão de dados. Esta discussão explica porque que é muitas vezes desejável mencionar explicitamente na especificação, condições tais como o máximo período de tempo admissível para a prestação de um serviço (relacionado com a granulosidade temporal do utilizador).

que é necessário para o funcionamento normal do sistema. Classifica ainda quatro tipos de redundância: *redundância de hardware*, *redundância de software*, *redundância de informação* e *redundância temporal*.

[56] (NA9) Um problema clássico no teste de *hardware* é a supressão de tais "falsas redundâncias", cujo efeito pode ser dissimular (certas) falhas e, como tal, tornar mais complicada a tarefa da geração de padrões de teste.

[57] Tentar explicar e dar um exemplo...

4.3. AVARIAS

Em função da definição adoptada para estrutura de um sistema, a questão se "avaria" se aplica a um sistema ou a um componente é simplesmente irrelevante, dado que um componente é, ele próprio, um sistema. Quando se consideram sistemas atómicos, a noção de avaria "elementar" aparece naturalmente.

Um sistema pode não avariar sempre da mesma maneira, o que acontece na generalidade dos casos. As maneiras como um sistema pode avariar são os seus *modos de avaria*, que podem ser caracterizados segundo três pontos de vista[58]: domínio, percepção por parte dos utilizadores do sistema e consequências no meio envolvente.

O *domínio da avaria* leva a distinguir:

- **avarias de valor**[59]: o valor do serviço prestado não está de acordo com a especificação;
- **avarias temporais**[60]: o tempo de prestação de um serviço não está de acordo com a especificação.

Estas definições gerais (não conformidade com a especificação) caracterizam as **avarias arbitrárias**. É também possível refinar os modos de avarias. Por exemplo, para a noção de **avaria temporal**[61], podemos distinguir avaria

[58] [Kop, 93] classifica as avarias segundo a sua: *natureza* (avarias de valor ou temporais), *percepção* (avarias consistentes ou inconsistentes), *efeitos* (avarias benignas ou malignas) e *frequência* (avarias pontuais ou repetidas). [Joh, 96] não classifica as avarias.

[59] Tal como em 11, as "avarias de valor" são definidas em ([Lap, 95a], [Lap, 95b], [Lap, 98]) como avarias em que o valor do serviço prestado deixa de permitir o cumprimento da *função* do sistema.

[60] Tal como em 11, as "avarias temporais" são definidas em ([Lap, 95a], [Lap, 95b], [Lap, 98]) como avarias em que as condições temporais da prestação de um serviço deixam de permitir o cumprimento da *função* do sistema.

[61] Tradução de "*timing failures*".

temporal *em avanço* ou *em atraso*, dependendo se o serviço é prestado demasiado cedo ou demasiado tarde. Uma classe de avarias que engloba alterações temporais e de valor é constituída pelas **avarias por paragem**[62]: a actividade do sistema, se existir, deixa de ser perceptível pelos utilizadores, passando a ser prestado um serviço de valor constante; o valor constante entregue pode variar de acordo com a aplicação, por exemplo o último valor correcto, um valor predeterminado, etc. Um caso particular das avarias por paragem é constituído pelas **avarias por omissão** ([Cri, 85b], [Ezh, 86]): não é prestado qualquer serviço. Tais avarias podem ser vistas com um limite comum para as avarias de valor (nenhum valor) e para as avarias temporais (infinitamente tarde); uma avaria por omissão persistente é chamada **avaria por omissão persistente**[63].Um sistema em que as avarias apenas podem ser – ou mais genericamente, onde as avarias são numa quantidade aceitável – avarias por paragem, denomina-se de **sistema avaria-pára**[64]. Um sistema que as avarias apenas podem ser – ou são numa certa quantidade – avarias por omissão persistente, é chamado de sistema **avaria-silencia-se**[65] ([Pow, 88]).

Quando um sistema tem diversos utilizadores, a *percepção da avaria* leva a distinguirem-se:

- **avarias consistentes**: todos os utilizadores do sistema têm a mesma percepção das avarias;
- **avarias inconsistentes**: os utilizadores do sistema poderão ter diferentes percepções de uma dada avaria; estas avarias, após [Lam, 82], passaram também a ser chamadas de *avarias Bizantinas*.

[62] Tradução de "*stopping failures*".

[63] Tradução de "*crash failure*". Optou-se pela não tradução "à letra" (qualquer coisa como "avaria ruinosa") para evitar confusões com "avaria catastrófica".

[64] Tradução de "*fail-stop*".

[65] (NA10) A noção de *processador* avaria-pára, tal como definida em no contexto dos sistemas distribuídos, pode ser vista como um exemplo de um sistema avaria-silencia-se.

A **gravidade**[66] das avarias resulta da classificação das *consequências das avarias* para o meio envolvente do sistema. Estas permitem portanto ordenar os modos de avaria. Um caso especial de grande interesse é o de sistemas cujos modos de avaria podem ser agrupados em duas classes, cuja gravidade difere consideravelmente:

- **avarias benignas**[67], onde as consequências são da mesma ordem de grandeza (medidas geralmente em termos de custos) do benefício proporcionado pelo serviço prestado pelo sistema na ausência de avarias;

[66] Tradução de "*severity*".

[67] Os conceitos de avarias benignas e catastróficas estão directamente associados com os conceitos de sistemas críticos e não críticos. [Kop, 93] define sistemas de tempo-real não críticos ("*soft real-time systems*") e sistemas de tempo-real críticos ("*hard real-time systems*"), dependendo das consequências para o meio envolvente das avarias temporais (não no domínio do valor), dando exemplos destes sistemas. Num sistema de comutação telefónico é exigida uma elevada disponibilidade, mas uma chamada mal comutada não tem consequências catastróficas, isto é, tem o mesmo valor que o serviço que se espera. Por isso, pode ser considerado como um sistema de tempo-real não crítico. Um exemplo de um sistema de tempo-real crítico é um sistema de sinalização ferroviária, onde uma avaria (tanto no domínio do valor como no domínio temporal) pode ter consequências catastróficas tanto para bens como para pessoas. É interessante referir ainda as ideias de [Mag, 95], que defende que um serviço de controlo de tempo real tem realmente dois *limites temporais* ("*deadlines*"): um *nominal* – quantificado num contexto de desempenho e cujo não cumprimento tem consequências benignas – e um *crítico* – definido num contexto de segurança e cujo não cumprimento pode ser catastrófico. O intervalo de tempo entre estes dois limites temporais é denominado de *complacência temporal* ("*grace time*"). Este conceito tinha sido já abordado por [Kir, 87], defendendo que a complacência temporal é o tempo que um sistema tolera uma avaria, sem sofrer danos.

- **avarias catastróficas**[68], onde as consequências são incomensuravelmente superiores ao benefício proporcionado pelo serviço prestado pelo sistema na ausência de avarias.

Um sistema onde todas as avarias são - ou mais genericamente, onde as avarias são numa quantidade aceitável – avarias benignas, é um **sistema avaria-seguro**[69]. A noção de gravidade de uma avaria permite definir a noção de criticalidade: a **criticalidade** de um sistema é a mais alta gravidade dos seus modos (possíveis) de avaria[70]. A relação entre modos de avaria e severidades de avaria é altamente dependente da aplicação. Contudo, existe uma vasta gama de aplicações onde a inactividade é considerada como uma situação naturalmente segura (por exemplo os transportes terrestres e a produção de energia), de onde resulta a correspondência directa que é muitas vezes feita entre sistemas avaria-pára e sistemas avaria-seguro ([Min, 67], [Nic, 89])[71].

4.4. PATOLOGIA DAS FALHAS

Os mecanismos de criação e manifestação de falhas, erros e avarias pode ser sumariado como se segue:

1) Uma falha torna-se **activa** quando produz um erro. Uma falha activa poderá ser a) uma falha interna que estava previamente **dormente** e que foi activada pelo processo computacional, ou b) uma falha externa. A maior parte das falhas internas transita entre o estado activo e dormente. As falhas físicas apenas podem afectar

[68] Avarias catastróficas ou avarias malignas.

[69] Tradução de "*fail-safe*". Segundo [Kir, 87], um sistema avaria-seguro pára sempre de forma a que o sistema seja conduzido a um estado seguro. Muitos sistemas dispõem de um mecanismo de salvaguarda que desactiva o sistema quando é detectada uma situação perigosa, para evitar avarias catastróficas. O sistema pode ser colocado num estado seguro sem intervenção do sistema computacional de controlo. Por exemplo, os comboios podem chegar a um estado seguro através da travagem de emergência (que é accionada quando se perde pressão) e no transporte de energia eléctrica, através de relês de protecção contra sobreintensidades. [Kop, 93] classifica um sistema de tempo-real crítico como *avaria-seguro* ou *avaria-operacional* (tradução de "*fail-operational*"). Nos sistemas avaria-seguro, o sistema computacional deve ter uma cobertura elevada de detecção de erros, isto é, a probabilidade de um erro ser detectado deve ser próxima de um. Contudo, existem aplicações onde não é possível identificar um estado seguro, isto é, é necessário que o sistema computacional garanta sempre um nível mínimo de serviço, como é o caso de um sistema de controlo de navegação, num avião. Neste caso, trata-se de um sistema avaria-operacional.

[70] (NA11) Como exemplo, os níveis de criticalidade aceites pela comunidade aeronáutica são definidos como se segue:

- crítico: funções para as quais a ocorrência de qualquer avaria inviabilizaria a continuação de um voo e de uma aterrissagem seguros;
- essencial: funções para as quais a ocorrência de qualquer avaria reduziria a capacidade do avião ou a aptidão da tripulação para lidar com condições adversas de operação;
- não essencial: funções onde uma avaria não poderia degradar nem a capacidade do avião nem a capacidade da tripulação.

[71] Mesmo sistemas com funcionalidades parecidas podem ter diferentes modos de avariar. Comparemos o caso da sinalização de tráfego rodoviário com a sinalização de tráfego ferroviário. Na sinalização de tráfego rodoviário, se houver uma avaria, é perfeitamente aceitável e seguro que o sistema deixe de prestar qualquer serviço (os semáforos apagam-se, simplesmente). Muito perigoso é se o serviço prestado tiver uma dada avaria de valor (por exemplo, dois semáforos verdes, ao mesmo tempo, para trajectórias perpendiculares). Mais crítico em termos de segurança é a sinalização de tráfego ferroviário, pois põe em risco um maior número de pessoas e bens. Este tipo de sistema de controlo já não poderá ser do tipo *avaria-pára*, pois terá de garantir pelo menos algumas funções (cancelas fechadas, impedir o choque de dois comboios, etc.). Chama-se a este tipo de sistema um sistema *avaria-operacional*, pois mantém pelo menos um modo degradado de funcionamento, garantindo um determinado número de funções consideradas críticas. Por exemplo, um sistema de controlo de umas escadas rolantes poderá ser perfeitamente do tipo *avaria-pára*, pois é seguro que uma pessoa termine a subida pelos seus próprios meios. No caso de um sistema de controlo de elevadores, terá de ser do tipo *avaria-operacional*, pois pode ser considerado crítico uma ou mais pessoas ficarem presas entre dois andares. O sistema deverá então garantir, mesmo de um modo degradado (mais devagar, ou em modo manual) que as pessoas saiam do elevador.

directamente os componentes de hardware, enquanto que as falhas humanas podem afectar qualquer componente.

2) Um erro pode estar latente ou detectado. Um erro está **latente**[72] quando ainda não foi reconhecido com tal; um erro pode ser **detectado** por um mecanismo ou algoritmo de detecção. Um erro pode desaparecer antes de ter sido detectado. Um erro pode propagar-se, o que geralmente acontece; ao propagar-se, um erro cria outro – novo - erro(s). Durante esta operação, a presença de falhas activas é apenas determinada pela detecção de erros[73].

3) Uma avaria ocorre quando um erro "passa pela" interface utilizador-sistema e afecta o serviço prestado pelo sistema. A avaria de um componente resulta na falha a) para o sistema que contem o componente e b) do ponto de vista dos outros componentes com que interage; os modos de avaria do componente avariado tornam-se então tipos de falha para os componente que com ele interagem.

Estes mecanismos permitem completar a "cadeia fundamental" [74,75]:

... → avaria → falha → erro → avaria → falha → ...

Como exemplos da patologia das falhas poderemos ter:

- o resultado do *erro* de um programador é uma falha (dormente) no software escrito (instruções ou dados errados); depois da sua activação (invocação do componente onde a falha reside e despoletar a instrução defeituosa, sequência de instruções ou dados com um

[72] [Joh, 96] define os conceitos *latência de uma falha* e *latência de um erro*: latência de uma falha é o intervalo de tempo entre a ocorrência de uma falha e o aparecimento de um erro devido a essa falha; latência de um erro é o intervalo de tempo entre a ocorrência de um erro e o aparecimento da avaria resultante. [Por, 98] define *latente* (adjectivo) como "que não se manifesta exteriormente, oculto, dissimulado" e *latência* como "tempo que decorre entre o começo de um estímulo e a resposta do paciente".

[73] É interessante o conceito de latência de uma falha e latência de um erro introduzido por [Joh, 96]. *Latência de uma falha* é o período de tempo entre a ocorrências de uma falha e o aparecimento de um erro devido a essa falha. *Latência de um erro* é o intervalo de tempo ente a ocorrência de um erro e o surgimento da avaria correspondente.

[74] Nesta altura no texto, aconselha-se a consulta de [Joh, 96], que apresenta os conceitos de falhas, erros e avarias recorrendo a um *modelo de três universos*:

- *universo físico*: onde ocorrem as falhas; engloba dispositivos semicondutores, elementos mecânicos, monitores, fontes de alimentação e outras entidades físicas que compõem um sistema; uma falha é um defeito ou alteração física de um componente no universo físico;
- *universo da informação*: onde ocorrem os erros; os erros afectam unidades de informação tais como o conteúdo de posições de memória num computador; um erro ocorre quando alguma unidade de informação se tornar incorrecta;
- *universo externo ou do utilizador*: é onde o utilizador de um sistema sente, em última instância, os efeitos das falhas e erros; o universo externo é onde ocorrem as avarias; a avaria é qualquer desvio do comportamento desejado ou esperado de um sistema.

[75] Merece uma atenção particular a leitura da secção '*2.4. Hierarchical failure propagation and masking*' de [Cri, 91], onde se aborda o problema da *hierarquia da propagação de avarias*. Segundo esta referência, o comportamento de uma avaria apenas pode ser classificado de acordo com uma certa especificação do sistema, num dado nível de abstracção. Se um sistema computacional é constituído por vários níveis hierárquicos de controlo, então uma avaria de um certo tipo num nível de abstracção inferior pode resultar numa avaria de outro tipo no nível de abstracção mais elevado. Por exemplo, suponhamos uma avaria de valor na camada física de uma rede de comunicação de dados que provoque que dois *bits* sejam corrompidos. Se a camada de ligação de dados (imediatamente acima da camada física) utilizar um código de detecção de erros que detecte pelo menos 2 *bits* erróneos e deite fora as mensagens corrompidas, então esta avaria propaga-se à camada de ligação de dados como uma avaria por omissão.

determinado padrão de entrada), a falha torna-se activa e produz um erro; se os dados erróneos afectarem o serviço prestado (em valor ou no tempo da sua prestação), ocorre uma avaria;

- um curto-circuito num circuito integrado é uma avaria (relativamente à especificação do circuito); a consequência (conexão fixa num dado valor *booleano*, alteração das funções do circuito, etc.) é uma falha que permanecerá dormente desde que não seja activada; a continuação do processo é idêntica ao exemplo precedente;
- uma perturbação electromagnética de energia suficiente é uma *falha*; esta falha poderá:
  a) causar directamente um erro, por exemplo pela interferência electromagnética nas cargas eléctricas que circulam ao longo dos fios condutores,
  b) provocar uma outra falha (interna); por exemplo, se a perturbação actua nas entradas de uma memória no modo de escrita e altera o valor de alguns bits, estes erros vão subsequentemente permanecer como falhas na memória; estas vão permanecer dormentes até aquele endereço de memória ser lido; a sequência erro-avaria desde a falha transitória externa até à falha interna ainda existe, mas ao nível electrónico;
- uma interacção homem-máquina inapropriada, executada por um operador, durante a vida operacional de um sistema é uma *falha* (do ponto de vista do sistema); a alteração dos dados resultantes é um *erro*; etc.;
- um *erro* do editor de um manual de manutenção ou de utilização pode resultar numa *falha* no manual correspondente (directivas defeituosas) que permanecerá dormente até que as referidas directivas não sejam aplicadas para fazer face a uma dada situação., etc.

Dos exemplos anteriores, percebe-se facilmente que a latência das falhas pode variar consideravelmente, dependendo da falha, da utilização que se faz do sistema, etc.

As falhas humanas podem ser acidentais ou intencionais. O exemplo anterior relativo ao erro de um programador e das suas consequências pode ser refraseado como se segue: um vírus[76] é criado por um programador malicioso; ele permanece dormente até ser activado (por exemplo, numa data predeterminada); nessa altura produz um erro que pode levar a uma sobrecarga da capacidade da memória ou a uma execução do programa mais lenta; como consequência, a prestação do serviço vai deteriorar-se[77], causando uma avaria.

A simplicidade destes exemplos foi deliberada. A vida real é, normalmente, muito mais complicada, tal como espelham os seguintes quatro exemplos:

a) uma dada falha num dado componente pode resultar de diferentes fontes; por exemplo, uma falha permanente num componente físico (curto-circuito à massa) pode resultar de:
   - uma falha física (causada por uma alteração num nível de comparação[78]),
   - um erro causado por uma falha de concepção (falha no microcódigo) propagando-se de cima-abaixo através das camadas e causando um curto-circuito entre duas saídas com uma duração suficientemente grande para provocar uma avaria com as mesmas consequências que a alteração no nível de comparação;
b) uma falha de uma dada classe pode, através da propagação de erros, criar uma falha de outra classe; por exemplo, a falha que levou a um

[76] Tradução de "*logic bomb*".

[77] Preferiu-se este termo a "*denial-of-service*".

[78] Tradução de "*threshold*".

erro durante a execução do microcódigo, no exemplo anterior, poderia ter sido uma falha transitória;

c) durante o processo de propagação, certos pontos de vista podem tornar-se – pelo menos temporariamente – menos importantes; por exemplo, quando lidamos com falhas externas que produzem erros de entrada durante a execução de componentes de software (portanto, invocando-o no seu - assim chamado – domínio excepcional de entrada ([Cri, 80])), o facto de a falha ser física ou humana pode não ser importante para o comportamento da avaria do componente dado;

d) uma avaria resulta muitas vezes da acção combinada de várias falhas; isto é particularmente verdadeiro quando consideramos aspectos de inviolabilidade: num sistema computacional, pode haver uma falha de concepção, quer acidental, quer intencional, se houver forma de evitar o controlo de acesso ao sistema; esta falha poderá permanecer dormente até que alguém malicioso faça uso disso para entrar no sistema; a entrada do intruso no sistema é uma falha de interacção intencional; quando o intruso entra no sistema (quando não deveria entrar), pode deliberadamente criar um erro, por exemplo alterando um ficheiro (ataque à integridade); quando este ficheiro for utilizado por um utilizador autorizado, o serviço vai ser afectado, ocorrendo uma avaria.

Dois comentários adicionais, relativamente aos termos "falha", "erro" e "avaria":

a) o seu uso exclusivo neste documento não impede a utilização, em situações especiais de termos que designam, de forma curta e não ambígua, uma classe específica de impedimento; isto aplica-se especialmente a falha (por exemplo defeito, deficiência[79]) e a avaria (tal como anomalia ou mau funcionamento)

b) a atribuição feita aos termos falha, erro e avaria toma simplesmente em conta a utilização corrente: i) prevenção de falhas, tolerância a falhas e diagnóstico de falhas, ii) detecção e correcção de erros, iii) taxa de avarias.

Finalmente, deve ser enfatizado que as definições dadas nesta secção são de natureza *sintática*; de acordo com isto, foram realçados os critérios as várias classificações efectuadas, sendo em nosso entender mais importantes do que as próprias classes consideradas.

[79] O autor dá com exemplo o termo "*bug*" (insecto, normalmente conotado como uma falha de software) e faz notar em rodapé (NA12) que se deve considerar a especialização do termo, tal como em [Gra, 86], que distingue "Heisenbugs" (falhas de software intermitentes) de "Bohrbugs" (falhas de software permanentes, "tal como o átomo de Bohr, sólido, facilmente detectado através de técnicas normalizadas e portanto enfadonho ("*boring*")).

5. OS MEIOS PARA OBTER CONFIANÇA NO FUNCIONAMENTO

5.1. DEPENDÊNCIA ENTRE OS MEIOS PARA OBTER CONFIANÇA NO FUNCIONAMENTO

Os conceitos de "como obter confiança no funcionamento" que aparecem nas definições básicas dadas no capítulo 1, representam de facto objectivos[80] que não podem ser totalmente alcançados, dado que as actividades correspondentes são levadas a cabo por seres humanos, sendo portanto imperfeitas. Estas imperfeições levam a *interdependências* entre os meios para a obtenção de confiança no funcionamento, que explicam porque que é que só a utilização *combinada* destes meios – preferencialmente em cada passo do processo de concepção e implementação – pode conduzir a um sistema computacional em que se deposita confiança no funcionamento. Estas interdependências podem ser esquematizadas como se segue: apesar da prevenção de falhas por intermédio de metodologias de concepção e regras construtivas (forçosamente imperfeitas, de forma a poderem ser utilizadas), aparecem falhas[81]. Daqui a necessidade da supressão de falhas[82]. A supressão de falhas é, por si só, imperfeita, dado que se utilizam componentes "chave-na-mão" - de hardware ou software – no sistema, de onde aparece a importância da previsão de falhas. A crescente dependência nos sistemas computacionais leva à necessidade de tolerância a falhas, que se baseia por sua vez em regras construtivas; daqui a supressão de falhas, a previsão de falhas, etc. Deve notar-se que o processo é ainda mais recursivo do que aparenta do anterior: os sistemas computacionais actuais são tão complexos que o seu projecto e a sua implementação necessitam de ser apoiados por ferramentas computadorizadas, de forma a serem viáveis economicamente (num sentido lato, incluindo a capacidade de terem sucesso num prazo aceitável). As próprias ferramentas têm também de ser de confiança, e por aí fora.

O raciocínio anterior ilustra a forte interacção entre a supressão de falhas e a previsão de falhas, motivando o seu agrupamento num só termo – *validação*. Isto, apesar do facto da validação estar muitas vezes limitada à supressão de falhas e associada com uma das actividades principais envolvidas na supressão de falhas, a verificação, como por exemplo em "V and V" ([Boe, 79]); em tal caso, a distinção faz-se segundo a diferença entre "construir correctamente o sistema" (relativo a verificação) e "construir o sistema correcto" (relativo a validação)[83]. O que aqui se propõe é simplesmente uma extensão deste conceito: a resposta à questão "estarei a construir o sistema correcto?" (supressão de falhas) sendo complementada com "durante quanto

---

[80] O autor refere-se à prevenção de falhas, tolerância a falhas, supressão de falhas e previsão de falhas.

[81] [Lev, 94] justifica a necessidade de tolerância a falhas pelas limitações da prevenção de falhas, focando os seguintes pontos:

- a supressão *a priori* de todas as falhas de um sistema é impossível, na maioria dos casos; embora durante o projecto e a implementação se deva a cabo a supressão de todas as falhas conhecidas, não se podem eliminar falhas cuja existência ainda não se revelou para nós;
- tendo em conta que se satisfaz a prevenção de todas as falhas, o sistema resultante é pouco flexível, no sentido em que é não tem soluções para lidar com um comportamento incorrecto; isto pode levar a uma manutenção difícil, quando necessária, pois as restrições colocadas à intervenção podem não permitir efectuá-la;
- quando a prevenção de falhas não é bem sucedida, podem surgir atrasos imprevistos; estes atrasos podem tornar-se críticos em sistemas que não admitem avarias temporais, podendo desprezar-se em sistemas que utilizam mecanismos de penalidades.

Portanto, apesar da importância da prevenção de falhas conhecidas, não deve ser aceite como a única solução para lidar com o comportamento incorrecto dos sistemas.

[82] Na versão francesa, o autor refere ainda: "quando um erro é gerado durante a verificação, é necessário um diagnóstico que permita determinar a falha(s) causadora do erro, a fim de a suprir".

[83] (NA13) É digno de nota que estas atribuições são muitas vezes invertidas, tal como no domínio dos protocolos de comunicação (ver por exemplo [Rud, 85]).

tempo estará ele correcto?" (previsão de falhas)[84]. Adicionalmente, a supressão de falhas está normalmente fortemente associada com a prevenção de falhas, formando em conjunto a **evitação de falhas[85]**, isto é, como obter um sistema livre de falhas. Além de sublinhar a necessidade de validação dos procedimento e mecanismos da tolerância a falhas, é muito interessante considerar a supressão de falhas e a previsão de falhas como dois componentes da mesma actividade – validação – no sentido em que permite uma melhor compreensão da noção de cobertura[86], e portanto de um importante problema introduzido pela recursividade acima descrita: *a validação da validação*, ou como obter confiança nos métodos e ferramentas usados para obter confiança no funcionamento do sistema. A **cobertura** representa uma medida da representatividade das situações às quais o sistema é submetido durante a sua validação, comparadas com as situações reais com que ele será confrontado na sua vida operacional[87,88]. Uma cobertura imperfeita reforça a relação entre a supressão de falhas e a previsão de falhas, pois pode considerar-se que a necessidade de previsão de falhas deriva de uma cobertura imperfeita da supressão de falhas.

Na sequela deste capítulo, vamos examinar à vez a tolerância a falhas, a supressão de falhas e a previsão de falhas; a prevenção de falhas não é tratada pois relaciona-se nitidamente com a engenharia "genérica" de sistemas[89,90].

---

[84] (NA14) Validação deriva de "validade" que encapsula duas noções:
- validade num dado momento, que tem a ver com supressão de falhas;
- validade durante um dado tempo, que tem a ver com previsão de falhas.

[85] Tradução de "*fault avoidance*". [Joh, 96] considera que a *evitação de falhas* é uma técnica utilizada para tentar prevenir a ocorrência de falhas; pode incluir revisões de projecto, verificação de componentes, teste e outros métodos de controlo da qualidade. [Wei, 97] considera que existem dois meios para a obtenção de confiança no funcionamento de um sistema: a *evitação de falhas* e a *tolerância a falhas*. A evitação de falhas previne a ocorrência de falhas no sistema e inclui a prevenção de falhas, a supressão de falhas e a previsão de falhas.

[86] Tradução de "*coverage*".

[87] (NA15) A noção de cobertura tal como definida aqui é muito genérica, podendo ser especializada de acordo com o seu campo de aplicação, isto é:
- cobertura de um teste de software, no que respeita ao seu texto, gráficos de controlo, etc.
- cobertura de um teste de um circuito integrado, no que respeita a um modelo de falhas;
- cobertura da tolerância a falhas, no que respeita a uma classe de falhas;
- cobertura de uma hipótese de falha, relativamente à realidade.

[88] [Lap, 93], por exemplo, utiliza o termo "cobertura" relativamente à tolerância a falhas, no sentido em *as imperfeições da tolerância a falhas* se devem a uma falta de *cobertura da tolerância a falhas*, constituindo uma forte limitação à obtenção de confiança no funcionamento de um sistema computacional. Tais imperfeições devem-se a falhas de concepção afectando os mecanismos de tolerância a falhas no que respeita às hipóteses de falhas efectuadas durante a concepção (falta de *cobertura da manipulação de falhas e erros*) ou a hipóteses de falhas que diferem das falhas que realmente acontecem durante a operação do sistema (falta de *cobertura dos modos de falha*). [Kir, 87] define cobertura (da redundância de verificação) como "a probabilidade de detectar um erro dentro de um tempo útil".

[89] Apesar da prevenção de falhas não ser objecto de análise deste texto, é interessante salientar que por exemplo [Nel, 90] refere que a utilização de componentes de elevada qualidade, de metodologias formais de concepção e de provas da correcção da concepção são formas de prevenir falhas no funcionamento de um sistema computacional.

[90] Alguns exemplos poderão ilustrar melhor o conceito de prevenção de falhas:
- o procedimento levado a cabo nos voos comerciais, antes da descolagem, proibindo a utilização de aparelhagem electrónica, nomeadamente telemóveis e computadores pessoais é uma medida preventiva de falhas externas. A utilização inadvertida de um telemóvel, por exemplo, poderá levar a uma falha num dos sistemas de controlo da navegação ou de telecomunicação, sendo esta uma falha externa, humana, intencional, mas sem dolo.
- O aviso automático da não utilização do cinto de segurança, em alguns automóveis, pode considerar-se como uma medida de prevenir falhas externas/internas.
- Uma medida para prevenir falhas poderá também ser o tipo específico de botão utilizado para reinicializar um computador pessoal ou uma máquina

5.2. TOLERÂNCIA A FALHAS

A tolerância a falhas ([Avi, 67]) é levada a cabo através do processamento de erros e pelo tratamento das falhas ([And, 81])[91]. O **processamento de erros** visa eliminar os erros de um sistema computacional, se possível antes da ocorrência de uma avaria; o **tratamento de falhas** destina-se a impedir a reactivação das falhas.

---

de calcular programável (o conhecido botão de "*reset*"). Normalmente, este botão está numa posição "não muito acessível", de modo a evitar que o utilizador o pressione involuntariamente, com qualquer parte do corpo.

[91] [Joh, 96] defende que a tolerância a falhas pode ser conseguida por muitas técnicas. A dissimulação de falhas é uma abordagem para tolerar falhas. Outra solução é detectar e localizar a falha, reconfigurando o sistema de forma a remover o componente defeituoso. *Reconfiguração* é o processo de eliminar uma entidade defeituosa de um sistema e restaurar o sistema para um estado operacional. Se for utilizado a técnica da reconfiguração, então dever-se-ão tomar em conta:

- *detecção de falhas*: é o processo de reconhecer a ocorrência de uma falha;
- *localização de falhas*: é o processo de determinar onde ocorreu a falha, de modo a que se possa levar a cabo uma recuperação adequada;
- *contenção de falhas*: é o processo de isolar uma falha, prevenindo que os seus efeitos se propaguem ao resto do sistema;
- *recuperação de falhas*: é o processo de permanecer operacional ou de manter o estado operacional através de uma reconfiguração, na presença de falhas.

No nosso parecer, a detecção de falhas não é um conceito correcto, dado que não são as falhas que se detectam, mas sim os erros (nunca é possível detectar uma falha se ela não resultar num estado erróneo do sistema). [Joh, 96] refere ainda, sem grande consistência no nosso entender, que os conceitos anteriores são equivalentes aos existentes para o *universo da informação*, mais especificamente:

- *detecção de erros*: é o processo de reconhecer a ocorrência de um erro;
- *localização de erros*: é o processo de determinar em que módulo específico é que ocorreu o erro;
- *contenção de erros*: é o processo evitar que o erro se propague ao resto do sistema;
- *recuperação de erros*: é o processo recuperar o estado operacional ou restaurar a integridade do sistema depois da ocorrência de um erro.

Estes procedimentos já nos parecem mais correctos, estando mais de acordo com os conceitos de processamento de erros e de tratamento de falhas defendidos neste documento, bem como em [Lap, 95a], [Lap, 95b] e [Lap, 98].

[Wei, 97] defende que a tolerância a falhas pode ser aplicada em três níveis distintos:

- *tolerância a falhas de hardware*: redundância nas comunicações, processadores replicados, memória adicional, fontes de alimentação redundantes;
- *tolerância a falhas de software*: blocos de recuperação, múltiplas versões de programas);
- *tolerância a falhas do sistema*: medidas específicas para cada aplicação, implementadas em *hardware* ou *software*).

O *processamento de erros* pode ser efectuado de duas maneiras[92]:

- **recuperação de erros**, onde um estado erróneo é substituído por um estado isento de erros[93]; esta substituição pode tomar duas formas ([And, 81]):
  - **recuperação para trás**, consiste em fazer o sistema voltar a um estado pelo qual o sistema já passou anteriormente, antes da ocorrência do erro; isto envolve a definição de pontos de recuperação, que são pontos no tempo durante a execução de um processo, para os quais o seu estado pode posteriormente ser restaurado[94];
  - **recuperação para a frente**, consiste em procurar um novo estado a partir do qual o sistema possa funcionar (frequentemente em modo degradado[95]).
- **compensação de erros**, onde o sistema dispõe de redundância suficiente para permitir que o serviço(s) prestado a partir de um estado erróneo esteja contudo isento de erros[96].

Quando se recorre à recuperação de erros, o estado erróneo necessita de ser (urgentemente) identificado com sendo erróneo, antes de o substituir; isto é o objectivo da **detecção de erros**, daqui o termo usualmente empregue de *detecção e recuperação de erros*. A associação num componente da sua funcionalidade de processamento, juntamente com mecanismos de detecção de erros, leva à noção de **componente auto-testável**, tanto em hardware ([Car, 68], [Wak, 78], [Nic, 89]) como em software ([Yau, 75], [Lap, 90a]); um dos maiores benefícios dos componentes auto-testáveis a possibilidade de dar uma clara definição da área limite do erro ([Sie, 82]). Quando se efectua a compensação de um erro num sistema composto por componentes auto-testáveis, organizados em classes que executam as mesmas tarefas, então a substituição de estados não é mais do que substituir, numa dada classe, um componente defeituoso por um não defeituoso. Aparece desta forma a correspondente abordagem à tolerância a falhas: *detecção e compensação de*

---

[92] O autor evoluiu em ([Lap, 95a], [Lap, 95b], [Lap, 98]) defendendo que o processamento de erros deve reger-se por *três* primitivas:

- **Detecção de erros**: que permite que o estado erróneo seja identificado como tal;
- **Diagnóstico de erros**: que permite a avaliação dos danos causados pelo erro detectado, ou por erros propagados antes da detecção;
- **Recuperação de erros**: onde um estado erróneo é substituído por um estado livre de erros; esta substituição pode tomar *três* formas:
  - recuperação para trás (definição equivalente à acima efectuada);
  - recuperação para a frente (definição equivalente à acima efectuada);
  - compensação: onde o estado erróneo contém redundância suficiente para permitir a sua transformação num estado isento de erros (definição ligeiramente diferente do corrente texto).

Esta ideia parece endereçar melhor o conceito de *recuperação de erros*, pois, no fundo, o objectivo de uma *compensação* também é a *recuperação* de um erro.

[93] O autor tem aqui uma pequena gralha, pois refere "*where an error-free state is substituted for the erroneous state*", quando é o contrário, pois a recuperação de um erro (implicando necessariamente a prévia detecção do mesmo), consiste em conduzir um sistema que se encontra num estado declaradamente erróneo a um estado considerado correcto ([Mag, 95]).

[94] Esta definição está um pouco confusa. [Mag, 95] refere que "a *recuperação para trás* consiste em, sempre que é detectado um erro, fazer *regredir* ("*rollback*") a execução do processo supostamente responsável pela sua introdução até um determinado ponto, denominado de *ponto de recuperação*, repor o sistema computacional, ou parte deste, no estado em que se encontrava aquando da passagem do processo em causa por esse ponto e proceder, a partir daí, ou à reexecução desse processo, ou à execução de uma sua réplica ([Bow, 93]). Um **ponto de recuperação** é portanto um *ponto* pelo qual um processo passou anteriormente e no qual não se constatou, em devido tempo, a existência de qualquer erro".

[95] Foi já referido que o modo de funcionamento passa a *degradado* se os recursos sobreviventes deixarem de ser suficientes para a prestação do serviço(s) *nominal*.

[96] Aqui ficaria melhor "de acordo com a especificação" em vez de "isento de erros", para não utilizar o termo "erro" quando a não concordância de um serviço prestado com a especificação, quer no domínio do valor, quer no domínio do tempo, se denomina de "avaria".

erros[97]. Por outro lado, a compensação pode ser aplicada sistematicamente, mesmo na ausência de erros, proporcionando assim uma **dissimulação de falhas**[98] (por exemplo, por voto maioritário). Contudo isto pode levar a uma perda desconhecida de redundância[99]. Portanto, as implementações práticas da dissimulação de erros geralmente envolvem detecção, que pode ser levada a cabo *depois* das substituição do estado[100].

As recuperações para trás e para a frente podem ser utilizadas conjuntamente: a recuperação para trás pode ser tentada primeiro; se o erro persistir, pode então tentar-se a recuperação para a frente. Na recuperação para a frente é necessário *avaliar os danos* causados pelo erro detectado, ou pelos erros propagados antes da detecção. No caso da recuperação para trás, a avaliação dos danos pode – em princípio – ser ignorada, desde que os mecanismos que permitem a substituição do estado erróneo num estado isento de erros não tenham sido afectados ([And, 81]).

O acréscimo de tempo de execução necessário para o processamento dos erros é radicalmente diferente, dependendo da forma de processamento de erros adoptada:

- na recuperação de erros, o acréscimo de tempo é maior quando ocorre um erro que antes dele ocorrer; no caso da recuperação para trás, este refere-se ao estabelecimento de pontos de recuperação, preparando o sistema para poder processar os erros;
- na compensação de erros, o acréscimo temporal necessário é o mesmo, ou quase o mesmo, independentemente se aparecem erros ou não[101].

Adicionalmente, a duração da compensação de erros é muito menor que a duração da recuperação de erros, devido à maior quantidade de redundância (estrutural). Esta observação:

a) é de elevada importância prática, pois condiciona, em muitas situações, a escolha de uma estratégia de tolerância a falhas, no que diz respeito à granulosidade temporal[102];

b) introduz uma relação entre o acréscimo de tempo de execução e a redundância estrutural; mais genericamente, um sistema redundante fornece sempre um comportamento redundante, implicando pelo menos um pequeno acréscimo de tempo de execução; este acréscimo de tempo pode ser suficientemente pequeno para não ser detectado pelo utilizador, o que apenas significa que o *serviço* não é

---

[97] (NA16) A detecção e compensação de erros pode ser vista como um caso limite da detecção e recuperação de erros, onde a recuperação é efectuada utilizando o presente estado (erróneo) do sistema, em vez de substituir o estado erróneo por um estado isento de erros.

[98] Tradução de "*fault masking*".

[99] Por exemplo, na redundância modular tripla, se um dos componentes redundantes falhar, o sistema não avaria (pois os outros dois são maioritários), mas perde redundância.

[100] [Nel, 90] refere que uma estratégia de tolerância a falhas inclui um ou mais dos seguintes componentes:

- *dissimulação*: correcção dinâmica dos erros gerados;
- *detecção*: detecção de um erro – o sintoma de uma falha;
- *contenção*: prevenção da propagação de um erro além de limites definidos;
- *diagnóstico*: identificação do módulo defeituoso, responsável por um erro detectado;
- *reparação/reconfiguração*: eliminação ou substituição de um componente defeituoso, ou de um mecanismo para o ultrapassar;
- *recuperação*: correcção do sistema para um estado aceitável de modo a que continue o seu funcionamento.

É pena que, embora referidos, estes conceitos tenham sido desenvolvidos de uma forma clara e estruturada.

[101] (NA17) Em ambos os casos, deve adicionar-se ao acréscimo temporal o tempo necessário para actualizar os registos do estado do sistema.

[102] É o caso dos sistemas de tempo-real, nomeadamente os críticos, onde na maior parte dos casos não se pode esperar que o sistema recupere (a substituição dos estados leva tempo, precioso neste tipo de sistemas), sendo necessária uma "imediata" compensação dos erros.

redundante; no extremo oposto está a "redundância temporal" (*comportamento* redundante obtido por repetição) que necessita de ser pelo menos iniciado por redundância estrutural, limitada mas presente; de forma simplista, quanto mais redundância estrutural se utilizar, menor será o acréscimo de tempo de execução.

O primeiro passo no *tratamento das falhas* é o **diagnóstico das falhas**, que consiste na determinação da causa(s) do erro(s), tanto em termos de localização como em termos de natureza. Seguidamente vêm as acções destinadas a cumprir o principal objectivo do tratamento das falhas: prevenindo que a(s) falha(s) sejam de novo activadas, tornando-a(s) pois passiva(s), i.e., a **desactivação de falhas**[103]. Isto é levado a cabo removendo o componente(s) considerado como defeituoso de execuções seguintes. Se o sistema não for capaz de prestar o mesmo serviço que anteriormente, poderá tomar lugar uma *reconfiguração*[104].

Se se estimar que o processamento de erros pode remover directamente a falha, ou se a probabilidade de voltar a ocorrer é suficientemente pequena, então não é necessário desactivar as falhas. Desde que não se leve a cabo a desactivação de falhas, pode ver-se a falha como uma **falha ténue**[105]; se, por outro lado, se leva a cabo a desactivação, a falha é considerada uma **falha acentuada**[106]. À primeira vista, as noções de falhas ténue e acentuada podem ser vistas como sinónimos, respectivamente, das noções anteriormente introduzidas de falhas temporárias e permanentes. De facto, a tolerância de falhas temporárias não necessita de tratamento de falhas, dado que, neste caso, a recuperação do erro eliminaria os efeitos da falha, que por sua vez desapareceu, desde que o processo de propagação não tenha criado uma falha permanente. Em suma, as noções de falhas ténue e acentuada são úteis devido às seguintes razões:

- distinguir uma falha permanente duma falha temporária é uma tarefa difícil e complexa, dado que a) uma falha temporária desaparece após um certo tempo, normalmente antes de ser efectuar o diagnóstico de falha e b) falhas de diferentes classes podem levar a erros muito similares[107]; portanto, a noção de falha ténue e acentuada incorpora de facto a subjectividade associada a estas dificuldades, incluindo o facto que uma falha pode ser declarada como uma falha ténue quando o diagnóstico da falha não tiver êxito;
- a capacidade destas noções incorporarem subtilezas dos modos de acção de algumas falhas transitórias; por exemplo, poderá ser dito que a falha interna dormente resultante da acção de partículas alfa (devido à ionização residual da embalagem dos circuitos), ou de iões pesados no espaço, em elementos de memória (no sentido lato do termo, incluindo *flip-flops*) é uma falha *temporária*? Tal falha dormente é, contudo, uma falha ténue.

As definições anteriores aplicam-se tanto às falhas físicas como às falhas de concepção: as classes de falhas que podem realmente ser toleradas dependem das hipóteses de falhas[108] que foram consideradas na fase de concepção,

---

[103] Tradução de "*fault passivation*", pois não existe o termo "passivação" em português. Curiosamente, o termo "*passivation*" também não existe em Inglês, pelo menos em [Lon, 95].

[104] O autor não define o que é uma *reconfiguração*. Posteriormente ([Lap, 95a], [Lap, 95b], [Lap, 98]), o autor define reconfiguração como a modificação da estrutura de um sistema de forma a que os componentes não avariados permitam prestar um serviço "aceitável", apesar de degradado; uma reconfiguração pode implicar o abandono de certas tarefas ou a atribuição de tarefas aos componentes não avariados. [Joh, 96] define reconfiguração como o processo de eliminar de um sistema uma entidade defeituosa e colocar o sistema num estado/condição operacional.

[105] Tradução de "*soft fault*".

[106] Tradução de "*hard or solid fault*".

[107] Tentar dar um exemplo...

[108] Inexplicavelmente, o autor não dá uma definição de *hipótese de falhas*. [Kop, 93] refere que a **hipótese de falhas** define os tipos e a frequência das falhas que um sistema tolerante a falhas deve ser capaz de gerir. Se o cenário de falhas inicialmente

dependendo portanto da *independência* das redundâncias no que respeita ao processo de criação e activação das falhas. Um exemplo poderá ser considerar a tolerância a falhas físicas e a tolerância a falhas de concepção. Um método (largamente utilizado) para conseguir tolerância a falhas é efectuar processamentos múltiplos através de múltiplas vias[109]. Quando se prevê a tolerância a falhas físicas, estas vias podem ser idênticas, assumindo que os componentes de hardware avariam de forma independente. Tal abordagem não é adequada à tolerância a falhas de concepção onde as diferentes vias têm de providenciar *serviços idênticos* através de *concepções separadas* ([Elm, 72], [Ran, 75], [Avi, 78]), i.e., através de uma **concepção diversificada**[110] ([Avi, 84]).

identificado se modificar, o sistema deve ainda fornecer um nível especificado de serviço. Se o meio envolvente gerar mais falhas do que foi especificado na hipótese de falhas, então mesmo um sistema tolerante a falhas pode avariar. O pior cenário que um sistema tolerante a falhas deve ser capaz de gerir é se ocorrerem ao mesmo tempo um *pico de carga* e o número máximo de falhas. O *pico de carga* (que se assume que seja gerado pelo meio envolvente do sistema) é definido pela **hipótese de carga**, e pode ser expresso pelo mínimo intervalo de tempo entre – ou a taxa máxima de – cada *transacção* (definida como a sequência de operações entre um estímulo do meio envolvente e uma resposta do sistema).

[109] Executar o mesmo cálculo em três processadores diferentes, por exemplo.

[110] Tradução de "*design diversity*". Em [Mag, 95], o autor por traduzir o termo "*design fault*" como "falha de concepção" e depois utiliza o termo "projecto diversificado" para traduzir "*design diversity*", o que não parece correcto (ver nota 44). De qualquer forma, essa referência explica bem o termo citando que, relativamente à utilização de redundância estrutural, "nem sempre a réplica de um componente é em tudo semelhante ao original. Muitas vezes, pese embora naturalmente sujeito às mesmas especificações, um componente de reserva tem uma concepção e/ou materialização diferente da do que pretende substituir. Neste caso, diz-se que o sistema teve um projecto (concepção) diversificado. Esta estratégia é indispensável quando um modelo de falhas inclui falhas de concepção e, muito em particular, falhas com origem no *software*". [Lap, 93] refere que, devido ao custo elevado da concepção diversificada, a tolerância falhas de *software* quase que se limita, actualmente, a algumas aplicações de segurança crítica, tais como a

Nos sistemas tolerantes a falhas, encontramos frequentemente situações envolvendo múltiplas falhas e/ou avarias. Do ponto de vista das suas causas, podemos distinguir:

- **falhas independentes**, que são atribuídas a causas diferentes;
- **falhas relacionadas**, que são atribuídas a uma única causa.

Como exemplos de falhas relacionadas poderemos citar fontes de alimentação, relógios[111] e especificações (únicas). Numa concepção diversificada, as falhas relacionadas podem também resultar de dependências nos projectos e implementações separados. As falhas relacionadas provocam normalmente **avarias de modo-comum**.

Outra noção útil na área dos sistemas tolerantes a falhas é a de **erros coincidentes**, i.e. erros criados pelo mesmo estímulo ([Avi, 86]).

A relação temporal entre avarias múltiplas leva-nos a distinguir:

- **avarias simultâneas**, que ocorrem dentro de uma determinada janela temporal;

aeronáutica, controlo de centrais nucleares ou os transportes ferroviários. A maior limitação ao comprovado melhoramento da confiança no funcionamento trazido pela concepção diversificada é constituída pelas inevitáveis correlações entre as variantes de *software* resultantes das diversas concepções. *Variantes* são os diferentes sistemas produzidos a partir de uma única especificação de serviço ([Lap, 90a]). Uma concepção diversificada tem pelo menos duas variantes e um votador, que monitoriza os resultados da execução das variantes. No âmbito da tolerância a falhas de *software*, as variantes são também denominadas de "alternativas" ("*alternates*"), "réplicas" ("*replicas*") ou de "versões" ("*versions*") (consultar [Lev, 94]). Um exemplo de concepção diversificada é o sistema de air-bag e pretensores, no automóvel: de forma a detectar a desaceleração, utilizam-se normalmente dois sensores: um piezoeléctrico e um de mercúrio.

[111] Se um relógio avariar (deixar de prestar o serviço pretendido), poderá provocar a criação de várias falhas em diferentes pontos de um sistema computacional (processamento, sincronização de comunicações, leitura de sensores, etc.). Estas falhas são *falhas relacionadas*, pois são atribuídas a uma única causa.

- **avarias sequenciais**, que não ocorrem dentro desse intervalo de tempo.

Embora a definição matemática de "simultâneo" tornasse nula a janela temporal, na prática a janela poderá ser razoavelmente grande, dependendo da aplicação. Daqui a noção de avarias "quase coincidentes" ser algumas vezes empregue ([Tri, 84]). Tipicamente, num sistema tolerante a falhas que tenha sido concebido para tolerar uma falha de cada vez, é necessário recuperar dos efeitos de uma falha antes do sistema poder tolerar a próxima falha. Neste contexto, a janela temporal que separa as avarias simultâneas das avarias sequenciais é o intervalo de tempo necessário para o processamento do erro e possivelmente para o tratamento da falha, durante o qual o sistema está vulnerável.

Um aspecto importante na coordenação das actividades de múltiplos componentes é o de evitar que a propagação dos erros afecte o funcionamento dos componentes onde não ocorreram falhas. Este aspecto torna-se particularmente importante quando um dado componente necessita de transmitir informação a outros componentes, informação essa que só esse componente detém. Exemplos típicos de tal *informação de fonte única* são dados locais de sensores, o valor de um relógio local, o ponto de vista local do estado de outros componentes, etc. A consequência desta necessidade de transmitir informação de fonte única de um componente para outros componentes é que os componentes que não falharam têm de chegar a um *acordo* de forma a que a informação que eles obtiveram seja empregue de uma forma mutuamente consistente. Foi dedicada uma atenção especial a este problema no campo dos sistemas distribuídos (ver por exemplo a difusão atómica ([Cri, 85a]), a sincronização de relógios ([Lam, 85], [Kop, 87]) ou protocolos de grupo[112] ([Cri, 88])). É importante perceber, contudo, que a presença inevitável de redundância estrutural em qualquer sistema tolerante a falhas[113] implica distribuição[114], a qualquer nível, mantendo-se o problema do acordo mutuo. Os sistemas tolerantes a falhas geograficamente localizados podem utilizar soluções para o problema do acordo, consideradas demasiado caras num sistema distribuído "clássico" de componentes comunicando por mensagens (por exemplo, andares intermédios ([Lal, 86]) ou andares múltiplos ([Fri, 82]) para assegurar a coerência interactiva).

O conhecimento de certas propriedades de um sistema pode permitir limitar a redundância, levando à chamada "tolerância a falhas de baixo custo". Exemplos destas propriedades são regularidades de natureza estrutural: códigos de detecção e correcção de erros ([Pet, 72]), estruturas de dados robustas ([Tay, 80]), multiprocessadores e redes de computadores ([Pra, 86], [Ren, 86]), tolerância a falhas baseada em algoritmos ([Hua, 82]). As falhas que são toleradas são portanto dependentes das propriedades consideradas na concepção, pois estas intervêm directamente na hipótese de falhas.

A sinalização da avaria de um componente aos seus utilizadores é de particular importância. Isto pode ser tomado em consideração na base das *excepções* ([Mel, 77], [Cri, 80], [And, 81]). As capacidades de *tratamento das excepções*[115] disponibilizadas em certas linguagens podem ser uma forma conveniente de implementar a recuperação de erros, especialmente a recuperação para a frente[116].

---

[112] Tradução de "*membership protocols*".

[113] Isto é muito discutível, pois existem outros tipos de redundância, tal como a redundância temporal, embora exista **quase sempre** alguma redundância estrutural. Ficaria melhor "a presença usual de redundância estrutural".

[114] Na tradução francesa o autor utiliza "..implica uma repartição dos recursos,...".

[115] Tradução de "*exception handling*".

[116] (NA18) A utilização do termo "excepção", devido à sua origem de resistir a situações excepcionais – não apenas erros – deve ser cuidadosamente utilizado no contexto da tolerância a falhas: poderia parecer contraditório ao facto de a tolerância a falhas poder ser considerada como um atributo natural dos sistemas

A tolerância a falhas é (também) um conceito recursivo: é essencial que os mecanismos destinados a implementar a tolerância a falhas sejam protegidos das falhas que os podem afectar. Exemplos disto são a replicação de votadores, dispositivos de teste auto-testáveis[117] ([Car, 68]), memória "estável" para os programas e dados de recuperação ([Lam, 81]).

A tolerância a falhas não se restringe a falhas acidentais. Tradicionalmente, a protecção contra intrusões envolve criptografia ([Den, 82]). Alguns mecanismos de detecção de erros são vocacionados ao mesmo tempo para falhas acidentais e intencionais (técnicas de protecção no acesso à memória, por exemplo) e foram propostas abordagens para a tolerância tanto a intrusões como a falhas físicas ([Fra, 86][Rab, 89]), bem como para tolerar lógica maliciosa ([Jos, 88]).

5.3. SUPRESSÃO DE FALHAS

A supressão de falhas é composta por três etapas: verificação, diagnóstico e correcção. A **verificação** consiste em verificar se o sistema satisfaz certas propriedades, denominadas de *condições de verificação* ([Che, 81]); se isso não acontecer, têm de ser dados os dois passos seguintes: diagnosticar a falha(s) que impediram as condições de verificação de serem cumpridas, e depois efectuar as correcções necessárias. Após a correcção, o processo deve ser recomeçado, de forma a assegurar que a supressão da falha não teve consequências indesejáveis; a verificação efectuada nesta fase é usualmente denominada de **verificação** (não) **regressiva**. As condições de verificação podem tomar duas formas:

- condições gerais, que se aplicam a uma dada classe de sistemas e são por isso – relativamente – independentes da especificação, por exemplo a ausência de bloqueios[118] e a conformidade com as regras de projecto e implementação;
- condições específicas ao sistema considerado, directamente deduzidas da sua especificação.

As técnicas de verificação podem ser classificadas dependendo envolvem ou não a activação do sistema. A verificação de um sistema sem o activar é uma **verificação estática**. Este tipo de verificação pode ser levado a cabo:

- no próprio sistema, na forma de a) uma análise estática (inspecções ([Mye, 79]), análise do fluxo de dados ([Ost, 76]), análise de complexidade ([McC, 76]), verificações efectuadas pelos compiladores, etc.) ou b) de uma prova de correcção (provar por indução matemática ([Hoa, 69], [Cra, 87]));
- num modelo do comportamento do sistema (Redes de Petri e autómatos de estados finitos, por exemplo), levando à *análise de comportamento* ([Dia, 82]).

Quando a verificação de um sistema passa pela sua activação, trata-se de uma **verificação dinâmica**; as entradas fornecidas ao sistema podem ser simbólicas, no caso da **execução simbólica**, ou estimadas no caso de testes de verificação, habitualmente designadas apenas por **teste**.

O teste exaustivo de um sistema considerando todas as entradas possíveis é impraticável, em termos genéricos. Os métodos utilizados para a determinação dos padrões de teste podem ser classificados de acordo com dois pontos de vista: o critério para a escolha das entradas de teste e a geração das entradas de teste.

O *critério* para a selecção das entradas de teste podem, por sua vez, ser decompostos em três pontos de vista:

---

computacionais, tomada em conta desde a fase inicial do projecto, e não como um atributo "excepcional".

[117] Tradução de "*self-checking checkers*".

[118] Tradução de "*deadlock*".

- o objectivo do teste: verificar se o sistema satisfaz a sua especificação (funcional) é um **teste de conformidade**, enquanto o teste destinado a revelar falhas é um **teste de procura de falhas**;
- o modelo do sistema: dependendo se o modelo do sistema é relativo à função ou à estrutura do sistema, leva respectivamente a um **teste funcional** ou a um **teste estrutural**;
- a existência de um modelo de falhas: se tal modelo existe, pode ser levado a cabo um **teste baseado em falhas** ([Mor, 90]), destinado a revelar classes específicas de falhas (por exemplo, falhas de colagem na produção de hardware ([Rot, 67]), falhas físicas afectando o conjunto de instruções de um microprocessador ([Tha, 78]), falhas de concepção em software ([Goo, 75][DeM, 78][How, 87]); se não existir um modelo de falhas, o critério[119] pode relacionar-se com a sensibilização de caminhos ([Rap, 85][Nta, 88]), com os valores limite das entradas em software ([Mye, 79]), etc.[120]

A *geração* das entradas de teste pode ser determinística ou probabilística:

- no **teste determinístico**, os padrões de teste são predeterminados por uma escolha selectiva, de acordo com o critério adoptado;
- no **teste aleatório** ou **estatístico**, os padrões de teste são seleccionados segundo uma distribuição de probabilidade do domínio de entrada; a distribuição e a quantidade de dados de entrada são determinados de acordo com o critério adoptado ([Dav, 81], [Dur, 84]).

[119] Refere-se ao critério de selecção das entradas de teste.

[120] (NA19) A possibilidade de definir um modelo de falhas está fortemente relacionada com o estágio do processo de concepção que é considerado: quanto mais avançado, maior a possibilidade de definir um modelo de falhas.

A combinação dos diversos pontos de vista leva a um determinado número de abordagens de teste, podendo algumas delas receber uma denominação condensada, como por exemplo:

- o teste estrutural, quando aplicado a hardware, geralmente significa procura de falhas, estrutural e baseado em falhas, enquanto que aplicado a software, geralmente significa procura de falhas, estrutural e não baseado em falhas;
- o teste de mutações[121] de software () é um teste de procura de falhas, estrutural, baseado em falhas e determinístico.

Observar as saídas de teste e decidir se são ou não satisfatórias é conhecido como o problema do *oráculo*[122] ([Adr, 82]). As condições de verificação podem aplicar-se a todo o conjunto de saídas ou apenas a um seu subconjunto (por exemplo, a assinatura[123], quando se testam falhas físicas em hardware ([Dav, 86]) ou um "oráculo parcial", quando se testam falhas de concepção no software ([Wey, 82]). Quando o teste concerne as falhas físicas, os resultados – comprimidos ou não – antecipados do sistema sob teste para uma dada sequência de entrada, são determinados por simulação ([Lev, 86]) ou a partir de um sistema de referência (sistema padrão). Para as falhas de concepção, a referência é normalmente a especificação; poderá também ser um protótipo, ou outra implementação da mesma especificação, no caso da concepção diversificada ("teste frente a frente"[124], ver [Bis, 88]).

Alguns métodos de verificação podem ser utilizados em conjunção. Por exemplo, a execução simbólica pode ser utilizada a) para facilitar a

[121] Tradução de "*mutation testing*".

[122] Segundo [Por, 98], resposta dada por uma divindade a quem a consultava; a própria divindade; lugar onde se davam os oráculos; profecia; revelação; resposta infalível; pessoa cujo conselho tem grande autoridade.

[123] Tradução de "*signature*", termo muito utilizado no teste de circuitos integrados, nomeadamente microprocessadores.

[124] Tradução de "*back-to-back testing*".

determinação dos padrões de teste ([Adr, 82]), ou b) como um método de prova de correcção ([Car, 78]).

Como a verificação tem de ser efectuada ao longo de toda a concepção do sistema, as técnicas acima descritas aplicam-se naturalmente às várias formas que um sistema toma durante a sua concepção: protótipo, componente, etc.

Verificar que um sistema não é capaz de fazer mais do que foi especificado é particularmente importante, no que respeita às falhas intencionais ([Gas, 88]).

Conceber um sistema de forma a facilitar a sua verificação é a **concepção tendo em vista a verificação**[125]. Esta abordagem é particularmente desenvolvida para o hardware, no que diz respeito às falhas físicas, onde as técnicas correspondentes são denominadas de *concepção tendo em vista o teste*[126] ([Wil, 83], [McC, 86]).

A supressão de falhas durante a fase operacional da vida de um sistema é a **manutenção correctiva**, destinada a preservar ou melhorar a capacidade do sistema para prestar um serviço de acordo com a especificação[127]. Este tipo de manutenção pode tomar duas formas:

- manutenção curativa, destinada a remover falhas que foram produzidas por um ou mais erros e foram sinalizadas;
- manutenção preventiva, destinada a remover falhas antes destas produzirem erros; estas falhas podem ser:
  - falhas físicas que apareceram desde a última manutenção preventiva;
  - falhas de concepção que conduziram a erros em outros sistemas similares ([Ada, 84]).

Estas definições[128] aplicam-se igualmente, tanto aos sistemas não tolerantes a falhas como aos sistemas tolerantes a falhas, cuja manutenção pode ser feita em funcionamento[129] (sem interromper a prestação do serviço) ou com o sistema parado[130]. Finalmente, deve notar-se que a fronteira entre a manutenção correctiva e o tratamento de falhas é relativamente arbitrária; em particular, a manutenção curativa pode ser considerada como um – último – meio para obter tolerância a falhas.

5.4. PREVISÃO DE FALHAS

A previsão de falhas é levada a cabo através de uma avaliação do comportamento do sistema, relativamente à ocorrência ou activação de falhas. A avaliação pode ser:

- *não probabilística*, tal como na determinação dos caminhos mínimos numa árvore de falhas, ou a análise do modo e efeito das falhas;

---

[125] Tradução de "*design for verifiability*".

[126] Tradução de "*design for testability*".

[127] (NA20) As outras formas de manutenção que normalmente se distinguem são ([Ram, 84]):

- *manutenção adaptável*, que ajusta o sistema de acordo com mudanças ambientais (por exemplo, mudança de sistema operativo ou de sistema de gestão de bases de dados);
- *manutenção perfectiva*, que melhora a função do sistema respondendo a alterações definidas pelo utilizador ou projectista, podendo envolver a supressão das falhas de especificação.

[128] (NA21) É digno de nota que nas discussões actuais sobre a não relevância do uso do termo "manutenção" quando aplicado a software, esquece-se simplesmente a etimologia da palavra: na Idade Média, a manutenção designava as acções efectuadas para manter um exército em combate, incluindo portanto as formas correctiva, adaptável e perfectiva de manutenção. A associação de manutenção à reparação de hardware é, na verdade, um desvio (recente); associar "manter" com a noção de *serviço*, permitiria fazer renascer o seu significado etimológico, eliminando ao mesmo tempo o foco da discussão.

[129] Tradução de "*on-line*".

[130] Tradução de "*off-line*".

- *probabilística*, destinada a determinar a conformidade do sistema com os objectivos de confiança no funcionamento, expressa em termos de probabilidades associadas com alguns dos atributos da confiança no funcionamento, que podem então ser expressos como *medidas* de confiança no funcionamento[131].

[131] Isto é particularmente verdade, pois tem sido dedicada uma atenção especial à quantificação dos atributos da confiança no funcionamento. [Kop, 93], [Mag, 95] e [Joh, 96], por exemplo, quantificam os atributos da confiança no funcionamento utilizando as seguintes medidas:

- *fiabilidade*: é a medida do contínuo fornecimento de um serviço correcto, medida pela probabilidade *R(t)* que o sistema ainda estará a prestar um serviço correcto depois de um intervalo de tempo *t*, se estava a funcionar correctamente em $t = 0$; em outras palavras, a fiabilidade é a probabilidade de o sistema estar a funcionar correctamente durante um determinado intervalo de tempo;
- *segurança*: é a probabilidade *S(t)* de que um sistema não tem uma avaria catastrófica dentro de um intervalo de tempo *t*; segurança é uma medida da capacidade de um sistema avariar-seguro;
- *disponibilidade*: é a probabilidade *A(t)* de um sistema estar a funcionar correctamente e estar disponível para executar a sua função no instante de tempo *t*;
- *capacidade de manutenção* (apenas em [Kop, 93] e [Joh, 96]): medida da facilidade com que se efectua a manutenção de um sistema, medida pela probabilidade *M(t)* de o sistema ser restaurado num instante de tempo *t*, se avariou no instante de tempo $t = 0$;
- *capacidade de desempenho* (apenas em [Joh, 96]): em muitos casos, é possível conceber sistemas que continuam a funcionar correctamente depois da ocorrência de falhas de hardware ou software, mas o desempenho diminui, de alguma forma; a capacidade de desempenho é a probabilidade *P(L,t)* de que o desempenho do sistema esteja no nível *L*, ou acima, no instante de tempo *t*;

Existem ainda dois atributos da confiança no funcionamento que não podem ser facilmente medidos:

- *capacidade de teste* (apenas em [Joh, 96]): é a capacidade de se poderem testar certos atributos de um sistema; a medida da capacidade de teste permite avaliar a facilidade com que certos testes podem ser levados a cabo;

A vida de um sistema é percebida pelo seu utilizador(es) como uma alternância entre dois estados do serviço prestado, relativamente à especificação:

- **serviço correcto**, onde o serviço prestado *está conforme* com a especificação[132];
- **serviço incorrecto**, onde o serviço prestado *não está conforme* a especificação.

Uma avaria é então uma transição de um serviço correcto para um serviço incorrecto, sendo a transição de um serviço incorrecto para um serviço correcto denominada de **restauração**. Quantificar a alternância da prestação de um serviço correcto-incorrecto permite definir a fiabilidade e a disponibilidade como atributos da confiança no funcionamento:

- **fiabilidade**: é uma medida da continuidade da prestação de um serviço correcto ou, de forma equivalente, o tempo para a avaria;
- **disponibilidade**: é uma medida da prestação do serviço correcto, relativamente à alternância entre serviço correcto e incorrecto.

Uma terceira medida normalmente considerada é a **capacidade de manutenção**[133], que pode ser definida como uma medida do tempo de restauração desde a última avaria ocorrida ou, de forma equivalente, uma medida da continuidade da prestação de um serviço incorrecto.

- *inviolabilidade* (apenas em [Kop, 93]): é a capacidade de um sistema prevenir o acesso ou manipulação não autorizados de informação.

[132] (NA22) A utilização do termo "correcto" foi deliberadamente restringida ao serviço prestado por um sistema, não o utilizando para caracterizar o próprio sistema: no nosso ponto de vista, não existem sistemas sem falhas, apenas existem sistemas que ainda não avariaram.

[133] Tradução de "*maintainability*". Não existe um vocábulo correspondente em Português. Curiosamente, não existe a palavra inglesa, pelo menos em [Lon, 95].

Como uma medida, a segurança pode ser vista como uma extensão de fiabilidade. Agrupemos o estado de serviço correcto como o estado de um serviço incorrecto subsequente a uma avaria benigna num estado chamado seguro (no sentido de não ter provocado danos catastróficos, não forçosamente de perigo); a **segurança** é então uma medida da continuidade da segurança, ou seja, do tempo para uma avaria catastrófica. A segurança pode então ser considerada como fiabilidade no que respeita a avarias catastróficas. Uma extensão directa de disponibilidade, i.e. uma medida de segurança que exprimisse o facto de estar num estado seguro, no que respeita à alternância entre estado seguro e serviço incorrecto depois de uma avaria catastrófica, não forneceria uma medida significativa. Quando ocorre uma avaria catastrófica, as suas consequências são geralmente tão importantes que a restauração do serviço não é de importância primordial, devido - pelo menos – às duas seguintes razões:

- torna-se secundário, em detrimento da reparação das consequências da catástrofe (no sentido lato do termo, incluindo aspectos legais);
- o longo período que decorre antes de ser permitido operar novamente o sistema (comissões de inquérito, etc.) levaria a valores numéricos não significativos.

Pode contudo definir-se uma medida híbrida do tipo fiabilidade-disponibilidade: uma medida da prestação de um serviço correcto, no que diz respeito à alternância entre serviço correcto e incorrecto depois de uma avaria benigna. Esta medida é de interesse no sentido em que fornece realmente uma quantificação da disponibilidade do sistema *antes* da ocorrência de uma avaria catastrófica, permitindo assim a quantificação do chamado "compromisso fiabilidade (ou disponibilidade) – segurança".

No caso de sistemas de alto desempenho[134], podem distinguir-se diversos serviços, bem como diversos modos de prestação dos serviços, desde a plena capacidade até à interrupção total, o que pode ser visto como cada vez menor número de prestações de serviços correctos. Para estes sistemas, as medidas de confiança no funcionamento relacionadas com desempenho são usualmente designadas por **capacidade de desempenho**[135] ([Mey, 78], [Smi, 88]).

As duas principais abordagens à previsão de falhas probabilística, destinadas a obter estimativas quantitativas das medidas da confiança no funcionamento são a modelização e o teste (avaliação). Estas abordagens são directamente complementares, dado que a modelização necessita de dados dos processos básicos modelizados (processo de avarias, processo de manutenção, processo de activação do sistema, etc.), que podem ser obtidos através do teste do sistema.

Quando se leva a cabo uma avaliação através de uma *modelização*, as abordagens diferem significativamente, dependendo se se considera que o sistema detém uma fiabilidade estável ou uma fiabilidade crescente, que podem ser definidas como s segue ([Lap, 90a]):

- **fiabilidade estável**: a capacidade do sistema para prestar um serviço correcto é *preservada* (identidade estocástica dos sucessivos tempos para a avaria);
- **fiabilidade crescente**: a capacidade do sistema para prestar um serviço correcto é *melhorada* (aumento estocástico dos sucessivos tempos para a avaria)[136].

Seguem-se as interpretações práticas de fiabilidade estável e de fiabilidade crescente:

---

[134] Tradução de "*multi-performance*".

[135] Tradução de "*performability*". Não existe um vocábulo correspondente em Português. Segundo [Lon, 95], o termo em Inglês também não existe.

[136] (NA23) A *diminuição de fiabilidade* (a capacidade do sistema para prestar um serviço correcto degrada-se, existindo portanto uma diminuição estocástica dos sucessivos tempos para a avaria) é possível, tanto em termos teóricos como práticos, por exemplo pela introdução de novas falhas durante acções correctivas, cuja probabilidade de activação é maior que a das falhas eliminadas. Nesse caso, espera-se que o decréscimo seja limitado no tempo e que a fiabilidade cresça globalmente, num período de observação alargado.

- fiabilidade estável: numa dada restauração, o sistema fica idêntico ao que era antes da restauração; isto corresponde às seguintes situações:
  - no caso de uma avaria no hardware, o componente avariado é substituído por outro, idêntico e não avariado;
  - no caso de uma avaria no software, o sistema é reinicializado com um padrão de entrada diferente do que levou à avaria;
- fiabilidade crescente: a falha cuja activação levou à avaria é diagnosticada como uma falha de concepção (no software ou no hardware), sendo removida.

A avaliação da confiança no funcionamento de sistemas com fiabilidade estável é normalmente composta por duas fases:

- *construção* do modelo do sistema a partir dos processos estocásticos elementares que modelizam o comportamento dos componentes do sistema e as suas interacções;
- *processamento* do modelo, de forma a obter as expressões e os valores das medidas de confiança no funcionamento do sistema.

A avaliação pode ser efectuada relativamente a a) falhas físicas ([Tri, 84]), b) falhas de concepção ([Lit, 79], [Arl, 88]) ou c) uma combinação das duas ([Lap, 84], [Pig, 88]). A confiança no funcionamento de um sistema depende fortemente do seu meio envolvente, quer seja no sentido lato do termo ([Hec, 87]) ou, mais especificamente, na sua carga ([Cas, 81], [Iye, 82]).

Muitos modelos de fiabilidade crescente têm sido propostos, dedicados a hardware ([Dua, 64]), software, ou ambos ([Lap, 90b]). A maior parte deles dedicam-se a software, destinando-se a avaliar ou a fiabilidade ([Yam, 85], [Mil, 86]) ou o número de falhas (que restam) ([Goe, 79], [Toh, 89]); como estes modelos se destinam a prever a fiabilidade futura, a partir das informações sobre avarias acumuladas do passado, tem sido dedicada uma atenção especial ao problema da predição ([Lit, 88]).

Embora não sejam – primariamente – destinados para verificar um sistema, os teste de avaliação ([Mil, 87], [Cho, 87]) podem ser caracterizados utilizando os pontos de vista definidos em '5.3. Supressão de Falhas': conformidade, funcional, baseado em falhas e estatístico[137]. Uma preocupação importante é que o perfil de entrada seja representativo do perfil operacional, daí o seu nome: **teste operacional**.

Quando avaliamos sistemas tolerantes a falhas, a consideração dos mecanismos de processamento de erros e de tratamento de falhas tem uma importância primordial ([Bou, 69], [Arn, 73]); a sua avaliação pode ser efectuada através de modelização ([Dug, 89]) ou por intermédio de testes, sendo denominada de injecção de falhas ([Arl, 89], [Gun, 89]).

[137] O autor deveria referir também o teste determinístico.

6. ATRIBUTOS DA CONFIANÇA NO FUNCIONAMENTO

Os atributos da confiança no funcionamento foram definidos em '1. Definições de Base', de acordo com diferentes propriedades, que podem ser mais ou menos enfatizadas, dependendo da aplicação pretendida para o sistema computacional considerado:

- a disponibilidade é sempre requerida, embora num grau variável, dependendo da aplicação;
- a fiabilidade, a segurança e a inviolabilidade podem ou não ser exigidas, de acordo com a aplicação[138].

Uma propriedade adicional, que pode ser vista como um requisito prévio para se preencherem as outras propriedades, é a **integridade**[139], i.e. a condição de permanecer intacto, no sentido lato do termo: a) para dados ou programas e b) no que respeita a falhas acidentais ou intencionais.

As variações da ênfase a dar aos atributos da confiança no funcionamento têm uma influência directa no doseamento equilibrado das técnicas a utilizar, descritas na secção anterior, de forma a que o funcionamento do sistema resultante seja de confiança. Isto é, globalmente, o problema mais difícil de resolver, dado que alguns atributos são antagónicos (por exemplo, disponibilidade e segurança, disponibilidade e inviolabilidade)[140], obrigando a fazer compromissos. Considerando os três maiores parâmetros a considerar na concepção de um sistema, i.e. custo, desempenho e confiança no funcionamento, o problema é ainda mais exacerbado pelo facto de o parâmetro da confiança no funcionamento ser menos compreendido que o custo e o desempenho ([Sie, 82]).

A capacidade de manutenção foi definida em '5.4. Previsão de Falhas' como uma medida da confiança no funcionamento. A **capacidade de manutenção** pode também ser considerada como um atributo da confiança no funcionamento, relativamente à facilidade que as acções de manutenção podem ser efectuadas.

A definição de inviolabilidade dada em '1. Definições de Base', isto é, a prevenção do acesso não autorizado e/ou manipulação de informação, deriva de [ITS, 90], que define inviolabilidade como a combinação de confidencialidade, a prevenção da modificação ou apagamento de informação, e disponibilidade como a prevenção da manipulação não autorizada de informação. É digno de nota que:

a) um acesso ou manipulação de informação não autorizados pode resultar tanto de uma falha acidental como de uma falhas intencional e que – como notado em '5.2. Tolerância a Falhas' – alguns mecanismos de protecção contra acesso não autorizado são comuns aos dois tipos de falhas;

b) relativamente às falhas intencionais, a noção de autorização deve ser entendida de forma alargada: um indivíduo autorizado que abusa da sua autoridade, viola de facto a autorização que lhe foi concedida, ao executar acções ilegítimas, tornando-se portanto um intruso.

Uma característica importante dos sistemas tolerantes a falhas é que, devido à existência de procedimentos de detecção de erros, são capazes de fornecer informação aos utilizadores, independentemente se o serviço prestado é ou

---

[138] Estas duas afirmações são muito discutíveis...

[139] [Lap, 95a], [Lap, 95b] e [Lap, 98] definem que integridade é um requisito para se obter disponibilidade, fiabilidade e segurança, mas pode não o ser para a confidencialidade. [Kir, 87] define integridade como "a prevenção da saída de dados errados". Um sistema avaria-pára é um sistema íntegro, neste ponto de vista.

[140] [Lev, 94] refere que o *desempenho* e a *fiabilidade* dos sistemas tolerantes a falhas são muitas vezes objectivos antagónicos. Neste contexto define o conceito de *eficiência* ("*effectiveness*") de um sistema como uma medida de até que ponto é que uma aplicação pode tolerar a existência de falhas e, ao mesmo tempo, cumprir os seus objectivos de desempenho.

não correcto. Esta propriedade é denominada de **credibilidade**[141], em [Fur, 90].

Avaliar se o funcionamento de um sistema é realmente de confiança – confiança justificada no serviço prestado – ou não, vai para além das técnicas de validação tal como foram abordadas na secção anterior, devido pelo menos às razões e limitações seguintes:

- verificar com exactidão a cobertura das hipóteses de concepção ou validação relativamente à realidade (por exemplo, o critério usado para determinar as entradas de teste deve tomar em consideração as falhas reais; as hipóteses de falhas na concepção de mecanismos de tolerância a falhas) implicaria o conhecimento e o domínio das tecnologias utilizadas, da utilização pretendida para o sistema, etc., que ultrapassa largamente o que é geralmente exequível;
- avaliar um sistema de acordo com alguns dos atributos da confiança no funcionamento, relativamente a algumas classes de falhas, é actualmente considerado como não exequível ou conduzindo a resultados não significativos: não existem bases probabilísticas teóricas ou não são – ainda – globalmente aceites; exemplos são a segurança no que respeita a falhas acidentais de concepção e a inviolabilidade no que diz respeito a falhas intencionais[142];
- as especificações relativamente às quais a validação é efectuada não são, geralmente, isentas de falhas – como em qualquer sistema.

Dentre as numerosas consequências destas considerações, devemos mencionar:

- a ênfase colocada no projecto e implementação de um sistema, quando da sua avaliação: os métodos e técnicas utilizados e como são aplicados; em alguns casos, é atribuída uma *nota* ao sistema, de acordo com a) a natureza dos métodos e técnicas utilizadas e b) uma avaliação da sua aplicação[143];
- a presença, na especificação de alguns sistemas tolerantes a falhas, do número de falhas que o sistema poderá tolerar[144], adicionalmente aos requisitos probabilísticos em termos de medidas da confiança no funcionamento; tal especificação não seria necessária se as limitações acima mencionadas pudessem ser ultrapassadas.

[141] Tradução de "*trustability*". Curiosamente, esta palavra não existe em Inglês, pelo menos em [Lon, 95].

[142] Começa a haver algum trabalho neste sentido, nomeadamente em [Joh, 96], que tem uma secção dedicada à avaliação das técnicas de confiança no funcionamento ('*1.3. Dependability Evaluation Techniques*'). [Wei, 97] merece também uma leitura, pois apresenta uma perspectiva do estado actual da investigação na área dos sistemas tolerantes a falhas, referindo temas como a 'injecção de falhas', 'medição e interpretação', 'modelização de fiabilidade' e 'prova de correcção', que poderão levar a uma melhor avaliação da confiança no funcionamento dos sistemas computacionais.

[143] (NA24) Por exemplo:
- os sistemas são classificados segundo o ponto de vista da inviolabilidade ([DoD, 85]), desde A1 ("concepção verificada") até D ("protecção mínima");
- o software utilizado nos aviões de transporte civis são classificados ([RTC, 85]) como Nível 1, 2 ou 3, de acordo com a criticalidade da função a ser cumprida pelo software: crítica, essencial, não-crítica.

[144] (NA25) Tais especificações são clássicas nas aplicações aeronáuticas, na forma de de um agrupamento de requisitos do tipo "avaria-operacional" ou "avaria-seguro".

CONCLUSÃO

É cada vez maior o número de pessoas individuais ou organizações que procuram ou concebem sistemas computacionais sofisticados, em cujos serviços necessitam de depositar uma grande confiança – quer seja para colocar em funcionamento *Caixas Multibanco*, para calcular a órbita de um satélite, controlar um avião ou uma central nuclear, ou manter a confidencialidade de uma base de dados sensível. Dependendo das circunstâncias, o foco será dado a diferentes propriedades desses serviços – por exemplo o tempo de resposta médio conseguido, a probabilidade de produzir os resultados esperados, a capacidade de evitar o aparecimento de avarias que poderão ser catastróficas para o meio envolvente do sistema ou o grau com que podem ser impedidas intrusões deliberadas. A noção de confiança no funcionamento permite, de uma forma muito conveniente, abarcar todas estas preocupações dentro de um quadro conceptual único. A confiança no funcionamento inclui portanto como casos particulares propriedades como fiabilidade, disponibilidade, segurança e inviolabilidade. Fornece também um meio de abordar o problema de que o que um utilizador normalmente necessita de um sistema é de um *equilíbrio apropriado* dessas propriedades.

GLOSSÁRIO

Aviso: este glossário é disponibilizado como um auxílio à leitura deste documento. Não deve portanto ser considerado isoladamente.

Falha acidental	Falha que aparece ou é criada fortuitamente.
Falha activa	Falha que provoca um erro.
Avaria arbitrária	Ver Avaria.
Sistema atómico	Sistema cuja estrutura interna não pode ser discernida, ou não se mostra relevante, podendo ser ignorada.
Atributos da confiança no funcionamento	Atributos que permitem avaliar a qualidade do sistema que resulta dos impedimentos à confiança no funcionamento e dos meios para os ultrapassar. Fiabilidade, disponibilidade, capacidade de manutenção, segurança, inviolabilidade e credibilidade.
Disponibilidade	Confiança no funcionamento, relativamente à capacidade para estar pronto a utilizar. Medida da prestação do serviço correcto, relativamente à alternância entre serviço correcto e incorrecto.
Evitação de falhas	Métodos e técnicas destinados a conceber um sistema isento de falhas. Prevenção de falhas e supressão de falhas.

Recuperação para trás	Forma de recuperação de erros que consiste em levar o sistema do estado erróneo a um estado anterior, isento de erros.
Comportamento de um sistema	O que um sistema faz.
Avaria benigna	Avaria cujos prejuízos são da mesma ordem de grandeza dos benefícios proporcionados pela correcta prestação do serviço.
Avaria catastrófica	Avaria cujas consequências são incomensuravelmente superiores aos benefícios proporcionados pela correcta prestação do serviço.
Erros coincidentes	Erros criados pelo mesmo estímulo.
Avarias de modo-comum	Avarias resultantes de falhas relacionadas.
Compensação de erros	Forma de processamento de erros onde o sistema dispõe de redundância suficiente de forma a prestar um serviço correcto a partir de um estado erróneo.
Componente de um sistema	Outro sistema.
Teste de conformidade	Teste cujo o objectivo é verificar se o sistema está de acordo com a especificação.
Avaria consistente	Avaria percebida da mesma forma por todos os utilizadores.
Serviço correcto	Serviço prestado em conformidade com a especificação do sistema.

Manutenção correctiva	Preservação ou melhoramento da capacidade de um sistema para prestar um serviço de acordo com a especificação. Supressão de falhas durante a vida operacional de um sistema.
Cobertura	Medida da representatividade das situações às quais o sistema é submetido durante a sua validação, comparadas com as situações reais com que ele será confrontado na sua vida operacional.
Avaria por omissão persistente	Avaria por omissão persistente.
Criticalidade (de um sistema)	A mais alta gravidade dos modos (possíveis) de avaria de um sistema.
Manutenção curativa	Manutenção correctiva destinada a remover falhas que foram produzidas por um ou mais erros e que foram sinalizadas.
Confiança no funcionamento	Fidedignidade de um sistema computacional no sentido em que, de uma forma justificada, se possa depositar confiança no serviço que ele presta.
Concepção diversificada	Uma abordagem à concepção de sistemas, envolvendo o fornecimento de serviços idênticos a partir de projectos e/ou implementações separadas.
Falha de concepção	Falha humana interna.
Concepção tendo em vista a verificação	Métodos e técnicas utilizados na concepção de uma sistema que facilitam a sua verificação.

Detecção de erros	O acto de identificar que um estado de um sistema é erróneo.
Erro detectado	Erro reconhecido como tal por intermédio de um mecanismo ou algoritmo de detecção.
Teste determinístico	Uma forma de teste onde os padrões de teste são predeterminados por uma escolha selectiva.
Diagnóstico de falhas	A acção de determinar a causa de um erro, em termos de localização e de natureza.
Falha dormente	Falha interna não activada pelo processo computacional.
Verificação dinâmica	Verificação de um sistema envolvendo a sua activação.
Meio envolvente de um sistema	Sistemas que interagiram ou interferiram, que estão a interagir ou interferir, ou com probabilidade vir interagir ou interferir com o sistema considerado.
Erro	Parte do estado de um sistema que poderá levar a uma avaria.
Falha externa	Falha resultante de uma interferência ou interacção do meio envolvente do sistema.
Avaria-seguro (sistema ~)	Sistema onde todas as avarias são - ou mais genericamente, onde as avarias são numa quantidade aceitável – avarias benignas.
Avaria-silencia-se (sistema ~)	Sistema em que as avarias apenas podem ser – ou são numa certa quantidade – avarias por omissão persistente.
Avaria-pára (sistema ~)	Sistema em que as avarias apenas podem ser – ou mais genericamente, onde as avarias são numa quantidade aceitável – avarias por paragem.
Avaria	Não conformidade com a especificação por parte do serviço prestado. Transição de um serviço prestado correcto para um serviço prestado incorrecto.
Falha	Causa julgada ou hipotética de um erro. Causa de um erro que se pretende ver evitada ou tolerada. Consequência, para um sistema, da avaria de outro sistema que interagiu ou está a interagir com o sistema considerado.
Teste baseado em falhas	Teste destinado a revelar classes específicas de falhas.
Teste de procura de falhas	Teste destinado a revelar falhas.
Previsão de falhas	Métodos e técnicas destinados a estimar o número actual, a incidência futura e as consequências das falhas.
Recuperação para a frente	Forma de recuperação de erros que consiste em procurar um novo estado para o sistema.
Função de um sistema	Para o que se destina o sistema.
Teste funcional	Uma forma de teste onde as entradas de teste são seleccionadas de acordo com um critério relacionado com a função do sistema.

Falha acentuada	Falha que necessita de desactivação.
Falha humana	Falha resultante de imperfeição humana.
Impedimentos à confiança no funcionamento	Circunstâncias indesejadas, mas ao mesmo tempo inesperadas, que resultam da não confiança no funcionamento. Falhas, erros e avarias.
Avaria inconsistente	Avaria tal que os utilizadores do sistema poderão ter diferentes percepções dela.
Serviço incorrecto	Serviço prestado que não está em conformidade com a especificação do sistema.
Falhas independentes	Falhas que são atribuídas a causas diferentes.
Integridade	Condição de permanecer intacto
Falha intencional	Falha criada deliberadamente.
Falha intermitente	Falha interna temporária. Falhas cujas condições de activação não podem ser reproduzidas ou raramente ocorrem.
Falha interna	Falha dentro de um sistema.
Intrusão	Falha externa de operação e intencional.
Erro latente	Erro não reconhecido como tal.

Capacidade de manutenção	Facilidade com que as acções de manutenção podem ser levadas a cabo. Medida da continuidade da prestação de um serviço incorrecto. Medida do tempo de restauração desde a última avaria ocorrida.
Lógica maliciosa	Falha intencional de concepção.
Dissimulação de falhas	O resultado da aplicação sistemática da compensação de erros, mesmo na ausência de erros.
Meios para a obtenção de confiança no funcionamento	Métodos e técnicas que permitem a) dotar o sistema com a capacidade de prestar um serviço em que se deposita confiança e b) alcançar confiança nesta capacidade.
Avaria por omissão	Avaria tal que não é prestado nenhum serviço.
Falhas de operação	Falhas que aparecem durante a exploração do sistema.
Teste operacional	Teste executado para avaliar a confiança no funcionamento de um sistema, com um perfil de entrada representativo das suas condições de operação.
Desactivação de falhas	As acções tomadas de forma a que uma falha não possa ser activada.
Capacidade de desempenho	Medida de confiança no funcionamento relacionada com o desempenho.

Falha permanente	Falha cuja ocorrência não depende de condições pontuais, quer sejam internas ou externas.
Falha física	Falha resultante de fenómenos físicos adversos.
Manutenção preventiva	Manutenção correctiva destinada a remover as falhas antes de elas se activarem.
Prevenção de falhas	Métodos e técnicas destinados a evitar a ocorrência ou introdução de falhas.
Processamento de erros	As acções tomadas para eliminar os erros de um sistema.
Obtenção de confiança no funcionamento	Métodos e técnicas destinados a dotar um sistema da capacidade de prestar um serviço de acordo com a especificação. Prevenção de falhas e tolerância a falhas.
Teste aleatório	Ver 'teste estatístico'.
Função de tempo-real	Função que se requer cumprida dentro de intervalos de tempo finitos, ditados pelo meio envolvente do sistema.
Sistema de tempo-real	Sistema que cumpre pelo menos uma função de tempo-real ou presta pelo menos um serviço de tempo-real.
Recuperação de erros	Forma de processamento de erros onde se substitui um estado erróneo por um estado isento de erros.
Ponto de recuperação	Pontos no tempo, durante a execução de um processo, para os quais o seu estado pode posteriormente ser restaurado.
Verificação regressiva	Verificação efectuada após uma correcção, de forma a verificar se a correcção não teve consequências indesejáveis.
Falhas relacionadas	Falhas atribuídas a uma única causa.
Fiabilidade	Confiança no funcionamento no que respeita à continuidade do serviço. Medida da continuidade da prestação de um serviço correcto. Medida do tempo para a avaria.
Fiabilidade crescente	A capacidade de um sistema prestar um serviço correcto é melhorada (aumento estocástico dos sucessivos tempos para a avaria).
Supressão de falhas	Métodos e técnicas para minorar a ocorrência (quantidade e gravidade) de falhas.
Restauração do serviço	Transição entre a prestação de um serviço incorrecto e a prestação de um serviço correcto.
Segurança	Confiança no funcionamento relativamente à não ocorrência de avarias catastróficas. Medida da continuidade na prestação de um serviço correcto ou incorrecto depois de uma avaria benigna. Medida do tempo para uma avaria catastrófica.
Inviolabilidade	Confiança no funcionamento relativamente à prevenção de acesso e/ou manipulação de informação não autorizados.

Componente auto-testável	Componente que associa a sua funcionalidade de processamento com mecanismos de detecção de erros.
Avarias sequenciais	Avarias que não ocorrem dentro de uma janela temporal predefinida.
Serviço	Comportamento de um sistema da perspectiva do utilizador do sistema.
Gravidade de uma avaria	Classificação das consequências das avarias para o meio envolvente do sistema.
Avarias simultâneas	Avarias que ocorrem dentro de uma janela temporal predefinida.
Falha ténue	Falha para a qual não se leva a cabo a desactivação.
Falha sólida	Ver 'falha acentuada'.
Especificação (de um sistema)	Descrição acordada dos requisitos do sistema.
Estado (de um sistema)	Uma situação em que se está, no que respeita a um conjunto de circunstâncias.
Fiabilidade estável	A capacidade do sistema para prestar um serviço correcto é preservada (identidade estocástica dos sucessivos tempos para a avaria).
Verificação estática	Verificação de um sistema sem o activar.
Teste estatístico	Forma de teste onde os padrões de teste são seleccionados segundo uma distribuição de probabilidade do domínio de entrada.

Avaria por paragem	Avaria em que a actividade do sistema, se existir, deixa de ser perceptível pelos utilizadores, passando a ser prestado um serviço de valor constante.
Estrutura (de um sistema)	O que permite a um sistema fazer o que faz.
Teste estrutural	Forma de teste em que as entradas de teste são seleccionadas de acordo com um critério relativo à estrutura do sistema.
Execução simbólica	Verificação dinâmica em que as entradas fornecidas ao sistema são simbólicas.
Sistema	Entidade que interagiu, que está a interagir ou capaz de vir a interagir com outras entidades. Conjunto de componentes agrupados de forma a interagir.
Falha temporária	Falha que permanece presente durante um período limitado de tempo.
Teste	Verificação dinâmica efectuada com entradas estimadas.
Avaria temporal	Avaria tal que o tempo da prestação do serviço não está em conformidade com a especificação.
Tolerância a falhas	Métodos e técnicas para que um sistema forneça um serviço de acordo com a sua especificação, mesmo na presença de falhas.
Falha transitória	Falha externa temporária e física.

Tratamento de falhas	Acções destinadas a impedir que as falhas sejam reactivadas.
Credibilidade	Capacidade de um sistema fornecer informação aos seus utilizadores acerca da correcção do serviço prestado.
Utilizador (de um sistema)	Outro sistema (físico ou humano) que interage com o sistema considerado.
Validação (da confiança no funcionamento)	Métodos e técnicas destinados a alcançar confiança na capacidade do sistema para prestar um serviço de acordo com a especificação.
Avaria de valor	Avaria em que o valor do serviço prestado não está de acordo com a especificação.
Verificação	Processo de verificar se um sistema satisfaz certas propriedades (condições de verificação) que podem ser a) genéricas, independentes da especificação ou b) específicas, deduzidas da especificação.

ÍNDICE PORTUGUÊS-INGLÊS

Atributos da confiança no funcionamento	Attributes of dependability
Compensação (de um erro)	Compensation (error ~)
Comportamento (de um sistema)	Behaviour (system ~)
Componente (de um sistema)	Component (system ~)
Componente auto-testável	Self-checking component
Concepção tendo em vista a verificação	Design for Verifiability
Cobertura	Coverage
Credibilidade (de um serviço)	Trustability[145] (system ~)
Criticalidade (de um sistema)	Criticality (system ~)
Fiabilidade crescente	Reliability Growth
Avaria	Failure
Avaria arbitrária	Arbitrary failure
Avaria benigna	Benign failure
Avaria catastrófica	Catastrophic failure
Avaria consistente	Consistent failure
Avarias de modo-comum	Common-mode failures
Avaria de valor	Value failure

[145] Ver 141.

Avaria inconsistente	Inconsistent failure
Avaria por paragem	Stopping failure
Avaria por omissão persistente	Crash failure
Avaria por omissão	Omission failure
Avarias sequenciais	Sequential failures
Avarias simultâneas	Simultaneous failures
Avaria temporal	Timing failure
Detecção (de erros)	Detection (error ~)
Diagnóstico (de falhas)	Diagnosis (fault ~)
Disponibilidade	Availability
Concepção diversificada	Design diversity
Supressão de falhas	Fault removal
Impedimentos (à confiança no funcionamento)	Impairments (to dependability)
Meio envolvente (de um sistema)	Environment (system ~)
Erro	Error
Erros coincidentes	Coincident errors
Erro detectado	Detected error
Erro latente	Latent error
Estado (de um sistema)	State (system ~)
Evitação de falhas	Fault avoidance
Execução simbólica	Symbolic execution

Falha	Fault
Falha acidental	Accidental fault
Falha activa	Active fault
Falhas relacionadas	Related faults
Falha de concepção	Design fault
Falha dormente	Dormant fault
Falha ténue	Soft fault
Falha acentuada, ou sólida	Hard, or solid, fault
Falha externa	External fault
Falha humana	Human-made fault
Falhas independentes	Independent faults
Falha intencional	Intentional fault
Falha intermitente	Intermittent fault
Falha interna	Internal fault
Falha de operação	Operational fault
Falha permanente	Permanent fault
Falha física	Physical fault
Falha temporária	Temporary fault
Falha transitória	Transient fault
Fiabilidade	Reliability
Fiabilidade estável	Stable reliability

Função (de um sistema)	Function (system ~)
Função de tempo-real	Real-time function
Integridade	Integrity
Intrusão	Intrusion
Lógica maliciosa	Malicious logic
Capacidade de manutenção	Maintainability[146]
Manutenção correctiva	Corrective maintenance
Manutenção curativa	Curative maintenance
Manutenção preventiva	Preventive maintenance
Dissimulação de falhas	Fault masking
Meios (para a obtenção de confiança no funcionamento)	Means (for dependability)
Obtenção (de confiança no funcionamento)	Procurement (of dependability)
Desactivação (de falhas)	Passivation[147] (fault)
Capacidade de desempenho	Performability[148]
Ponto de recuperação	Recovery point
Recuperação para a frente	Forward recovery
Prevenção de falhas	Fault prevention
Previsão de falhas	Fault forecasting
Recuperação de erros	Error recovery
Recuperação para trás	Backward recovery
Restauração (de um serviço)	Restoration (service ~)
Segurança	Safety
Inviolabilidade	Security
Serviço	Service
Serviço correcto	Correct service
Serviço incorrecto	Incorrect service
Serviço de tempo-real	Real-time service
Gravidade (de uma avaria)	Severity (failure ~)
Especificação (de um sistema)	Specification (system ~)
Estrutura (de um sistema)	Structure (system ~)
Confiança no funcionamento	Dependability
Sistema	System
Sistema avaria-pára	Fail-stop system
Sistema avaria-silencia-se	Fail-silent system
Sistema atómico	Atomic system
Sistema avaria-seguro	Fail-safe system
Sistema de tempo-real	Real-time system
Teste	Testing

[146] Ver 133.Curiosamente, ao longo do texto, o termo utilizado é "*maintainability*".

[147] Ver 103.

[148] Ver 135.

Teste aleatório ou estatístico	Random, or statistical, testing
Teste baseado em falhas	Fault-based testing
Teste de conformidade	Conformance testing
Teste determinístico	Deterministic testing
Teste funcional	Functional testing
Teste de procura de falhas	Fault-finding testing
Teste operacional	Operational testing
Teste estrutural	Structural testing
Tolerância a falhas	Fault tolerance
Processamento de erros	Error processing
Tratamento de falhas	Fault treatment
Utilizador (de um sistema)	User (system ~)
Validação (da confiança no funcionamento)	Validation (dependability ~)
Verificação	Verification
Verificação dinâmica	Dynamic verification
Verificação regressiva	Regression verification
Verificação estática	Static verification

REFERÊNCIAS DA VERSÃO PORTUGUESA

[Bow, 93] N. Bowen, D. Pradhan, *Processor and Memory-Based Checkpoint and Rollback Recovery*, IEEE Computer, pp. 22-31, February 1993.

[Cri, 91] Flaviu Cristian, *Basic Concepts and Issues in Fault-Tolerant Distributed Systems*, in Int. Workshop on Operating Systems of the 90's and Beyond, Dagstuhl Castle, Germany, July 8-12, 1991.

[Joh, 96] B. W. Johnson, *An Introduction to the Design and Analysis of Fault-Tolerant Systems*, in D. K. Pradhan, *Fault-Tolerant Computer System Design*, Prentice-Hall, 1996.

[Kir, 87] H. D. Kirrman, *Fault Tolerance in Process Control: an overview and examples of european products*, IEEE Micro, October 1987.

[Kop, 91] H. Kopetz, *Event-Triggered versus Time-Triggered Real-Time Systems*, in Int. Workshop on Operating Systems of the 90's and Beyond, Dagstuhl Castle, Germany, July 8-12, 1991.

[Kop, 93] H. Kopetz, P. Veríssimo, *Real Time and Dependability Concepts*, in Distributed Systems, S. Mullender (ed.), Addison-Wesley, ISBN 0-201-62427-3, 1993.

[Lap, 92] Jean-Claude Laprie (ed.), *Dependability: Basic Concepts and Terminology, in English, French, German, Italian and Japanese*, Dependable Computing and Fault-Tolerance, Vol. 5, Springer-Verlag, ISBN 3-211-82296-8, 1992.

[Lap, 93] Jean-Claude Laprie (ed.), *Dependability – from Concepts to Limits*, in Proc. SAFECOMP'93, Poznan, Poland, pp. 157-168, Springer-Verlag, 1993.

[Lap, 95a] Jean-Claude Laprie (ed.), *Guide de la Sûreté de Fonctionnement*, Laboratoire d'Ingénierie de la Sûreté de Fonctionnement, Cépaduès-Éditions, ISBN 2-85428-382-1, 1995.

[Lap, 95b] Jean-Claude Laprie (ed.), *Dependability – Its Attributes, Impairements and Means*, in Predictably Dependable Computing Systems, ESPRIT Basic Research Series, Springer, ISBN 3-540-59334-9, 1995.

[Lap, 98] Jean-Claude Laprie (ed.), *Dependability Handbook*, Preliminary Version, Laboratoire d'Ingénierie de la Sûreté de Fonctionnement, LAAS Report nº 98-346, 1998.

[Lev, 94] S. T. Levi, A. K. Agrawala, *Fault-Tolerant System Design*, McGraw-Hill International Editions, ISBN 0-07-037515-1, 1994.

[Lev, 95] Nancy G. Leveson, *Safeware: system safety and computers*, Addison-Wesley, ISBN 0-201-11972-2, 1995.

[Lon, 95] Longman, *Dictionary of Contemporary English*, 3th Edition, Longman Group Ltd., ISBN 0-582-23750-5, 1995.

[Mag, 95] António P. Magalhães, *Estabilização dos Controladores de Tempo-Real Através da Complacência Temporal dos Objectos Controlados*, tese para a obtenção do grau de Doutor em Engenharia Electrotécnica e de Computadores, submetida à Faculdade de Engenharia da Universidade do Porto, Março de 1995.

[Nel, 90] V. Nelson, *Fault-Tolerant Computing: Fundamental Concepts*, IEEE Computer, pp.19-25, July 1990.

[OLO, 98] *European Summer School on Reliability and Safety of Human-Machine Systems*, Trainning Material, Crete, Grece, 6-13 September 1998.

[Pin, 48] Eduardo Pinheiro, Dicionário da Língua Portuguesa, Livraria Figueirinhas, Porto, 1948.

[Por, 85] Porto Editora, *Dicionário de Inglês-Português*, Porto Editora Lda., 1985.

[Por, 98] Porto Editora, *Dicionário da Língua Portuguesa*, 8ª edição, Porto Editora Lda., 1998.

[Ver, 89] Paulo Veríssimo, Rogério de Lemos, *Confiança no Funcionamento: Proposta para uma Terminologia em Português*, Relatório Técnico, Outubro de 1989.

[Ver, 93] Paulo Veríssimo, *Real-Time Communication*, in Distributed Systems, S. Mullender (ed.), Addison-Wesley, ISBN 0-201-62427-3, 1993.

[Wei, 97] Charles B. Weinstock, David P. Gluch, *A Perspective on the State of Research in Fault-Tolerant Systems*, Special Report CMU/SEI-97-SR-008, Software Engineering Institute, Carnegie Mellon University, Pittsburgh, Pennsylvania, June 1997.

REFERÊNCIAS DA VERSÃO ORIGINAL

[Ada, 84] E. N. Adams, "*Optimizing preventive service of software products*", IBM Journal of Research and Development, vol. 28, nº1, pp. 2-14, January 1984.

[Adr, 82] W. R. Adrion, M. A. Branstad, J. C. Cherniavsky, "*Validation, verification and testing of computer software*", Computing Surveys, vol. 14, nº2, pp. 159-192, June 1982.

[And, 81] T. Anderson, P. A. Lee, "*Fault Tolerance – Principles and Practice*", Prentice Hall, 1981.

[Arl, 88] J. Arlat, K. Kanoun, J. C. Laprie, "*Dependability evaluation of software fault-tolerance*", in Proc. 18th IEEE Int. Symp. On Fault Tolerant Computing (FTCS-18), Tokyo, pp.142-147, June 1988; also in IEEE Trans. On Computers, vol. 39, nº4, pp. 504-513, April 1990.

[Arl, 89] J. Arlat, Y. Crouzet, J. C. Laprie, "*Fault injection for dependability validation of fault-tolerant computing systems*", in Proc. 19th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-19), Chicago, pp.348-355, June 1989.

[Arn, 73] T. F. Arnold, "*The concept of coverage and its effect on the reliability model of repairable systems*", IEEE Transactions on Computers, vol. C-22, pp. 251-254, June 1973.

[Avi, 67] A. Avizienis, "*Design of Fault Tolerant Computers*", in Proc. Fall Joint Computer Conf., pp. 733-743, 1967.

[Avi, 78] A. Avizienis, "*Fault Tolerance, the survival attribute of digital systems*", Proceedings of the IEEE, vol. 66, nº10, pp. 1109-1125, October 1978.

[Avi, 84] A. Avizienis, J. P. Kelly, "*Fault Tolerance by design diversity: concepts and experiments*", Computer, vol. 17, nº8, pp. 67-80, August 1984.

[Avi, 86] A. Avizienis, J. C. Laprie, "*Dependable Computing: from concepts to design diversity*", Proceedings of the IEEE, vol. 74, nº5, pp. 629-638, May 1986.

[Bis, 88] P. G. Bishop, "*The PODS diversity experiment*", in Software Diversity in Computerised Control Systems, U. Voges editor, Wien: Springer-Verlag, pp. 51-84, 1988.

[Boe, 79] B. W. Boehm, "*Guidelines for verifying and validating software requirements and design specifications*", in Proc. EURO IFIP'79, London, pp. 711-719, September 1979.

[Boe, 88] B. W. Boehm, "*A spiral model of software development and enhancement*", IEEE Computer, pp. 61-72, May 1988.

[Bou, 69] W. G. Bouricius, W. C. Carter, P. R. Schneider, "*Reliability Modeling Techniques for Self-Repairing Computer Systems*", in Proc. 24th ACM National Conf., pp. 295-309, 1969.

[Car, 68] W. C. Carter, P. R. Schneider, "*Design of dinamically checked computers*", in Proc. IFIP'68 Cong., Amsterdam, pp. 878-883, 1968.

[Car, 78] W. C. Carter, W. H. Joyner, D. Brand, H. A. Ellozy, J. L. Wolf, "*An improved system to verify assembled programs*", in Proc. 8th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-8), Toulouse, France, pp. 165-170, June 1978.

[Car, 79] W. C. Carter, "*Fault detection and recovery algorithms for fault-tolerant systems*", in Proc. EURO IFIP'79, London, pp. 725-734, September 1979.

[Car, 82] W. C. Carter, "*A Time for Reflection*", in Proc. 12th IEEE Int. Symp. On Fault Tolerant Computing (FTCS-12), Santa Monica, California, p.41, June 1982.

[Cas, 81] X. Castillo, D. P. Siewiorek, "*Workload, performance and reliability of digital computing systems*", in Proc. 11th IEEE Int.

Symp. On Fault Tolerant Computing (FTCS-11), Portland, Maine, California, pp. 84-89, June 1981.

[Che, 81] M. H. Cheheyl, M. Gasser, G. A. Huff, J. K. Miller, "*Verifying security*", Computing Surveys, vol. 13, nº3, pp. 279-339, September 1981.

[Cho, 87] C. K. Cho, "*Quality Programming*", Wiley, New York, 1987.

[Cra, 87] D. Craigen, "*Strengths and weaknesses of program verification systems*", in Proc. 1st European Software Engineering Conf., Strasbourg, France, pp. 421-429, September 1987.

[Cri, 80] F. Cristian, "*Exception handling and software fault tolerance*", in Proc. 10st IEEE Int. Symp. On Fault Tolerant Computing (FTCS-10), Kyoto, Japan, pp. 97-103, October 1980; also in IEEE Trans. on Computers, vol. C-31, nº6, pp. 531-540, June 1982.

[Cri, 85a] F. Cristian, H. Aghili, R. Strong, D. Dolev, "*Atomic broadcast: from simple message diffusion to Byzantine agreement*", in Proc. 15st IEEE Int. Symp. On Fault Tolerant Computing (FTCS-15), Ann Arbor, Michigan, pp. 200-206, June 1985.

[Cri, 85b] F. Cristian, "*Exceptions, failures and errors*", Technique et Science Informatique, vol. 4, nº3, pp. 385-390, 1985; in French, English version avail. As IBM Research Report nº RJ 4130, September 1983.

[Cri, 88] F. Cristian, "*Agreeing on who is present and who is absent in a synchronous distributed system*", in Proc. 18st IEEE Int. Symp. On Fault Tolerant Computing (FTCS-18), Tokyo, pp. 206-211, June 1988.

[Dav, 81] R. David, P. Thévenod-Fosse, "*Random testing of integrated circuits*", IEEE Trans. on Instrumentation and Measurement, vol. IM-30, nº1, pp. 20-25, March 1981.

[Dav, 86] R. David, "*Signature analysis for multiple output circuits*", IEEE Trans. on Computers, vol. C-35, nº9, pp. 830-837, September 1986.

[DeM, 78] R. A. DeMillo, R. J. Lipton, F. G. Sayward, "*Hints on test data selection: help for the practising programmer*", Computer, pp. 34-41, April 1978.

[Den, 82] D. E. Denning, "*Cryptography and Data Securuty*", Addison-Wesley, 1982.

[Dia, 82] M. Diaz, "*Modeling and Analysis of communication and cooperation protocols using Petri net based models*", Computer Networks, vol. 6, nº6, pp. 419-441, December 1982.

[DoD, 85] Department of Defence Standard, "*Department of Defence trusted computer system evaluation criteria*", DoD 5200.28-STD, December 1985.

[Dua, 64] J. T. Duane, "*Learning curve approach to reliability monitoring*", IEEE Trans. on Aerospace, vol. 2, pp. 563-566, 1964.

[Dug, 89] J. B. Dugan, K. S. Trivedi, "*Coverage modeling for dependability analysis of fault-tolerant systems*", IEEE Trans. on Computers, vol. 38, nº6, pp. 775-787, June 1989.

[Dur, 84] J. W. Duran, S. C. Ntafos, "*An evaluation of random testing*", IEEE Trans. on Software Engineering, vol. SE-10, nº4, pp. 438-444, July 1984.

[Elm, 72] W. R. Elmendorf, "*Fault-tolerant programming*", in Proc. 2nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-2), Newton, Massachusetts, pp. 79-83, June 1972.

[Ezh, 86] P. D. Ezhilchelvan, S. K. Shrivastava, "*A characterisation of faults in systems*", in Proc. 5th Symp. on Reliability in Distributed Software and Databases Systems, Los Angeles, pp. 215-222, January 1986.

[Fra, 86] J. M. Fray, Y. Deswarte, D. Powell, "*Intrusion tolerance using fine-grain fragmentation-scattering*", in Proc. 1986 IEEE Symp. on Security and Privacy, Oakland, pp. 194-201, April 1986.

[Fri, 82] S. G. Frison, J. H. Wensley, "*Interactiveconsistency and its impact on the design of TMR systems*", in Proc. 12nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-12), Santa Monica, California, pp. 228-233, June 1982.

[FTC, 82] Special session "*Fundamental concepts of fault tolerance*", in Proc. 12nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-12), Santa Monica, California, pp. 228-233, June 1982:

D. E. Morgan, "*Report of subcommittee on models, fundamental concepts and terminology*", pp. 3-5.

A. Avizienis, "*The four-universe information system model for the study of fault tolerance*", pp. 6-13.

H. Kopetz, "*The failure fault model*", pp. 14-17.

J. C. Laprie, A. Costes, "*Dpendability: a unifying concept for reliable computing*", pp. 18-21.

A. S. Robinson, "*A user oriented perspective of fault tolerant system models and terminologies*", pp. 22-28.

T. Anderson, P. A. Lee, "*Fault tolerance terminology proposals*", pp. 29-33.

P. A. Lee, D. E. Morgan, editors, "*Fundamental concepts of fault tolerant computing*", pp. 34-38.

[Fur, 90] D. A. Fura, A. K. Somani "*Trustability: a dependability measure for systems with failure reporting capability*", University of Washington, Seattle, Tech, Rep. EE-FTL-90-03, 1990.

[Gas, 88] "*Building a Secure Computer System*", Van Nostrand Reinhold, 1988.

[Goe, 79] A. L. Goel, K. Okumoto, "*Time-dependent error-detection rate model for software and other performance measures*", IEEE Trans. on Reliability, vol. R-28, nº3, pp. 465-484, August 1979.

[Gol, 82] J. Goldberg, "*A time for integration*", in Proc. 12nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-12), Santa Monica, California, pp. 42, June 1982.

[Goo, 75] J. B. Goodenough, S. L. Gerhart, "*Toward a theory of test data selection*", IEEE Trans. on Software Engineering, vol. SE-1, nº2, pp. 156-173, June 1975.

[Gra, 86] J. N. Gray, "*Why do computer stop and what can be done about it?*", in Proc. 5th Symp. on Reliability in Distributed Software and Databases Systems, Los Angeles, pp. 3-12, January 1986.

[Gun, 89] U. Gunneflo, J. Karlsson, J. Torin, "*Evaluation of error detection schemes using fault injection by heavy-ion radiation*", in Proc. 19nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-19), Chicago, Massachusetts, pp. 340--347, June 1989.

[Hec, 87] H. Hecht, E. Fiorentino, "*Reliability assessment of spacecraft electronics*", in Proc. 1987 Annual Reliability and Maintainability Symp.

[Hoa, 69] C. A. Hoare, "An axiomatic basis for computer programming", Communications of the ACM, vol. 12, nº10, pp. 576-583, October 1969.

[Hos, 60] J. E. Hosford, "*Measures of Dependability*", Operations Reasearch, vol. 8, nº1, pp. 204-206, 1960.

[How, 87] W. E. Howden, "*Functional Program Testing and Analysis*", McGraw-Hill, 1987.

[Hua, 82] K. K. Huang, J. A. Abraham, "*Low cost schemes for fault tolerance in matrix operations with processor arrays*", in Proc. 12nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-12), Santa Monica, California, pp. 330-337, June 1982.

[IEC, 85] IEC 191, "*Reliability, Maintainability and Quality of Service*", chapter 191 of the International Electrotechnical Vocabulary, Document 1-IEV-191-Central Office-1243 and 56-IEV-191-

Central Office-119, Geneva: International Electrotechnical Commission, 1985.

[IEE, 82] IEEE Std. 729, "*IEEE Standard Glossary of Software Engineering Terminology*", New York: IEEE, 1982.

[ITS, 90] "*Information Technology Security Evaluation Criteria*", Harmonised criteria of France, Germany, the Netherlands, the United Kingdom, May 1990.

[Iye, 82] R. K. Iyer, S. E. Butner, E. J. McCluskey, "*A statistical failure/load relationship: results of a multi-computer study*", IEEE Trans. on Computers, vol. C-31, pp.697-706, July 1982.

[Jes, 77] D. C. Jessep, "*Fault-tolerant computing, definition of terms*", Report IEEE Computer Society P610/DI, February 1977.

[Jos, 88] M. K. Joseph, A. Avizienis, "*A fault tolerance approach to computer viruses*", in Proc. 1988 Symp. on Security and Privacy, Oakland, pp. 52-58, April 1988.

[Kop, 87] H. Kopetz, W. Ochsenreiter, "*Clock synchronisation in distributed real-time systems*", IEEE Trans. on Computers, vol C-36, nº8, pp. 933-940, August 1987.

[Kui, 85] B. Kuipers, "*Common sense reasoning about causality: deriving behaviour from structure*", in Qualitative Reasoning about Physical Systems, D. G. Bobrow editor, MIT Press, pp. 169-203, 1985.

[Lal, 86] J. H. Lala, "*A Byzantine resilient fault tolerant computer for nuclear power plant applications*", in Proc. 16nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-16), Vienna, Austria, pp. 338-343, July 1986.

[Lam, 81] B. W. Lampton, "*Atomic transactions*", in Distributed Systems – Architecture and implementation, Lecture Notes in Computer Science 105, Berlin: Springer-Verlag, chap. 11, 1981.

[Lam, 82] B. W. Lampton, R. Shostak, M. Pease, "*The Byzantine generals problem*", ACM Trans. on Programming Languages and Systems, vol. 4, nº3, pp. 382-401, July 1982.

[Lam, 85] B. W. Lampton, P. M. Melliar-Smith, "*Synchronising clocks in the presence of faults*", Journal of the ACM, vol. 32, nº1, pp. 52-78, January 1985.

[Lap, 84] J. C. Laprie, "*Dependability evaluation of software systems in operation*", IEEE Trans. on Software Engineering, vol SE-10, nº6, pp. 701-714, November 1984.

[Lap, 85] J. C. Laprie, "*Dependable computing and fault tolerance: concepts and terminology*", in Proc. 15st IEEE Int. Symp. On Fault Tolerant Computing (FTCS-15), Ann Arbor, Michigan, pp. 2-11, June 1985.

[Lap, 89] J. C. Laprie, "*Dependability: A Unifying Concept for Reliable Computing and Fault Tolerance*", in Dependability of Resilient Computing Systems, Blackwell Scientific Publications, T. Anderson editor, pp. 1-28, 1989.

[Lap, 90a] J. C. Laprie, J. Arlat, C. Beounes, K. Kanoun, "*Definition and analysis of hardware- and software-fault-tolerant architectures*", IEEE Computer, vol. 23, nº7, pp. 39-51, July 1990.

[Lap, 90b] J. C. Laprie, C. Beounes, M. Kaaniche, K. Kanoun, "*The transformation approach to the modelling and evaluation of the reliability and availability growth of systems in operation* ", in Proc. 20st IEEE Int. Symp. On Fault Tolerant Computing (FTCS-20), Newcastle, United Kingdom, July 1990; extended version in IEEE Trans. on Software Engineering, "*The KAT (knowledge-action-transformation) approach to the modelling and evaluation of reliability and availability growth*", vol. 17, nº4, pp. 370-382, April 1991.

[Lev, 86] Y. Levendel, "*Fault simulation*", in Fault-Tolerant Computing, Theory and Techniques, D. K. Pradhan editor, Englewood Clifs: Prentice Hall, pp. 184-264, 1986.

[Lit, 79] B. Littlewood, "*Software reliability model for modular program structure*", IEEE Trans. on Reliability, vol. R-28, pp. 241-246, August 1979.

[Lit, 88] B. Littlewood, "*Forecasting software reliability*", in Software Reliability Modelling and Identification, S. Bittanti editor, Springer-Verlag, pp. 140-209, 1988.

[McC, 76] T. J. McCabe, "*A complexity measure*", IEEE Trans. on Software Engineering, vol. SE-2, nº4, pp. 308-320, December 1976.

[McC, 86] E. J. McCluskey, "*Design for testability*", in Fault-Tolerant Computing, Theory and Techniques, D. K. Pradhan editor, Englewood Cliffs: Prentice Hall, pp. 95-183, 1986.

[Mel, 77] P. M. Melliar-Smith, B. Randell, "*Software reliability: the role of programmed exception handling*", ACM SIGPLAN Notices, vol. 12, nº3, pp. 95-100, March 1977; also in Reliable Computer Systems, S. K. Shrivastava editor, Springer-Verlag, pp. 143-153, 1985.

[Mey, 78] J. F. Meyer, "*On evaluating the performability of degradable computing systems*", in Proc. 8nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-8), Toulouse, France, pp. 44-49, June 1978.

[Mil, 87] H. D. Mills, M. Dyer, R. C. Linger, "*Clean room software engineering*", IEEE Software, pp. 19-25, September 1987.

[Mil, 86] D. R. Miller, "*Exponential order statistic models of software reliability growth*", IEEE Trans. on Software Engineering, vol. SE-12, nº1, pp. 12-24, January 1986.

[Min, 67] H. Mine, Y. Koga, "*Basic properties and a construction method for fail-safe logical systems*", IEEE Trans. on Electron. Computers, vol. EC-16, nº6, pp. 282-289, June 1967.

[Mor, 90] L. J. Morell, "*A thery of fault-based testing*", IEEE Trans. on Software Engineering, vol. 16, nº8, pp. 844-857, August 1990.

[Mye, 79] G. J. Myers, "*The Art of Software Testing*", John Wiley & Sons, 1979.

[Nic, 89] M. Nicolaidis, S. Noraz, B. Courtois, "*A generalised theory of fail-safe systems*", in Proc. 19nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-19), Chicago, USA, pp. 398-406, June 1989.

[Nor, 83] D. A. Norman, "*Design rules based on analyses of human error*", Communications of the ACM, vol. 26, nº4, pp. 254-258, April 1983.

[Nta, 88] S. C. Ntafos, "*A comparison of some structural testing strategies*", IEEE Trans. on Software Engineering, vol. SE-14, nº6, pp. 868-874, June 1988.

[Ost, 76] L. J. Osterweil, L. D. Fodsick, "*DAVE – A validation error detection and documentation system for Fortran programs*", Software Practice and Experience, pp. 473-486, October-December 1976.

[PDC, 90] PDCS Workshop Report nºW6, "*Real-time systems, specific closed workshop*", vol. 1, ESPRIT Basic Research Action nº1092, September 1990.

[Pet, 72] W. W. Peterson, E. J. Weldon, "*Error-Correcting Codes*", MIT Press, 1972.

[Pig, 88] P. I. Pignal, "*An analysis of hardware and software availability exemplified on the IBM 3725 communication controller*", IBM J. of Research and Development, vol. 32, nº2, pp. 268-278, March 1988.

[Pow, 88] D. Powell, G. Bonn, D. Seaton, P. Verissimo, F. Waeselynck, "*The Delta-4 approach to dependability in open distributed computing systems*", in Proc. 18nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-18), Tokyo, Japan, pp. 264-251, June 1988.

[Pra, 86] D. K. Pradhan, "*Fault-tolerant multiprocessor and VLSI-based system communication architectures*", in Fault-Tolerant Computing, Theory and Thecniques, D. K. Pradhan editor, englewood Cliffs: Prentice Hall, pp. 467-576, 1986.

[Rab, 89] M. O. Rabin, "*Efficient dispersal of information for security, load balancing and fault tolerance*", Journal of the ACM, vol. 36, nº2, pp. 335-348, April 1989.

[Ram, 84] C. V. Ramamoorthy, A. Prakash, W. T. Tsai, Y. Usuda, "*Software engineering: problems and perspectives*", IEEE Computer, pp. 191-209, October 1984.

[Ran, 75] B. Randell, "*System structure for software fault tolerance*", IEEE Trans. on Software Engineering, vol. SE-1, nº2, pp. 220-232, June 1975.

[Ran, 78] B. Randell, P. A. Lee, P. C. Treleaven, "*Reliability issues in computer system design*", Computing Surveys, vol. 10, nº2, pp. 123-165, June 1978.

[Ran, 86] B. Randell, "*Reliability and security issues in distributed computing systems*", in Proc. 5th Symp. on Reliability in Distributed Software and Database Systems, Los Angeles, pp. 113-118, January 1986.

[Rap, 85] S. Rapps, E. J. Weyuker, "*Selecting software test data using data flow information*", IEEE Trans. on Software Engineering, vol. SE-11, nº4, pp. 367-375, April 1985.

[Ren, 86] D. A. Rennels, "*On implementing fault-tolerance in bynary hipercubes*", in Proc. 16nd IEEE Int. Symp. on Fault Tolerant Computing (FTCS-16), Vienna, Austria, pp.344-349, July 1986.

[RTC, 85] "*Software considerations in airborne systems and equipment certification*", Document nº RTCA/DO-178A, Radio Technical Commission for Aeronautics, March 1985.

[Rot, 67] J. P. Roth, W. G. Bourricius, P. R. Schneider, "*Programmed algorithms to compute tests to detect and distinguish between failures in logic circuits*", IEEE Trans. on Electronic Computers, vol. EC-16, pp. 567-579, October 1967.

[Rud, 85] H. Rudin, "*An informal overview of formal protocol specification*", IEEE Communications Magazine, vol. 23, nº3, pp. 46-52, March 1985.

[Sch, 83] R. D. Schlichting, F. B. Schneider, "*Fail-stop processors: an approach to designing fault-tolerant computing systems*", ACM Trans. on computing Systems, vol. 1, nº3, pp. 222-238, August 1983.

[Sie, 82] D. P. Siewiorek, d. Johnson, "*A design methodology for high reliability systems: the Intel 432*", in D. P. Siewiorek, R. S. Swarz, "*The Theory and Practice of Reliable System Design*", Digital Press, pp. 621-636, 1982.

[Smi, 88] R. M. Smith, K. S. Trivedi, A. V. Ramesh, "*Performability analysis: measures,an algorithm and a case study* ", IEEE Trans. on Computers, vol. 37, nº4, pp.406-417, April 1988.

[Tay, 80] D. J. Taylor, D. E. Morgan, J. P. Black, "*Redundancy in Data structures: improving software fault-tolerance*", IEEE Trans. on Software Engineering, vol. SE-6, nº6, pp. 383-394, November 1980.

[Tha, 78] S. M. Thatte, J. A. Abraham, "*A methodology for functional level testing of microprocessors*", in Proc. 8th IEEE Int. Symp. on Fault Tolerant Computing (FTCS-8), Toulouse, France, pp. 90-95, June 1978.

[Toh, 89] Y. Tohma, K. Tokunaga, S. Nagase, Y. Murata, "*Structural approach to the estimation of the number of residual faults based*

on the hyper-geometric distribution", IEEE Trans. on Software Engineering, vol. SE-15, nº3, pp. 345-355, March 1989.

[Tri, 84] K. S. Trivedi, "*Reliability evaluation for fault-tolerant systems*", in Mathematical Computer Performance and Reliability, G. Iazeolla, P. J. Courtois, A. Hordijk, Eds. Amsterdam: North Holland, pp. 403-414, 1984.

[Wak, 78] J. F. Wakerly, "*Error-Detecting Codes, Self-Checking Circuits and Applications*", New York: Elevier North Holland, 1978.

[Wey, 82] E. J. Weyuker, "*On testing non-testable programs*", The Computer Journal, vol. 25, nº4, pp. 465-470, 1982.

[Wil, 83] T. W. Williams, "*Design for testability – A survey*", Proceedings of IEEE, vol. 71, nº1, pp. 98-112, January 1983.

[Yam, 85] S. Yamada, S. Osaki, "*Software reliability growth modeling: models and applications*", IEEE Trans. on Software Engineering, vol. SE-11, nº12, pp. 1431-1437, December 1985.

[Yau, 75] S. S. Yau, R. C. Cheung, "*Design of self-checking software*", in Proc. 1975 Int. Conf. On Reliable Software, Los angeles, USA, pp. 450-457, April 1975.

[Zie, 76] B. P. Ziegler, "*Theory of modeling and simulation*", New York: John Wiley, 1976.