Sistemas de Tempo-Real

Notas de curso realizado em Agosto de 2006 na
Universidade Federal do Rio Grande do Norte, Natal, Brasil

Francisco Vasques
Faculdade de Engenharia da
Universidade do Porto
http://www.fe.up.pt/~vasques

Sistemas de Segurança Crítica

- **Introdução**
  - Exemplo de Acidentes
    - » Automatização do "London Ambulance Service"
    - » Fracasso dos Mísseis "Patriot"
    - » Acidente com Airbus A-320 em Habsheim
    - » Lições Retiradas
  - Conceitos Básicos e Terminologia
- **Análise da Ocorrência de Situações Perigosas ("*Hazard Analysis*")**
- **Análise do Risco ("*Risk Analysis*")**
- **Tolerância a Falhas**

Automatização do "London Ambulance Service"

- **O "London Ambulance Service" (LAS)**
  - Sistema de ambulâncias, cobrindo uma área de 250km2 para servir 7-12 milhões de potenciais utilizadores;
  - Trata diariamente 5000 pacientes, 2000-2500 chamadas telefónicas, das quais 1300-1600 de emergência;
- **Automatização do LAS**
  - No centro do LAS está um centro de despacho responsável pelo atendimento de chamadas, localização de ambulâncias, despacho da ambulância que melhor possa servir cada paciente e monitorização do seu estado ("a caminho", "no local", "em direcção ao hospital", "livre", etc.

Automatização do "London Ambulance Service"

- **Caso de estudo**
  - Transformou-se num caso de estudo acerca de "como não conceber, desenvolver ou implementar um sistema crítico em termos de segurança";

Automatização do "London Ambulance Service"

- **Arranque do LAS**
  - Dia de arranque: 26/10/1992, 07:00h;
    O centro de comando foi modificado para receber um novo sistema "paperless", colocando o sistema anterior inoperacional;
  - O sistema automatizado estava baseado num AVLS ("Automatic Vehicle Location System") para identificar ambulâncias, para efectuar alocação de recursos e para comunicar com os MDT ("Mobile Data Terminals") dos veículos;
  - As tripulações foram informadas para manterem um tráfego de voz reduzido com a central.

Automatização do "London Ambulance Service"

- **Arranque do LAS**
  - 10:00h.

    O numero de chamadas aumenta e o sistema começa a ter dificuldades. O AVLS não consegue manter actualizada a posição e o status das ambulâncias e começa a despachar incorrectamente (e mesmo a despachar múltiplas ambulâncias para o mesmo local);

Automatização do "London Ambulance Service"

- **Arranque do LAS**
  - Notificações de excepção começam a ser geradas em cascata. A dificuldade de resposta a um grande numero de excepções pela parte dos operadores, leva a que as mais antigas deixem de estar visíveis nos terminais, provocando a geração de um numero crescente de mensagens de excepção;
  - O aumento inerente da lentidão do sistema, leva a que os pacientes coloquem chamadas suplementares nas filas de espera, aumentando ainda mais a sua lentidão.

Automatização do "London Ambulance Service"

- **Frustração**
  - Sob pressão, as tripulações deixaram de informar o centro de despacho acerca do seu status (via MDT).
  - A aparente falta de recursos para alocar, sobrecarregou a execução do software de alocação de recursos, aumentando ainda mais o atraso do sistema;
  - As tripulações das ambulâncias acostumadas a atrasos de alguns minutos na resposta a chamadas de paciente, começam a responder com várias horas de atraso.

Automatização do "London Ambulance Service"

- **2 casos tiveram ampla repercussão na imprensa:**
  - Uma pessoa cuja mãe tivera um ataque de coração ao inicio da tarde, decidiu levá-la de taxi para o hospital após 6h de espera. Às 2:00AM recebeu uma chamada telefónica, a perguntar se ainda precisava de assistência.
  - Uma ambulância respondendo a uma chamada colocada 8h antes, verificou que o corpo tinha acabado de ser retirado por uma agência funerária...

Automatização do "London Ambulance Service"

- **Após o colapso inicial...**
  - O despacho do LAS passou a semi-manual, com as chamadas atendidas e impressas automaticamente.
  - A selecção passou a ser efectuada manualmente em conjunto com a estação mais próxima do incidente, e enviada para o MDT da ambulância seleccionada.
  - Este sistema funcionou razoavelmente até às 2:00 de 4/11 (9 dias após o arranque), começando então a ficar mais lento, até parar;

Automatização do "London Ambulance Service"

- **Após o colapso inicial...**
  - A re-inicialização dos computadores não resolveu o problema. O servidor de backup não pode ser utilizado pois não tinha sido totalmente implementado, nem testado em modo semi-manual;
  - Em consequência as chamadas deixaram de poder ser impressas, e deixou de haver comunicação com os MDTs.
  - Os operadores passaram a sistema manual, garantindo que todas as chamadas gravadas em fita magnética foram atendidas, com a mobilização das ambulâncias efectuada via rádio e telefone.

Automatização do "London Ambulance Service"

- **O que correu mal?**
  - "*On 26 and 27 October 1992 the computer system itself did not fail in a technical sense. Response times did on occasions become unacceptable, but overall the system did what it had been designed to do*."

    Relatório da comissão de inquérito, 1993.

Automatização do "London Ambulance Service"

- **O que de facto correu mal**
  - No arranque do sistema, o software estava incompleto, não tinha sido ajustado, nem tinha sido completamente testado (nomeadamente sob cenários com a carga adequada).
  - O sistema de backup estava inoperacional, e o sistema anterior (baseado em papel) tinha sido abandonado.
  - Nestas condições, restringir a utilização do rádio e depender unicamente de um sistema automático foi correr um risco desmesurado.

Automatização do "London Ambulance Service"

- **O que de facto correu mal**
  - Os factores humanos foram totalmente negligenciados:
    - » mudança súbita na configuração do local de trabalho;
    - » fim absoluto do habitual sistema de backup "em papel";
    - » fim da comunicação informal com as tripulações e as estações locais;

  levaram a que os operadores se sentissem fora do sistema, sem poderem influenciar a sua evolução.

Automatização do "London Ambulance Service"

- **O que de facto correu mal**
  - A concepção do sistema falhou, porque foi considerado que a informação sobre a localização e o status de cada ambulância estaria sempre disponível;
    - » existiam problemas com as rotinas de comunicação e as interfaces eram fracas;
  - O sistema era pouco fiável, havia bloqueios frequentes com recurso habitual à sua re-inicialização.

Automatização do "London Ambulance Service"

- **O que de facto correu mal**
  - A interface com o operador era muito fraca:
    - » impossível identificar chamadas repetidas;
    - » impossível prioritizar mensagens urgentes;
    - » quando o écran ficava cheio de mensagens, as mais antigas desapareciam;
    - » erros no software de alocação de recursos;
  - Tempos de resposta lentos para certas operações baseadas em interfaces gráficas.

Automatização do "London Ambulance Service"

- **O que de facto correu mal**
  - O crash de 4/11 foi devido a um erro de manutenção.
  - Uma rotina de teste que efectuava uma alocação de memória cada vez que uma ambulância era seleccionada, foi esquecida no sistema.
  - Ao fim de 3 semanas, a capacidade de memória esgotou, levando à paragem do sistema.

Automatização do "London Ambulance Service"

- **Principais conclusões**
  - Uma das principais causas do colapso foi a pressão política para ter o sistema a funcionar pelo menor preço e no mais curto espaço de tempo possível;
  - Infelizmente, a Comissão de Inquérito não fez nenhuma avaliação detalhada do custo final associado ao colapso do sistema.

Automatização do "London Ambulance Service"

- **Recomendação da Comissão de Inquérito**
  - "*The development of a strategy for the future of Computer Aided Dispatch (CAD) within the London Ambulance System (LAS) must involve a full process of consultation between management, staff, trade union representatives and the Service's information technology advisers* […]"
  - "*What is certain is that the next CAD system must be made to fit the Service's current or future organisational structure and agreed operational procedures. This was not the case with the current CAD.*"

Sistemas de Segurança Crítica

- **Introdução**
  - Exemplo de Acidentes
    - » Automatização do “London Ambulance Service”
    - » Fracasso dos Mísseis “Patriot”
    - » Acidente com Airbus A-320 em Habsheim
    - » Lições Retiradas
  - Conceitos Básicos e Terminologia
- **Análise da Ocorrência de Situações Perigosas (“*Hazard Analysis*”)**
- **Análise do Risco (“*Risk Analysis*”)**
- **Tolerância a Falhas**

Fracasso dos Mísseis "Patriot"

- **Cenário: Guerra do Golfo, 1991;**
  - Em 25/2/1991 um míssil Scud passou através das defesas anti-míssil Patriot, provocando a morte ou ferimentos a 28 e 98 soldados, respectivamente.
  - A avaria que levou a este acidente está documentada, e tem uma explicação simples: concepção deficiente do sistema de controlo.
  - O sistema de controlo de um míssil Patriot examina em contínuo o céu para detectar eventuais alvos. Caso algo seja detectado, estreita a "zona de detecção" em torno do objecto detectado para identificação. Em seguida, caso seja identificado um alvo, deve-o seguir com precisão.

Fracasso dos Mísseis "Patriot"

- **O que correu mal...**
  - O sistema foi concebido para funcionar durante no máximo algumas horas, antes de ser transportado para uma nova localização (cenário de guerra Europeu).
  - No Golfo, o sistema estava em funcionamento contínuo há mais de 100h, provocando uma perda de precisão dos cálculos efectuados.
  - Devido a esta perda de precisão, a "zona de detecção" desviou-se do alvo. Em consequência, o míssil Scud atravessou o sistema anti-mísseis.

Fracasso dos Mísseis "Patriot"

- **Razões técnicas**
  - O cálculo da localização do míssil era efectuada com base nos valores de velocidade e de tempo (inteiros).
  - A precisão dos cálculos estava limitada pelas conversões inteiros/reais e pelo facto de os registos serem unicamente de 24 bits.
  - Em consequência, a precisão da "zona de detecção" era inversamente proporcional à velocidade do míssil e ao tempo de funcionamento do sistema.

Fracasso dos Mísseis "Patriot"

- **Conclusões**
  - Este acidente pode ser interpretado como consequência de um erro de programação, visto ter sido provado que a especificação funcional dos requisitos estava correcta.
  - A modificação das condições de operação (ambiente) colocou em evidência um modo de avaria crítico, que anteriormente não se tinha manifestado.
  - Falhas conhecidas (como a que causou o desvio da zona e detecção) e procedimentos de "desenrasca" (como a necessária re-inicialização periódica) devem estar devidamente documentados.

Sistemas de Segurança Crítica

- **Introdução**
  - Exemplo de Acidentes
    - » Automatização do "London Ambulance Service"
    - » Fracasso dos Mísseis "Patriot"
    - » Acidente com Airbus A-320 em Habsheim
    - » Lições Retiradas
  - Conceitos Básicos e Terminologia
- **Análise da Ocorrência de Situações Perigosas ("*Hazard Analysis*")**
- **Análise do Risco ("*Risk Analysis*")**
- **Tolerância a Falhas**

Acidente com Airbus A-320 em Habsheim

- **Sistema FBW ("Fly-by-Wire")**
  - O Airbus A320-100 foi o primeiro avião comercial a implementar um sistema FBW.
- **As principais características do FBW no A320-100 são:**
  - Facilidade no treino de pilotagem (interface de pilotagem idênticas em diferentes tipos de Airbus)
  - Simplicidade na manutenção, devido à modularidade do sistema de controlo, que permite a troca simples de qualquer dos sistemas computacionais de bordo;
  - Implementação de algoritmos de controlo avançados, para a obtenção de "neutral static stability", ou seja, a tendência de manter o perfil de voo ("attitude") quando os pilotos libertam os comandos;

Acidente com Airbus A-320 em Habsheim

- **As principais características do FBW no A320-100 são:**
  - Devido ao facto de o sistema de controlo ser maioritariamente definido em software, permite a sua fácil revisão para inclusão de novos requisitos resultantes da experiência operacional;
  - Sistema de detecção e monitorização de falhas que armazena e disponibiliza dados sobre todas as avarias no final de cada voo;
  - Segurança reforçada contra erros acidentais de pilotagem, através da definição de "flight envelopes". Este sistema impede velocidades superiores ou inferiores a limites pré-estabelecidos (mesmo no caso de descidas abruptas), diminuindo o risco de avarias estruturais;

Acidente com Airbus A-320 em Habsheim

- **Acidente de Habsheim**
  - Em 26/6/1988, um Airbus A320-100 teve um acidente em Habsheim, Mulhouse, França, durante um voo de demonstração;
  - Estava previsto efectuar uma primeira passagem “lenta” a 30m de altitude e 25º de inclinação (“near stall”), e em seguida, efectuar uma 2ª passagem “rápida” a 30 m de altitude;
  - A 2ª passagem não foi efectuada, devido ao facto de o avião não ter conseguido subir no final da 1ª passagem.

Acidente com Airbus A-320 em Habsheim

» Como resultado do acidente (e do fogo que deflagrou), morreram 4 dos 130 passageiros (34 feridos). Dos 6 membros da tripulação, 4 ficaram feridos.

Acidente com Airbus A-320 em Habsheim

- **O que correu mal (conclusão da Comissão de Inquérito):**
  - Altitude de voo inferior à dos obstáculos existentes;
  - Velocidade demasiado reduzida (para maximizar angulo de ataque), com motores em estado "*idle*";
  - Aceleração tardia.

    "*Too low, too slow, too late*"
  - Conclusão: Erro Humano, visto ter sido apurado que o avião se encontrava em correcto estado de funcionamento.

Acidente com Airbus A-320 em Habsheim

- **O que correu mal (conclusões do piloto Michel Asseline)**
  - As sequências de treino efectuadas e as garantias do construtor, levaram a tripulação a um estado de sobre-confiança acerca da manobrabilidade do aparelho;
    - » Por exemplo, os manuais de voo garantiam que o voo a elevados graus de inclinação eram seguros. Os pilotos poderiam colocar o "flight stick" na posição extrema, que o software de controlo garantiria a não ultrapassagem do "flight envelope" seguro.
  - A especificação das altitudes correctas de passagem (30m / 100m) não foi incluída no plano de voo.

Acidente com Airbus A-320 em Habsheim

- **O que correu mal (conclusões do piloto Michel Asseline)**
  - A tripulação não compareceu na reunião de segurança (comparência obrigatória), devido a erro da Air France;
    - » Nesta reunião teriam tido conhecimento das limitações impostas ao voo de demonstração.
  - O dossier de voo (efectuado pelos serviços de segurança) foi entregue tardiamente à tripulação. Como resultado, esta só soube que havia árvores no final da pista com 10-15m de altura quando estavam em pleno voo...

Acidente com Airbus A-320 em Habsheim

- **O que correu mal (conclusões do piloto Michel Asseline)**
  - O piloto pensava estar a voar a 30m de altitude, quando se encontrava a voar unicamente a 10m de altitude. Duas razões foram apuradas para este facto:
    - » O aeroporto de Habsheim sendo menor que os aeroportos tradicionais, falseou as referências visuais do piloto;
    - » O altímetro utilizado pelo piloto apresentava um erro de 25m.
      A Air France argumenta que o piloto não efectuou a sua correcta calibração, enquanto que o piloto suspeita de uma falha no software de comunicação entre sistemas (conforme relatado noutros casos).

Acidente com Airbus A-320 em Habsheim

- **O que correu mal (análise posterior)**
  - Através da análise do acidente, parece claro que este não se deveu a uma avaria no software, no sentido de "ocorrência de falha de software que tenha provocado um desvio do comportamento especificado".
    - » No entanto este ponto não foi devidamente clarificado, devido a uma clara obstrução da Air France à completa análise das causas do acidente.
  - O que parece claro é que os requisitos para os sistemas embarcados foram inadequadamente especificados, para o caso de o avião ser operado em condições limite.

Acidente com Airbus A-320 em Habsheim

- **O que correu mal (análise posterior)**
  - Adicionalmente, o sistema de controlo tem um impacto severo na forma de pilotagem. A sobre-confiança e a dependência do piloto no sistema de controlo foram uma causa clara para o acidente.
  - É possível o construtor defender que “todos os sistemas tiveram um desempenho conforme o especificado”.
  - No entanto, através da análise das sequências de diversos acidentes, fica evidente que a forma como se comportaram os sistemas computacionais embarcados teve um impacto evidente em cada um dos acidentes.

Sistemas de Segurança Crítica

- **Introdução**
  - Exemplo de Acidentes
    - » Automatização do "London Ambulance Service"
    - » Fracasso dos Mísseis "Patriot"
    - » Acidente com Airbus A-320 em Habsheim
    - » Lições Retiradas
  - Conceitos Básicos e Terminologia
- **Análise da Ocorrência de Situações Perigosas ("*Hazard Analysis*")**
- **Análise do Risco ("*Risk Analysis*")**
- **Tolerância a Falhas**

Principais Lições Retiradas

- **O software avaria**
  - Um sistema avaria quando o serviço prestado não está em conformidade com a especificação.
  - A avaria pode ser devida à activação de uma falha de concepção (falha latente). Se esta falha reside no software, então o software avaria.

Principais Lições Retiradas

- **Contra argumentos (o software não avaria…):**
  - O código fonte tem a sua própria especificação, logo "o serviço prestado está de acordo com a especificação"…
  - Errado, porque:
    - » O código objecto pode não ser uma transformação correcta do código fonte (avaria no compilador)
    - » O código fonte do software pode não implementar o requerido por uma especificação de nível superior (exemplo de avaria nos mísseis Patriot)

Principais Lições Retiradas

- **O software é uma abstracção, que não funciona por si próprio, logo não pode avariar**
  - No entanto, quando implementado num sistema deixa de ser uma abstracção, e passa a fazer parte de um sistema com uma implementação física (exemplo de avaria por deficiente alocação de memória no LAS).
- **O software não se desgasta**
  - Correcto, mas irrelevante. Isto só significa que as causas de uma avaria no software diferem das causas de uma avaria no hardware

Principais Lições Retiradas

- **O software é sempre parte integrante de um sistema mais vasto**
  - A avaliação de desempenho do software deve ser efectuada para o ambiente de execução pretendido
  - A verificação do software deve ser efectuada durante as fases de especificação e programação
  - A validação do software deve ser efectuada em operação.

Principais Lições Retiradas

- **As falhas de concepção podem ser introduzidas em diferentes níveis**
  - Quando os requisitos escritos não estão em conformidade com os requisitos "reais".
    - » Estes requisitos são por vezes expressos de uma forma implícita ou informal. "*It's just what I asked for, but not what I want*".
  - Quando a especificação não está em conformidade com os requisitos escritos
  - Quando o código fonte não está em conformidade com os requisitos
    - » Ex.: avaria devida ao cálculo de janela nos mísseis Patriot
  - Quando o código máquina não está de acordo com o código fonte
    - » Ex.: devido a avaria ou documentação insuficiente de compilador

Principais Lições Retiradas

- **A especificação errada de requisitos é uma das maiores causas de acidentes**
  - O sistema pode ter sido adequadamente verificado (garantia de conformidade com a especificação) mas inadequadamente validado (garantia de conformidade com os requisitos)

    "*The software is behaving to specification, but its behavior causes sufficient problems for the user for it to count as failure*"

Principais Lições Retiradas

- **Falhas de concepção singulares raramente são a causa de um acidente**
  - Na maior parte dos casos, em sistemas complexos, os acidentes ocorrem quando múltiplas avarias interagem de uma forma não expectável;
  - No caso particular do software, as falhas nem sempre podem ser localizadas num simples módulo. Por exemplo,
    - » no caso de deficiências na especificação de requisitos;
    - » no caso de desempenho inadequado do sistema (por exemplo, no colapso do LAS);
    - » ou no caso de interacções complexas entre múltiplas interfaces.

Principais Lições Retiradas

- **O operador é parte do sistema**
  - Sempre que um erro humano possa provocar acidentes, este facto deve ser tomado em consideração na avaliação da segurança do sistema
  - Exemplo:
    - » Inicialmente vários dos acidentes dos Airbus foram atribuídos a erros humanos;
    - » Posteriormente foram detectadas insuficiências graves a nível da concepção das interfaces homem-máquina.
- **O erro humano é inevitável**
  - A concepção de um sistema deve garantir a máxima robustez contra erros humanos e fornecer um ambiente de trabalho para o operador que minimize a probabilidade de erro
  - O erro humano é frequentemente uma desculpa para uma fraca concepção.

Principais Lições Retiradas

- **O erro do operador é habitualmente uma desculpa para uma deficiente concepção**
  - "...*technically, nothing at all failed*..."
  - A intenção é, por vezes, culpar o operador, por forma a ilibar o fabricante/fornecedor do sistema;
  - No entanto, é sempre pertinente analisar cuidadosamente a razão pela qual o operador errou...

Principais Lições Retiradas

- **Maior complexidade do sistema implica normalmente a sua menor fiabilidade**
  - A interacção (acoplamento) entre diferentes modos de funcionamento de vários sistemas significa que, por vezes, é extraordinariamente difícil analisar a possibilidade de ocorrência de situações perigosas;
  - A utilização de sistemas computacionais tende a incrementar a complexidade dos sistemas e o acoplamento entre os seus diferentes modos de funcionamento.

Principais Lições Retiradas

- **Precaução ilimitada**
  - Foi já diversas vezes demonstrado que o software não pode ser quantitativamente certificado para o nível de integridade requerido pela maior parte dos sistemas de segurança crítica;
  - Logo, a regra óbvia será: "*never let software be a single point of failure*".

Principais Lições Retiradas

- **Conclusões**
  - Por razões comerciais (vantagens competitivas), meios computacionais serão cada vez mais utilizados para suportar aplicações críticas.
  - Em consequência, um investimento elevado deve ser colocado na melhoria progressiva da sua confiança no funcionamento.
  - A análise da ocorrência de acidentes e a compreensão das suas causas devem ser o primeiro passo para a melhoria da confiança no funcionamento de sistemas críticos.
  - Para tal torna-se necessária a existência de uma investigação independente, capaz de gerar relatórios credíveis acerca das causas reais dos acidentes.

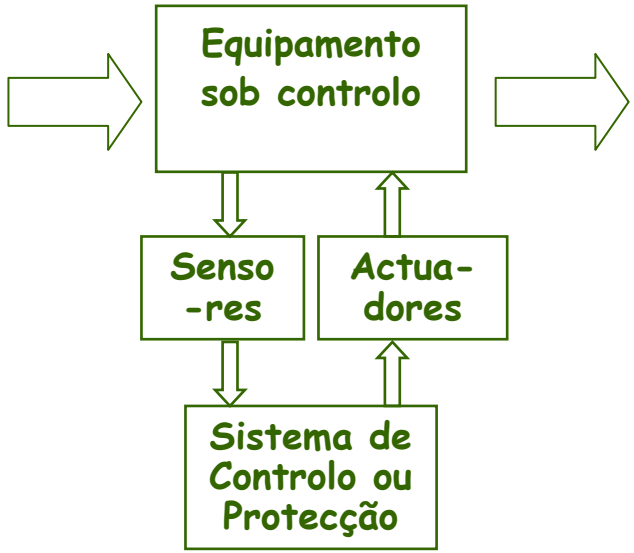
Sistemas de Segurança Crítica

- **Introdução**
  - Exemplo de Acidentes
  - Conceitos Básicos e Terminologia
    - » Sistemas Computacionais e Segurança ("*Safety*")
    - » Critérios de Concepção / Desenvolvimento
- **Análise da Ocorrência de Situações Perigosas ("*Hazard Analysis*")**
- **Análise do Risco ("*Risk Analysis*")**
- **Tolerância a Falhas**

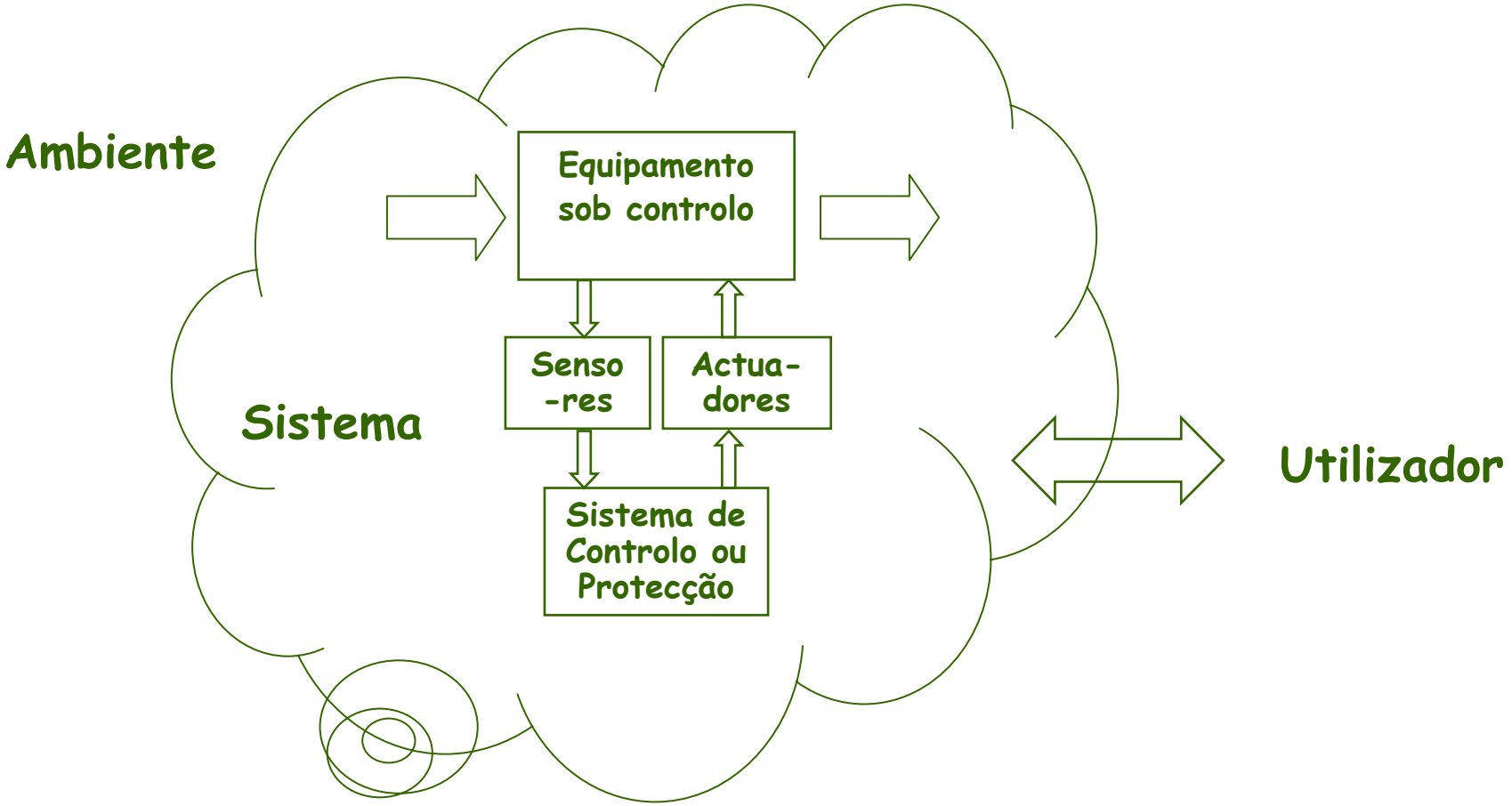
- **Definição de Segurança ("*Safety*")**
  - Confiança no funcionamento relativamente à não ocorrência de avarias catastróficas (avarias que colocam em causa a vida humana ou o ambiente).
- **O que é um Sistema de Segurança Crítica ("*Safety-Critical System*")?**
  - Um sistema de segurança crítica ("*safety-critical system*") é aquele no qual a segurança (não ocorrência de avarias catastróficas) é garantida, em tempo de concepção/ implementação .

- **Sobre a Segurança**
  - A segurança de um sistema depende essencialmente de uma concepção / implementação segura, e não de adicionar "segurança" a um sistema já desenvolvido.
  - A segurança de um sistema equaciona o sistema como um todo, e não como um conjunto de subsistemas ou componentes.

- **Sistema de Controlo.**
  - Determina a forma de operação de um sistema (simples <-> complexo).
  - Caso especificado, também fornece funções de segurança (sistema de segurança crítica);
- **Sistema de Protecção.**
  - Utiliza sensores para detectar condições de falha e produz saídas tendentes a minorar (anular) os seus efeitos;
  - Caso especial: "*shutdown system*".



Sistemas Computacionais e Segurança



- **Sistema, Ambiente, Utilizador**
  - Um Sistema é uma entidade que interage ou interfere com outras entidades, i.e., com outros sistemas. Estes outros sistemas constituem o seu meio envolvente (Ambiente).
  - Um Utilizador de um sistema é aquela parte do ambiente 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

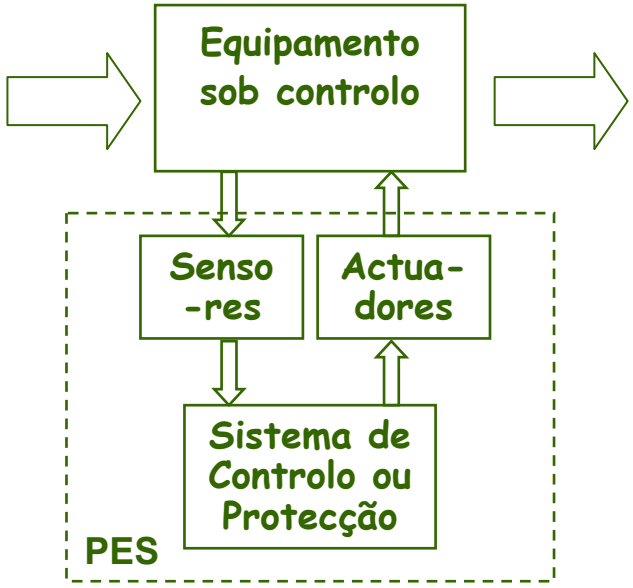
Sistemas Computacionais e Segurança

- **Tendência: "*Software is a pervasive Enabling technology*"**
  - » Actualmente, múltiplas funções de segurança crítica são já suportadas por sistemas computacionais;
  - » Os sistemas embarcados terão um papel dominante na nossa interacção com sistemas computacionais, mais cedo do que o esperado;
  - » O desenvolvimento de sistemas embarcados será brevemente um dos maiores clientes de tecnologia de segurança crítica (hardware/software/...)

Sistemas Computacionais e Segurança

- **Utilização de Sistemas Computacionais para o suporte de Aplicações de Segurança Crítica**
  - Incremento substancial desde os anos 70, devido às potencialidades e ao reduzido custo destes componentes
- **Gama de aplicação em Sistemas de Segurança Crítica:**
  - Desde sofisticados sistemas aviónicos até controladores de máquinas de lavar; ou desde controladores para centrais nucleares até sistemas de ABS.
  - A utilização de microprocessadores mede-se na escala dos milhões de unidades para aplicações domésticas / aplicações no ramo automóvel ou na escala das unidades para sistemas dedicados, por exemplo na área do controlo industrial.

- **"Programmable Electronic Systems" (PES)**
  - Denominação tradicional para equipamento de controlo/ protecção baseado em sistema computacional, sob a forma de:
    - » Computadores convencionais;
    - » Micro-controladores
    - » Controladores Lógicos Programáveis (PLCs)



- **Vantagens na utilização de PES para o suporte de Aplicações de Segurança Crítica**
  - » Elevada capacidade de processamento (algoritmos complexos, grande n.º de entradas/saídas, etc.), o que permite a implementação de funções de controlo complexas (impossíveis de implementar de outra forma);
  - » Operação do sistema computacional caracterizada por: alta velocidade / baixo consumo / dimensão física reduzida;
  - » Elevada fiabilidade de hardware, devido a um elevado grau de integração (redução do numero de interfaces externas susceptíveis a interferências);

- **Vantagens na utilização de PES para o suporte de Aplicações de Segurança Crítica**
  - » Elevada flexibilidade para evolução do sistema (implicando a redução de custos);
  - » O custo associado a um PES é habitualmente menor dos que os custos associados a sistemas similares;
  - » A disponibilidade de uma considerável capacidade de processamento permite a implementação de rotinas sofisticadas para diagnóstico, monitorização, verificação de gamas de funcionamento, encravamentos, etc.

- **Desvantagens na utilização de PES**
  - Elevada complexidade, quer dos componentes de hardware utilizados (o mais simples micro-controlador é composto por milhares de componentes), quer dos componentes de software:
    - » maior dificuldade na concepção ⇨ maior probabilidade para a existência de erros de concepção;
    - » maior dificuldade para as operações de teste ⇨ maior probabilidade de ocorrência de falhas não detectadas;
    - » maior dificuldade de compreensão da operação do sistema ⇨ maior probabilidade de ocorrência de erros humanos na instalação, manutenção ou utilização.

- **Desvantagens na utilização de PES**
  - O comportamento de um sistema computacional (PES) é extraordinariamente difícil de prever (identificação dos seus modos de avaria).
    - » O numero de possíveis modos de avaria do hardware é “quase ilimitado”, pelo que operações exaustivas de teste não são possíveis.
    - » Exceptuando os casos mais simples, qualquer programa é demasiado complexo para ser exaustivamente testado (devido às interacções com o SO, interfaces, etc.).

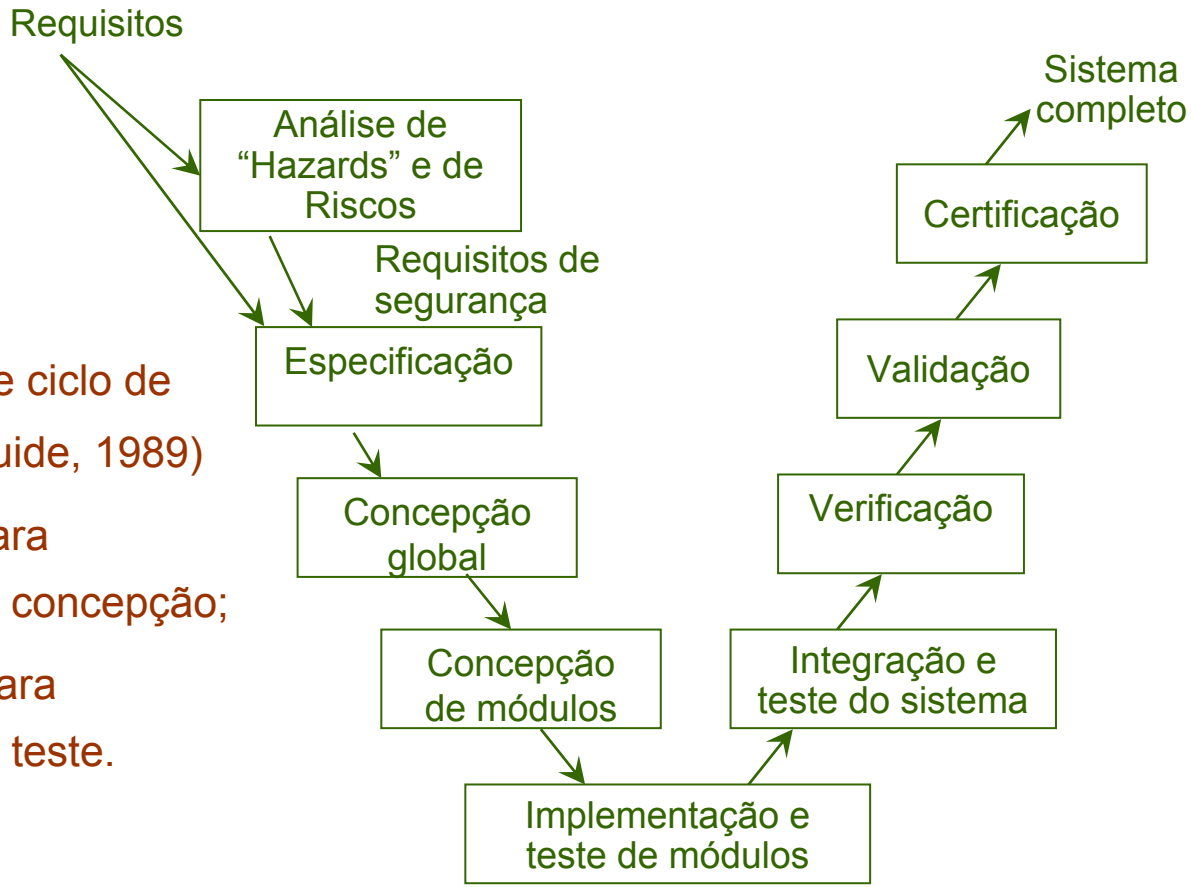
Sistemas de Segurança Crítica

- **Introdução**
  - Exemplo de Acidentes
  - Conceitos Básicos e Terminologia
    - » Sistemas Computacionais e Segurança ("*Safety*")
    - » Critérios de Concepção / Desenvolvimento
- **Análise da Ocorrência de Situações Perigosas ("*Hazard Analysis*")**
- **Análise do Risco ("*Risk Analysis*")**
- **Tolerância a Falhas**

Critérios de Concepção / Desenvolvimento

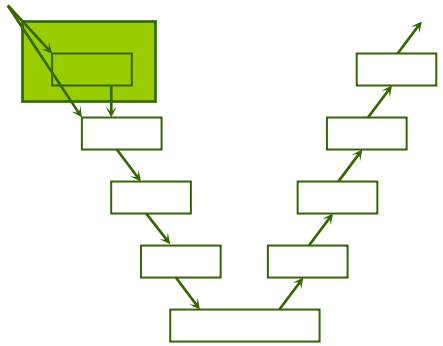
- **Noção de ciclo de vida de desenvolvimento:**

  - Exemplo: Modelo de ciclo de vida em V (Starts guide, 1989)
    - » "Top-Down" para actividades de concepção;
    - » "Bottom-Up" para actividades de teste.



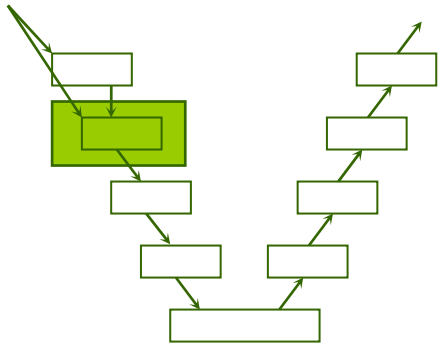
Critérios de Concepção / Desenvolvimento

- **Requisitos Funcionais e Requisitos de Segurança**
  - A Análise de Situações Perigosas ("Hazard Analysis") e de Riscos gera os Requisitos de Segurança do sistema
    - » definição do que o sistema deve fazer (ou não deve fazer) para ter um funcionamento seguro.



Critérios de Concepção / Desenvolvimento

- **Especificação**
  - A partir da definição de requisitos (funcionais, não funcionais e de segurança) deverá ser gerada a Especificação do sistema, incluindo:
    - » definição do Nível de Integridade de Segurança do sistema;
    - » definição de um conjunto de medidas a tomar para garantir a segurança do sistema.



Critérios de Concepção / Desenvolvimento

- **Níveis de Integridade de Segurança (SIL)**
  - Para reflectir a importância da correcta operação de um sistema definem-se diferentes níveis de integridade.
  - O nível de integridade seleccionado para um sistema determina os métodos de desenvolvimento que devem ser utilizados durante o seu ciclo de vida.

Table A.1 – SIL-table

Tolerable Hazard Rate THR per hour and per function	Safety Integrity Level
$10^{-9} \leq THR < 10^{-8}$	4
$10^{-8} \leq THR < 10^{-7}$	3
$10^{-7} \leq THR < 10^{-6}$	2
$10^{-6} \leq THR < 10^{-5}$	1

Critérios de Concepção / Desenvolvimento

- **Concepção e Implementação**

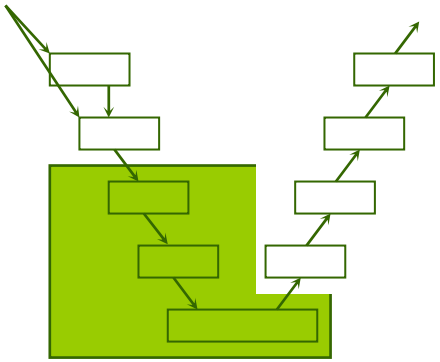
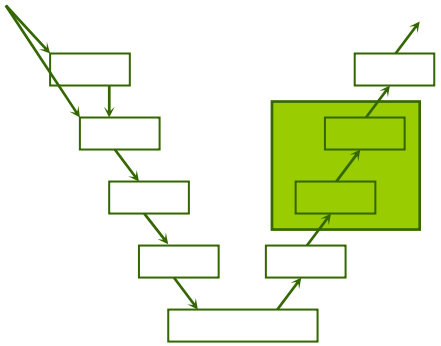


Table A.15 – Programming Languages (D.4)
Referenced by clause 10

TECHNIQUE/MEASURE		Ref	SWS IL0	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1.	ADA	B.62	R	HR	HR	R	R
2.	MODULA-2	B.62	R	HR	HR	R	R
3.	PASCAL	B.62	R	HR	HR	R	R
4.	Fortran 77	B.62	R	R	R	R	R
5.	'C' or C++ (unrestricted)	B.62	R	-	-	NR	NR
6.	Subset of C or C++ with coding standards	B.62 B.38	R	R	R	R	R
7.	PL/M	B.62	R	R	R	NR	NR
8.	BASIC	B.62	R	NR	NR	NR	NR
9.	Assembler	B.62	R	R	R	-	-
10.	Ladder Diagrams	B.62	R	R	R	R	R
11.	Functional Blocks	B.62	R	R	R	R	R
12.	Statement List	B.62	R	R	R	R	R

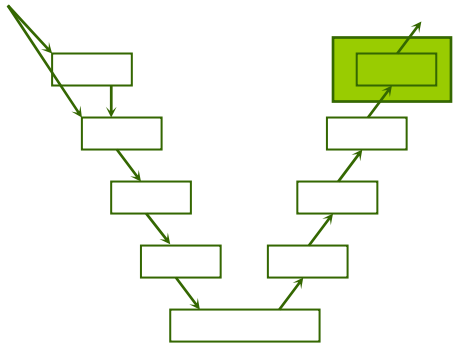
Critérios de Concepção / Desenvolvimento

- **Verificação e Validação**
  - A especificação errada de requisitos é uma das maiores causas de acidentes:
    - » O sistema pode ter sido adequadamente verificado (garantia de conformidade com a especificação)...
    - » mas inadequadamente validado (garantia de conformidade com os requisitos).



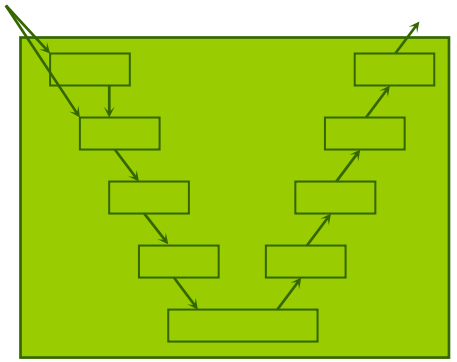
Critérios de Concepção / Desenvolvimento

- **Certificação**
  - É necessário que uma entidade externa certifique que todo o sistema crítico foi concebido e desenvolvido através da utilização de procedimentos seguros.



Critérios de Concepção / Desenvolvimento

- **Âmbito da Segurança**
  - A garantia da segurança de um sistema não está unicamente ligada ao hardware e/ou software utilizados, envolvendo todos os aspectos ligados ao ciclo de vida do sistema, desde a sua concepção, até à sua instalação, utilização e manutenção.



Sistemas de Segurança Crítica

- **Introdução**
- **Análise da Ocorrência de Situações Perigosas ("*Hazard Analysis*")**
  - Análise de Modos de Avaria e Efeitos ("*Failure Modes and Effects Analysis*" - *FMEA*)
  - Estudos de Operabilidade e de Situações Perigosas ("*Hazard and Operability Studies*" - *HAZOP*)
  - Análise por Árvore de Falhas ("*Fault Tree Analysis*" - *FTA*)
- **Análise do Risco ("*Risk Analysis*")**
- **Tolerância a Falhas**

Análise da Ocorrência de Situações Perigosas

- **Avaliação Qualitativa**
  - Uma Situação Perigosa ("Hazard") é uma situação na qual existe um perigo real ou potencial para a vida humana ou para o ambiente.
  - "*Hazard Analysis*": Identificação de cadeia(s) de acontecimentos conducentes à ocorrência de situações perigosas
    - » Identificação sistemática de todas as possíveis ameaças contra a Segurança (Análise Qualitativa).

Análise de Modos de Avaria e Efeitos - FMEA

- **Metodologia FMEA**
  - Selecciona cada um dos componentes (ou funções) do sistema, e determina quais os seus modos de avaria;
  - Considerando individualmente cada modo de avaria, "segue" os seus efeitos para determinar as suas consequências.
    - » Pressupostos ⇨ modos de avaria de cada componente;
    - » Análise ⇨ consequências de cada avaria individual.

Análise de Modos de Avaria e Efeitos - FMEA

- **Vantagens / Desvantagens**
  - Pontos fortes
    - » Detecção dos casos em que uma simples avaria pode resultar numa situação perigosa;
    - » Pode ser aplicada em diferentes níveis do sistema, e com diferentes níveis de detalhe;
    - » Fomenta o espírito de equipa e o conhecimento detalhado do sistema pelos membros da equipa que efectua a análise.

Análise de Modos de Avaria e Efeitos - FMEA

- **Vantagens / Desvantagens**
  - Pontos fracos
    - » Não consideração da avaria simultânea de múltiplos componentes;
    - » Como existem modos de avaria que não resultam em situações perigosas, a análise exaustiva desses casos é desnecessária.

Análise de Modos de Avaria e Efeitos - FMEA

- **Exemplo**

[Storey, 96]

FMEA for a microswitch

Ref No.	*Unit*	*Failure mode*	*Possible cause*	*Local effects*	*System effects*	*Remedial action*
1	Tool guard switch	Open-circuit contacts	(a) faulty component (b) excessive current (c) extreme temperature	Failure to detect tool guard in place	Prevents use of machine – system fails safe	Select switch for high reliability and low probability of dangerous failure Rigid quality control on switch procurement
2		Short-circuit contacts	(a) faulty component (b) excessive current	System incorrectly senses guard to be closed	Allows machine to be used when guard is absent – dangerous failure	Modify software to detect switch failure and take appropriate action
3		Excessive switch-bounce	(a) ageing effects (b) prolonged high currents	Slight delay in sensing state of guard	Negligible	Ensure hardware design prevents excessive current through switch

Estudos de Operabilidade e de Situações Perigosas - HAZOP

- **Metodologia HAZOP**
  - Desenvolvida no âmbito da industria química (ICI, 60s). Utiliza uma série de "palavras chave" para investigar os efeitos de desvios das condições normais de operação, durante cada fase de funcionamento do sistema;
    - Exemplo: "O que aconteceria se…"
  - Uma análise HAZOP é baseada numa investigação rigorosa e sistemática de cada potencial desvio identificado.

Estudos de Operabilidade e de Situações Perigosas - HAZOP

- **Metodologia HAZOP**
  - Os estudos de HAZOP são tipicamente conduzidos por equipas multidisciplinares de 6-8 engenheiros.
  - Partindo de uma especificação básica do sistema, a equipa investiga o efeito de potenciais desvios no funcionamento normal do sistema;
  - Para cada desvio, é colocada uma série de questões: “porquê? “quais as consequências”?

Estudos de Operabilidade e de Situações Perigosas - HAZOP

- **Metodologia HAZOP**
  - Para cada potencial situação perigosa identificada, é colocada uma série de questões extra: “quando”? “em que situação”? “que correcções devem ser efectuadas”?
  - O objectivo de uma análise HAZOP é o de avaliar prioridades, por forma a identificar as áreas críticas que justificam investigação suplementar.

Estudos de Operabilidade e de Situações Perigosas - HAZOP

- **Exemplo**

[Storey, 96]

Guide word	*Chemical plant*	*Computer-based system*
No	No part of the intended result is achieved	No data or control signal exchanged
More	A quantitative increase in the physical quantity	A signal magnitude or a data rate is too high
Less	A quantitative decrease in the physical quantity	A signal magnitude or a data rate is too low
As well as	The intended activity occurs, but with additional results	Redundant data sent in addition to intended value
Part of	Only part of the intended activity occurs	Incomplete data transmitted
Reverse	The opposite of what was intended occurs, for example reverse flow within a pipe	Polarity of magnitude changes reversed
Other than	No part of the intended activity occurs, and something else happens instead	Data complete but incorrect

Estudos de Operabilidade e de Situações Perigosas - HAZOP

- **Exemplo [Storey, 96]**

Attribute	*Guide word*	*Possible meaning*
Data flow	More	More data is passed than expected
	Less	Less data is passed than expected
Data rate	More	The data rate is too high
	Less	The data rate is too low
Data value	More	The data value is too high
	Less	The data value is too low
Repetition time	More	The time between output updates is too high
	Less	The time between output updates is too low
Response time	More	The response time is longer than required
	Less	The response time is shorter than required

Estudos de Operabilidade e de Situações Perigosas - HAZOP

- **Exemplo**

[Storey, 96]

Item	Inter-connection	Attribute	Guide word	Cause	Consequence	Recommendation
1	Sensor supply line	Supply voltage	No	PSU, regulator or cable fault	Lack of sensor signal detected and system shuts down	
2			More	Regulator fault	Possible damage to sensor	Consider overvoltage protection
3			Less	PSU or regulator fault	Incorrect temperature reading	Include voltage monitoring
4		Sensor current	More	Sensor fault	Incorrect temperature reading, possible loading of supply	Monitor supply current
5			Less	Sensor fault	Incorrect temperature reading	As above
6	Sensor output	Voltage	No	PSU, sensor or cable fault	Lack of sensor signal detected and system shuts down	
7			More	Sensor fault	Temperature reading too high – results in decrease in plant efficiency	Consider use of duplicate sensor
8			Less	Sensor mounted incorrectly or sensor failure	Temperature reading too low – could result in overheating and possible plant failure	As above

Análise por Árvore de Falhas - FTA

- **Metodologia FTA**
  - Metodologia gráfica: construção de uma árvore a partir de cada Situação Perigosa identificada (“Top Event”) e análise das suas possíveis causas;
    - » Caso exista informação anterior fidedigna sobre o funcionamento do sistema, podem ser utilizados como origem da árvore Acidentes (ou Incidentes) anteriores;
    - » Alternativamente, a informação de entrada pode ser obtida através de uma análise FMEA ou HAZOP.

Análise por Árvore de Falhas - FTA

- **Vantagens**
  - Particularmente eficaz para demonstrar os efeitos de variações de parâmetros ou de valores “fora da escala” sobre a segurança do sistema.
  - O facto de se analisarem unicamente cadeias de acontecimentos que garantidamente levam à ocorrência de situações perigosas, reduz o espaço de análise necessário.

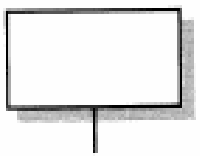
Análise por Árvore de Falhas - FTA

- **Notação específica**
  - Uma falha primária no sistema ocorre quando, em condições normais (segundo a especificação) de funcionamento, um seu componente avaria;
  - Uma falha secundária no sistema ocorre quando um seu componente avaria devido a terem sido excedidas as suas condições normais de funcionamento;
  - Uma falha de comando ocorre quando um componente reage (responde) em circunstâncias inesperadas.

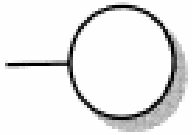
Análise por Árvore de Falhas - FTA

- **Notação**

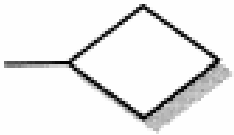
[Storey, 96]



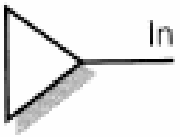
Fault event resulting from other events

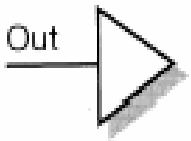


Basic event, taken as an input



Fault event not fully traced to its source. It is taken as an input but its causes may be unknown



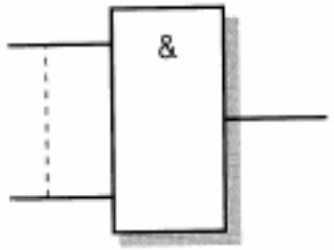


The triangle symbol is used to link trees. The 'in' symbol indicates an input from another tree (on another sheet). The 'out' symbol appears in place of the 'top event' and indicates that this point forms the input to another tree

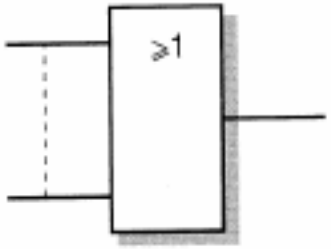
Análise por Árvore de Falhas - FTA

- **Notação**

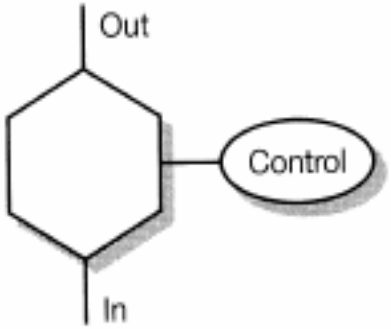
[Storey, 96]



The output event occurs if ALL the inputs occur



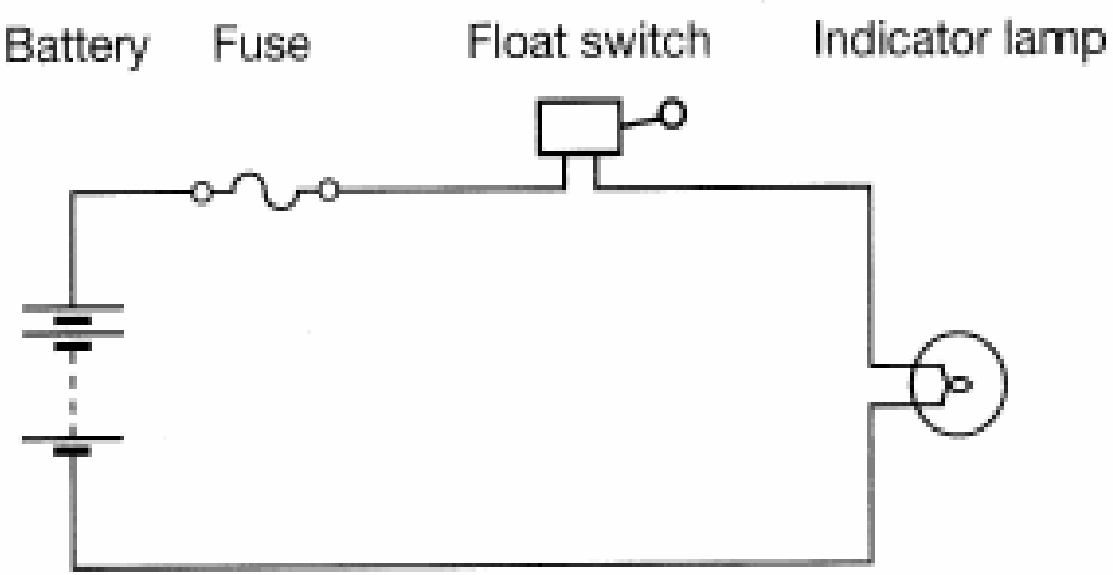
The output event occurs if ANY of the inputs occur, either alone or in combination



The control condition determines whether the input event appears at the output

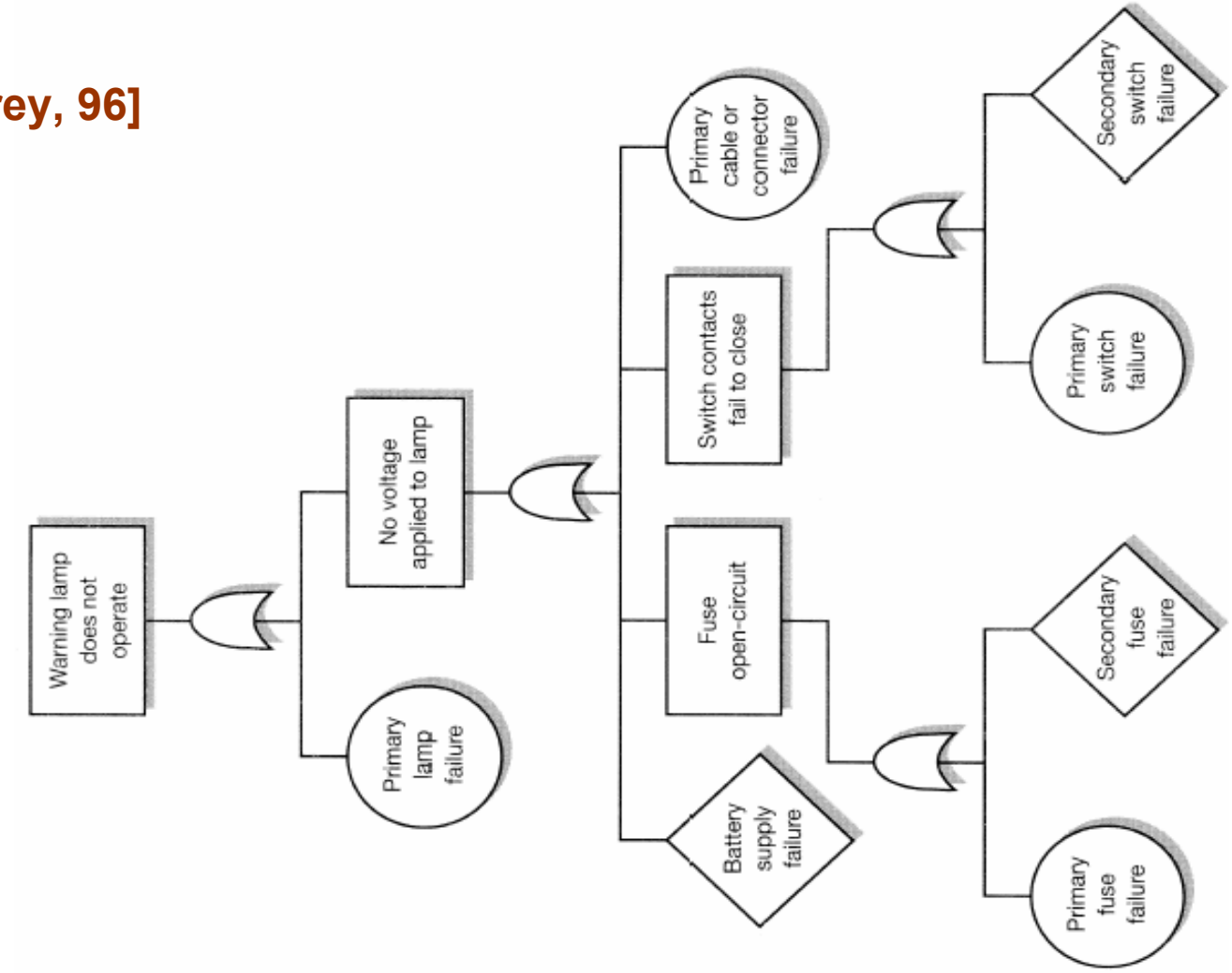
Análise por Árvore de Falhas - FTA

- **Exemplo [Storey, 96]**
  - » "Top event": lâmpada não indica "nível baixo de óleo"



Análise por Árvore de Falhas - FTA

- **Exemplo [Storey, 96]**



Sistemas de Segurança Crítica

- **Introdução**
- **Análise da Ocorrência de Situações Perigosas ("*Hazard Analysis*")**
- **Análise do Risco ("*Risk Analysis*")**
  - Considerações sobre o Risco
  - Considerações sobre a classificação do Risco
  - Considerações sobre a tolerabilidade do Risco
  - Níveis de Integridade da Segurança
- **Tolerância a Falhas**

Considerações sobre o Risco

- **Definições**
  - Um Acidente é um evento ou uma sequência de eventos que tem como consequência a morte ou o ferimento de pessoas, ou prejuízos ambientais ou materiais.
  - Um Incidente ("*Near Miss*") é um evento ou uma sequência de eventos inesperado(a) que não resulta em perdas, mas que noutras circunstâncias tem esse potencial.

Considerações sobre o Risco

- **Avaliação Quantitativa**
  - A relevância de uma determinada Situação Perigosa está relacionada com os Acidentes que daí podem resultar.
  - Dois factores devem ser considerados:
    - » as potenciais consequências de um acidente resultante;
    - » a frequência (ou probabilidade) de ocorrência do referido acidente.

Considerações sobre o Risco

- **Avaliação Quantitativa**
  - Risco ("*Risk*") é a combinação da probabilidade de ocorrência de uma situação perigosa específica, e das suas consequências.
  - Análise do risco ("*Risk Analysis*"):
    - » Análise quantitativa do risco (probabilidade) associado a uma cadeia de acontecimentos na origem de uma situação perigosa específica.
  - O resultado do processo de análise do Risco é a sua classificação numa de várias classes, em função da criticalidade e da probabilidade de ocorrência de cada situação perigosa específica

Sobre a Classificação do Risco [IEC 61508]

Risk classification of accidents

Frequency	Consequence			
	Catastrophic	Critical	Marginal	Negligible
Frequent	I	I	I	II
Probable	I	I	II	III
Occasional	I	II	III	III
Remote	II	III	III	IV
Improbable	III	III	IV	IV
Incredible	IV	IV	IV	IV

- Classe I - Risco intolerável
- Classes II e III - Risco ALARP
- Classe IV - Risco tolerável

Sobre a Classificação do Risco [IEC 61508]

Interpretation of risk classes

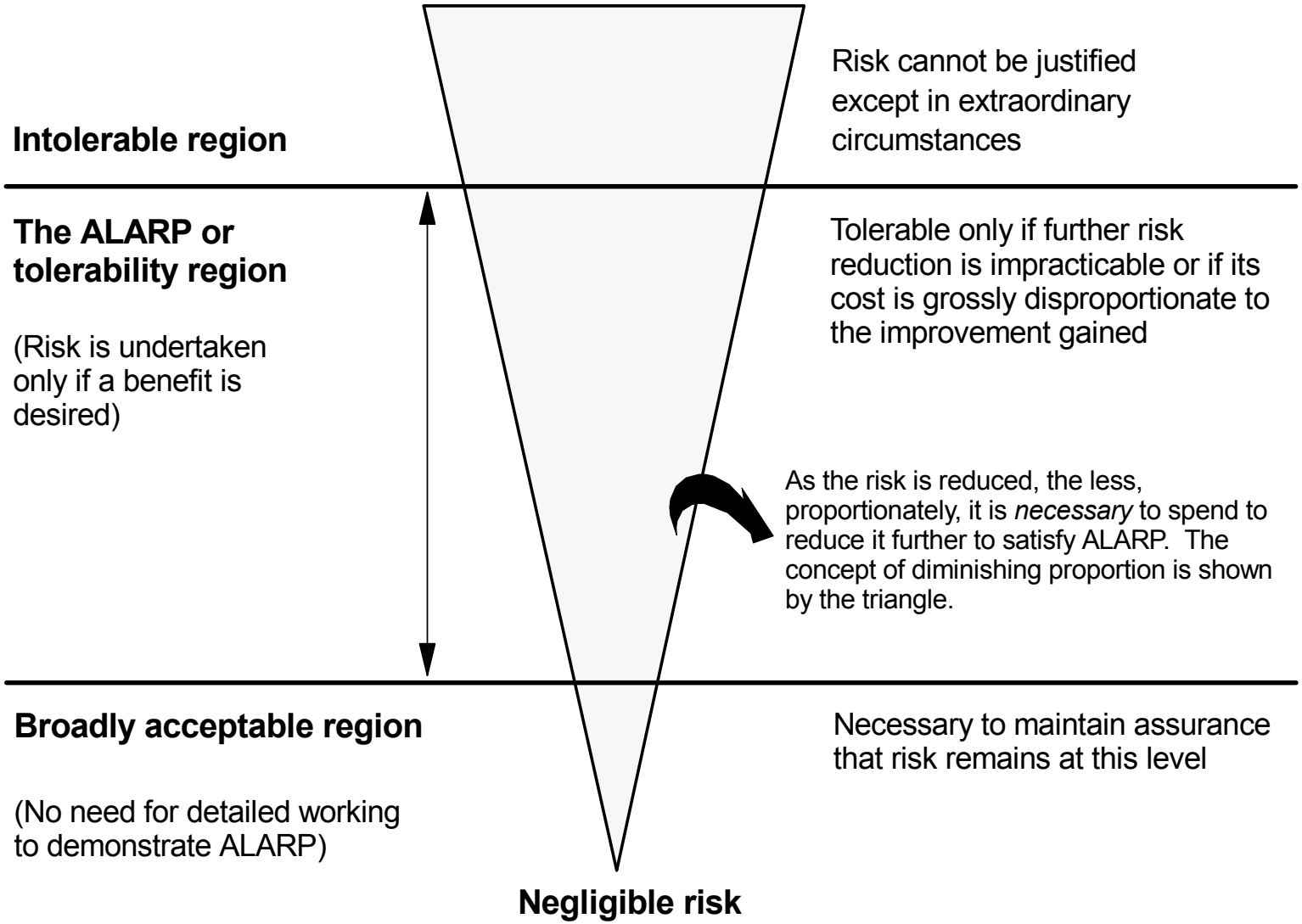
Risk class	Interpretation
Class I	Intolerable risk
Class II	Undesirable risk, and tolerable only if risk reduction is impracticable or if the costs are grossly disproportionate to the improvement gained
Class III	Tolerable risk if the cost of risk reduction would exceed the improvement gained
Class IV	Negligible risk

- Classe I - Risco intolerável
- Classes II e III - Risco ALARP
- Classe IV - Risco tolerável

Sobre a tolerabilidade do Risco [IEC 61508]

- Risco inaceitável, que nunca poderá ser justificável excepto em circunstâncias excepcionais;
- Risco aceitável, quando pode ser negligenciado.
- Risco tolerável - ALARP ("*As Low As Reasonably Practicable*"), quando o custo da sua redução ultrapassa largamente o benefício decorrente dessa redução;
  - » Um Risco na gama ALARP, caso seja de fácil redução nunca poderá ser considerado tolerável.
  - » Em sistemas de Integridade Elevada, cabe à entidade de certificação determinar se um determinado Risco terá ou não que ser reduzido.

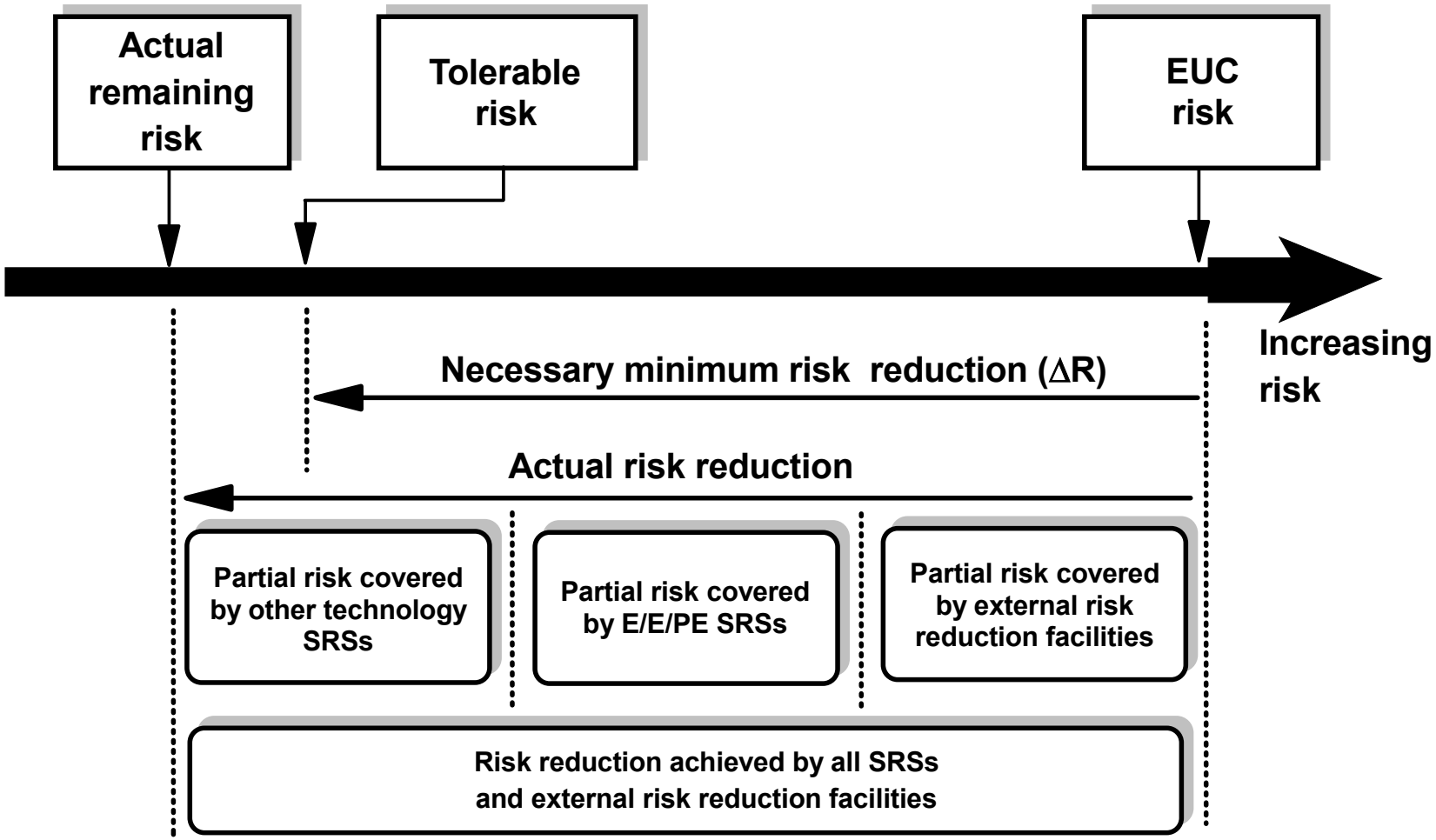
Sobre a tolerabilidade do Risco [IEC 61508]



Sobre a tolerabilidade do Risco [IEC 61508]

- **Gestão do Risco**
  - A gestão do processo de desenvolvimento de um Sistema de Segurança Crítica pode ser vista como a gestão de um processo de Redução de Risco.
  - Terminologia
    - » EUC - "*Equipment Under Control*"
    - » SRS - "*Safety Related System*"
    - » ΔR - "Risk Reduction"

Sobre a tolerabilidade do Risco [IEC 61508]



Níveis de Integridade da Segurança

- **Níveis de Integridade da Segurança**
  - Relacionados com o nível de Redução de Risco exigível:
    - » Caso seja necessária uma elevada Redução do Risco (potencial de Risco elevado), os mecanismos de redução do risco devem ser de elevado nível de confiança.
    - » Caso a necessidade de Redução de Risco seja baixa, esses mecanismos poderão ser mais simples.

Níveis de Integridade da Segurança

- **Níveis de Integridade da Segurança**
  - Diferentes tipos de requisitos de Segurança, levaram à definição de Níveis de Integridade da Segurança (SIL).
- **Definição**
  - *"Safety Integrity is the probability of a safety-related system satisfactorily performing the required safety functions, under all the stated conditions, within a stated period of time."* [IEC 61508]

Níveis de Integridade da Segurança

- **Níveis de Integridade da Segurança**
  - Apesar de ser possível exprimir quantitativamente a Integridade da Segurança, é prática habitual atribuir a cada sistema relacionado com a Segurança um Nível de Integridade da Segurança (de entre 4 níveis).
  - Estes vários níveis têm associados requisitos numéricos, tais como as gamas de "avarias por ano", ou "probabilidade de avaria durante um acidente".

Níveis de Integridade da Segurança

- **Exemplo [IEC 61508]**

Safety integrity level	**Low demand mode of operation** (Average probability of failure to performs its design function on demand)
4	$\geq 10^{-5}$ to $< 10^{-4}$
3	$\geq 10^{-4}$ to $< 10^{-3}$
2	$\geq 10^{-3}$ to $< 10^{-2}$
1	$\geq 10^{-2}$ to $< 10^{-1}$

– Para sistemas a operarem em "*demand mode operation*", a medida da integridade da Segurança relevante é a probabilidade de não conformidade com a especificação (avaria) da função, quando esta é pedida (ex: relevante para "*shut-down systems*").

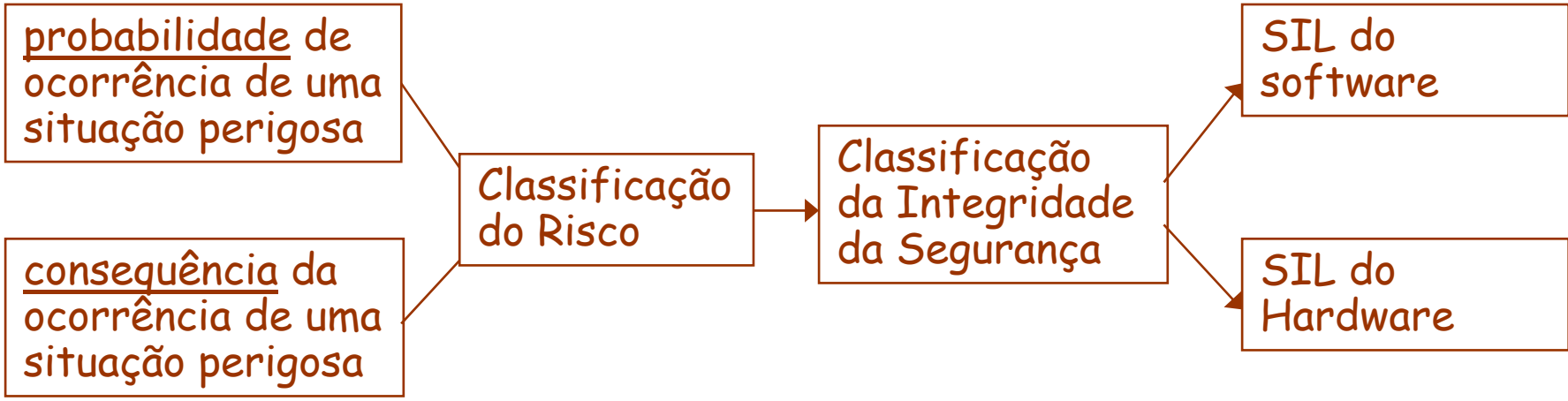
Níveis de Integridade da Segurança

- **Exemplo [IEC 61508]**

Safety integrity level	**High demand/continuous mode of operation** (Probability of a dangerous failure per hour)
4	$\geq 10^{-9}$ to $< 10^{-8}$
3	$\geq 10^{-8}$ to $< 10^{-7}$
2	$\geq 10^{-7}$ to $< 10^{-6}$
1	$\geq 10^{-6}$ to $< 10^{-5}$

  - Para sistemas a operarem em "*continuous / high demand mode operation*", a medida da integridade da Segurança relevante é a probabilidade de avaria por hora (1 ano = 8760 horas).

Níveis de Integridade da Segurança



- Um Nível da Integridade da Segurança (SIL) é atríbuido ao software para definir a importância da exactidão dos módulos de software.
- O nível seleccionado determina os métodos de desenvolv. e de teste a utilizar ao longo do ciclo de vida.

Níveis de Integridade da Segurança

- **A atribuição dos SIL a um sistema está relacionada com o nível de Risco do sistema.**
  - Não esquecer que:
    - » Risco é a medida combinada da probabilidade de ocorrência de uma situação perigosa e das suas consequências;
    - » Integridade da Segurança é a medida da probabilidade de um sistema de Segurança desempenhar correctamente as suas tarefas, para as condições e durante um período de tempo especificados.

Níveis de Integridade da Segurança

- **Atribuição dos SIL**
  - Os vários standards de Segurança definem gamas de taxas de avaria alvo para cada Nível de Integridade da Segurança.
  - A atribuição dos SIL é efectuada em função dessas taxas de avaria alvo (definidas para cada nível), e das taxas de avaria máxima toleráveis resultantes da Análise do Risco.

Níveis de Integridade da Segurança

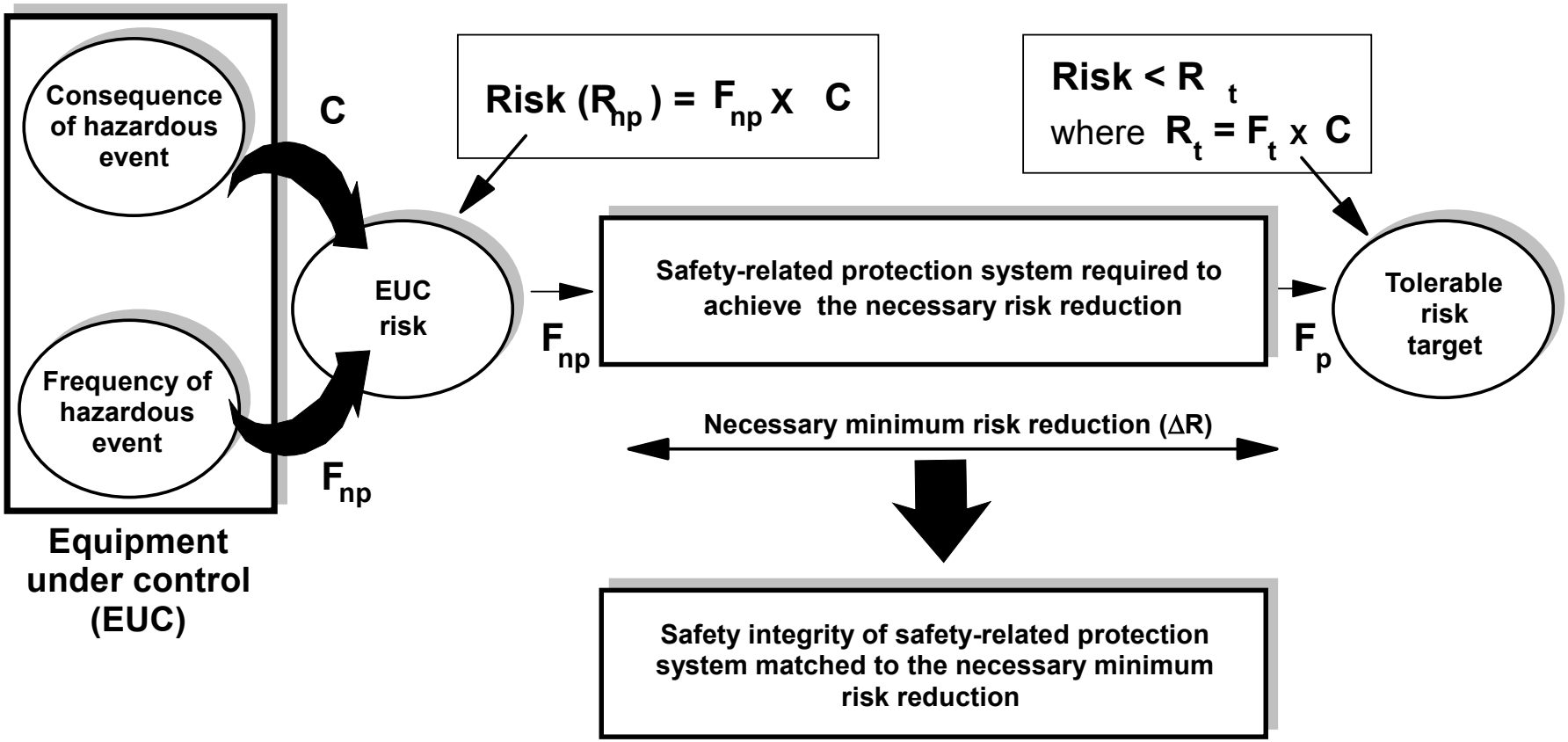
- **Determinação Quantitativa dos SIL [IEC 61508]**
  - 1. Determinar o Risco Tolerável (em termos numéricos)
    - » Ex.: uma determinada consequência especificada não deve ocorrer com uma frequência superior a uma vez em 10^4 anos (F_t)
  - 2. Determinar o Risco do equipamento sob Controlo (EUC)
    - » Determinar a frequência (F_{np}) associada ao Risco do EUC através da análise de taxas de avaria para situações similares, ou através de métodos de previsão adequados.
    - » Determinar a consequência da ocorrência da situação perigosa em análise (C)

Níveis de Integridade da Segurança

- **Determinação dos SIL [IEC 61508]**
  - 3. Verificar se é necessária efectuar uma Redução de Risco (ΔR) suplementar.
  - 4. Caso seja necessário efectuar essa redução, determinar a probabilidade máxima de avaria "*low demand*" requerida para o sistema de protecção: $PFD_{avg}=(F_t/F_{np})=\Delta R$
    - » Considerando que a consequência da ocorrência da situação perigosa (*C*) em análise se mantém constante
  - 5. O SIL pode ser obtido através da utilização da tabela adequada.

Níveis de Integridade da Segurança

- **Determinação dos SIL [IEC 61508]**



Níveis de Integridade da Segurança

- **Atribuição do SIL**
  - Após ter sido atribuído o SIL a um sistema, a sua concepção e o seu método de desenvolvimento devem garantir que o SIL definido é realizado.
  - Os standards relacionados com a Segurança garantem, para cada SIL, um conjunto de procedimentos que responde à questão anterior.

Sistemas de Segurança Crítica

- **Introdução**
- **Análise da Ocorrência de Situações Perigosas ("*Hazard Analysis*")**
- **Análise do Risco ("*Risk Analysis*")**
- **Tolerância a Falhas**
  - Falhas, Erros e Avarias;
  - Modelos de Falhas;
  - Conceitos Básicos de Tolerância a Falhas;
  - Técnicas para Tolerância a Falhas (Hw/Sw).

Falhas, Erros e Avarias

- **Confiança no funcionamento** (“***Dependability***”)
  - Propriedade que permite que um utilizador de um sistema computacional possa depositar uma confiança justificada no serviço que ele presta.
- **Impedimentos à Confiança no Funcionamento**
  - A Avaria de um sistema ocorre quando o serviço prestado deixa de estar conforme a especificação;
  - Erro é um estado do sistema que pode levar a uma avaria.
  - A causa hipotética de um erro é uma Falha.

Falhas, Erros e Avarias

Falhas, Erros e Avarias

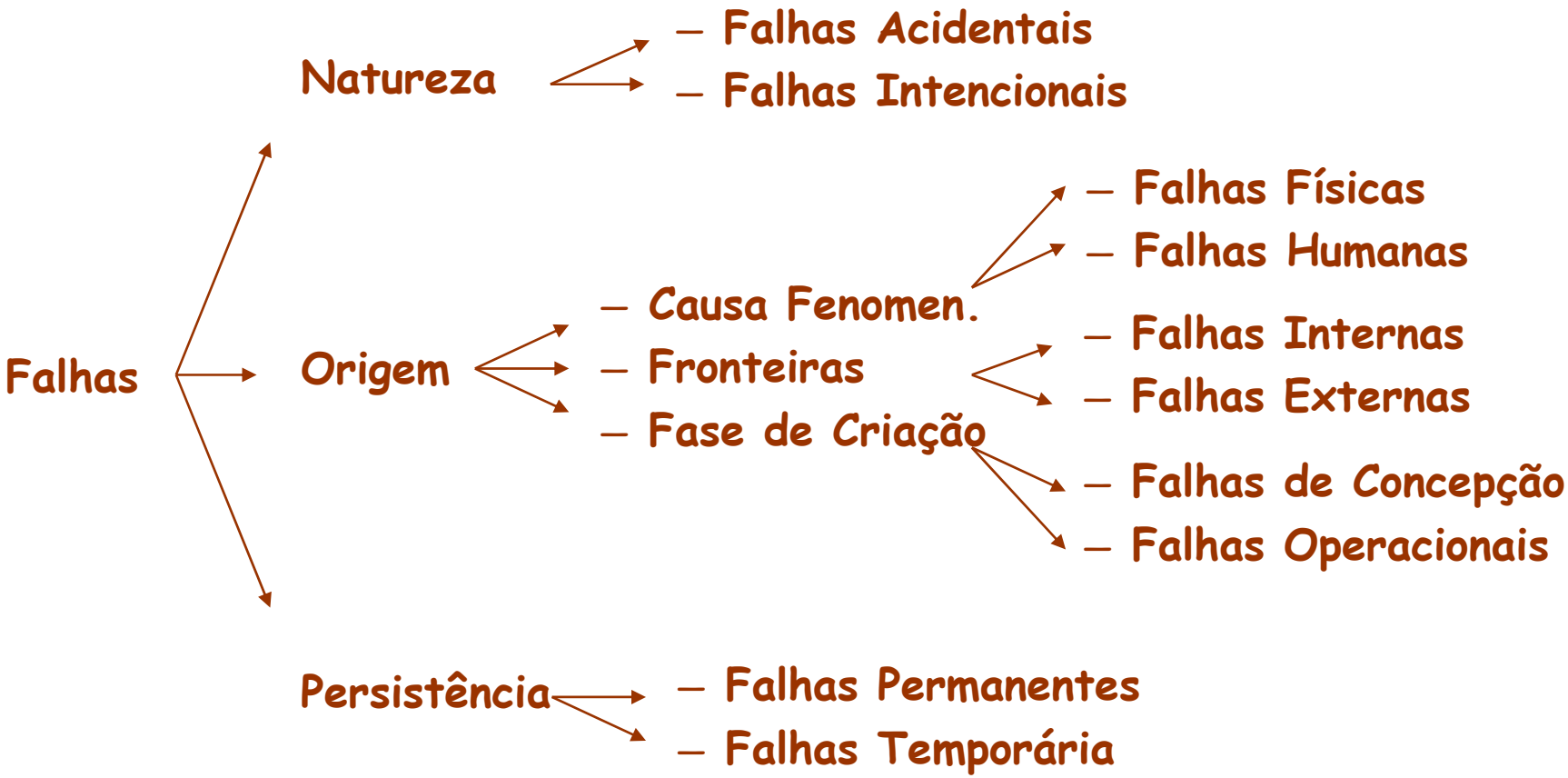
- **Cadeia "Falha → Erro → Avaria"**
  - Uma falha torna-se activa quando produz um erro. Uma falha activa poderá ser:
    - » uma falha interna que estava previamente dormente e que foi activada pelo processo computacional;
    - » uma falha externa.
  - Um erro está latente quando ainda não foi reconhecido com tal; um erro pode ser detectado por um mecanismo ou algoritmo de detecção.
  - Um erro pode propagar-se, o que geralmente acontece; ao propagar-se, um erro cria outro novo erro(s).
  - Uma avaria ocorre quando um erro "passa pela" interface utilizador-sistema e afecta o serviço prestado pelo sistema.

Falhas, Erros e Avarias

- **Transições de estado "Falha → Erro → Avaria"**

Falhas, Erros e Avarias

- **Classes de Falhas**



- **6. Tolerância a Falhas**
  - Falhas, Erros e Avarias;
  - Modelos de Falhas;
  - Conceitos Básicos de Tolerância a Falhas;
  - Técnicas para Tolerância a Falhas (Hw/Sw);

Modelos de Falhas

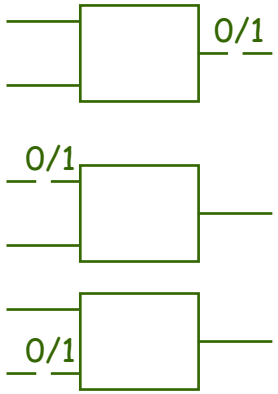
- **Modelos de Falhas:**
  - Um modelo de falhas pretende determinar quais os possíveis efeitos das falhas sobre o comportamento do sistema em consideração;
  - A simulação da ocorrência de falhas é fundamental para a concepção de componentes (hw/sw) tolerantes a falhas.

Modelo de Falhas "Single Stuck-at"

- **Pressupostos.**
  - Função "vista" como caixa-preta;
  - Falha modelada como:
    - » erro na entrada ou na saída;
    - » entrada ou saída "colada" a 1 ou a 0.
  - Modelo válido para o caso de falhas permanentes.

- Caso sem falhas:

- Casos com falhas:

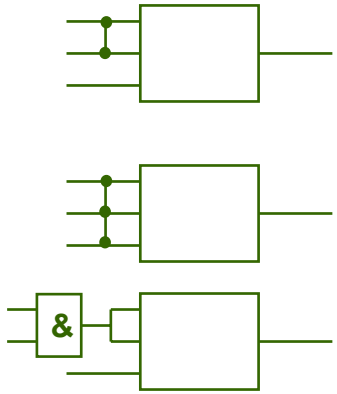


Modelo de Falhas "Curto-Circuito"

- **Pressupostos.**
  - Função "vista" como caixa-preta;
  - Falha modelada como:
    - » Curto-circuito acidental entre dois ou mais nós do circuito.
  - Modelo válido para o caso de falhas permanentes.

- Caso sem falhas:

- Casos com falhas:



Modelos de Falhas em Software

- **Modelos de Falhas em Software**
  - O teste de mutação inclui uma abordagem idêntica, visto que se supõe que as falhas são simples modificações no código do programa (outro tipo de falhas tem que ser também analisado).
- **"*Bug*" – Falha no Software**
  - O software não avaria intermitentemente. Todas as avarias são sistemáticas, e têm como origem a fase de concepção.

Modelos de Falhas em Sw.

- **"*Bug*" – Falha no Software**
  - Falhas típicas em sistemas de Software:
    - » Falha na especificação;
    - » Falha na codificação;
    - » Erros lógicos nos cálculos;
    - » "*Overflows*" / "*Underflows*";
    - » Utilização de variáveis não inicializadas.

Modelos de Falhas em Software

- **Problemas a considerar:**
  - A modificação de uma variável pode afectar a exactidão de qualquer rotina de um componente que aceda a essa variável (problema parcialmente resolvido por encapsulamento);
  - A corrupção do código-máquina pode resultar num funcionamento arbitrário de qualquer componente;

Falhas em Módulos de Software

- **Falhas de componentes:**
  - » Falha por omissão persistente ("*crash*"); pressupõe um sistema silencioso em caso de avaria ("*fail-silent*");
  - » Funcionamento corrompido durante um intervalo de tempo limitado superiormente (resolúvel com utilização de "*watchdog*");
  - » Funcionamento corrompido durante um intervalo de tempo não limitado.

Falhas em Módulos de Software

- **Falhas de comunicação:**
  - Re-ordenamento de mensagens;
  - Perda de mensagens;
  - Mensagens corrompidas;
  - Mensagens repetidas;
  - Mensagens arbitrariamente modificadas (falha bizantina).

Modelos de Falhas: Discussão

- **Modelo de falhas grosseiro**
  - (+) Mecanismos de tolerância a falhas simplificados;
  - (+) Algoritmos de teste mais eficientes;
  - (-) Falhas potencialmente relevantes não consideradas.
- **Modelo de falhas detalhado:**
  - (+) Maior relevância na cobertura do modelo;
  - (-) Mecanismos de tolerância a falhas mais custosos.

Modelos de Falhas: Discussão

- **Cobertura do Modelo de Falhas**
  - 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.

- **6. Tolerância a Falhas**
  - Falhas, Erros e Avarias;
  - Modelos de Falhas;
  - Conceitos Básicos de Tolerância a Falhas;
  - Técnicas para Tolerância a Falhas (Hw/Sw);

Conceitos Básicos de Tolerância a Falhas

- **Dependência entre os meios para a Obtenção e a Validação da Confiança no Funcionamento:**
  - » prevenção de falhas;
  - » tolerância a falhas;
  - » supressão de falhas;
  - » previsão de falhas.

Falhas, Erros e Avarias

Conceitos Básicos de Tolerância a Falhas

- **Dependência entre os meios para obter Confiança no Funcionamento ("*Dependability*")**
  - Interdependências entre os meios para a obtenção de confiança no funcionamento:
    - » Apesar da prevenção de falhas por intermédio de metodologias de concepção adequadas (forçosamente imperfeitas, de forma a poderem ser utilizadas), aparecem falhas.

Conceitos Básicos de Tolerância a Falhas

- **Justificação para a necessidade de tolerância a falhas**
  - Limitações da prevenção de falhas:
    - » a supressão a priori de todas as falhas de um sistema é impossível;
      - embora durante o projecto se devam suprimir todas as falhas conhecidas, não se podem eliminar falhas cuja existência ainda não foi revelada;
    - » quando a prevenção de falhas não é bem sucedida, podem surgir atrasos imprevistos.

Conceitos Básicos de Tolerância a Falhas

- **Justificação para a necessidade de tolerância a falhas**
  - A supressão de falhas é, por si só, imperfeita, dado que se utilizam componentes "chave-na-mão"; Daqui decorre a importância da previsão de falhas.
  - A forte interacção entre a supressão e a previsão de falhas, motiva o seu agrupamento num só termo – Validação;
    - » Supressão de falhas: "estarei a construir o sistema correcto?"
    - » Previsão de falhas: "durante quanto tempo estará ele correcto?"
  - Daqui decorre a necessidade da Tolerância a Falhas.

Conceitos Básicos de Tolerância a Falhas

- **Etapas da tolerância a falhas:**
  - 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.

Conceitos Básicos de Tolerância a Falhas

- **Processamento de Erros**
  - Detecção de erros:
    - » permite que o estado erróneo seja identificado como tal;
    - » Quando se recorre à recuperação de erros, o estado erróneo necessita de ser identificado com sendo erróneo, antes de ser substituído.
  - Diagnóstico de erros:
    - » 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 (para trás, para a frente, compensação).

Conceitos Básicos de Tolerância a Falhas

- **Recuperação de erros:**
  - Recuperação para trás
    - » Consiste em fazer o sistema voltar a um estado pelo qual o sistema já passou anteriormente, antes da ocorrência do erro;
    - » Envolve a definição de pontos de recuperação, para os quais o estado do processo pode posteriormente ser restaurado.
  - Recuperação para a frente
    - » Consiste em procurar um novo estado a partir do qual o sistema possa funcionar (frequentemente em modo degradado).
  - Compensação
    - » Onde o estado erróneo contém redundância suficiente para permitir que o serviço(s) prestado a partir de um estado erróneo esteja contudo isento de erros.

Conceitos Básicos de Tolerância a Falhas

- **Recuperação para trás / para a frente (discussão)**
  - » A recuperação para trás pode ser tentada primeiro; se o erro persistir, tenta-se a recuperação para a frente.
  - » Na recuperação para a frente é necessário avaliar os danos causados pelo erro detectado.
  - » Na recuperação para trás, a avaliação dos danos pode ser ignorada, desde que os mecanismos que permitem a substituição do estado erróneo num estado isento de erros não tenham sido afectados.

Conceitos Básicos de Tolerância a Falhas

- **Compensação de erros (discussão)**
  - Pode ser aplicada sistematicamente, mesmo na ausência de erros, proporcionando assim uma dissimulação de falhas
    - » Por exemplo, voto majoritário;
  - Pode levar a uma perda desconhecida de redundância:
    - » Por exemplo, na redundância modular tripla, se um dos componentes redundantes falhar, o sistema não avaria (pois os outros dois são majoritários), mas perde redundância.

Conceitos Básicos de Tolerância a Falhas

- **Tratamento de Falhas**
  - Primeiro passo ⇨ Diagnóstico das falhas
    - » determinação da causa(s) do erro(s), tanto em termos de localização como em termos de natureza.
  - Segundo passo ⇨ Desactivação de falhas
    - » Prevenir que a(s) falha(s) sejam de novo activadas, tornando-a(s) passiva(s).
    - » Por exemplo, removendo o componente considerado como defeituoso de execuções seguintes.

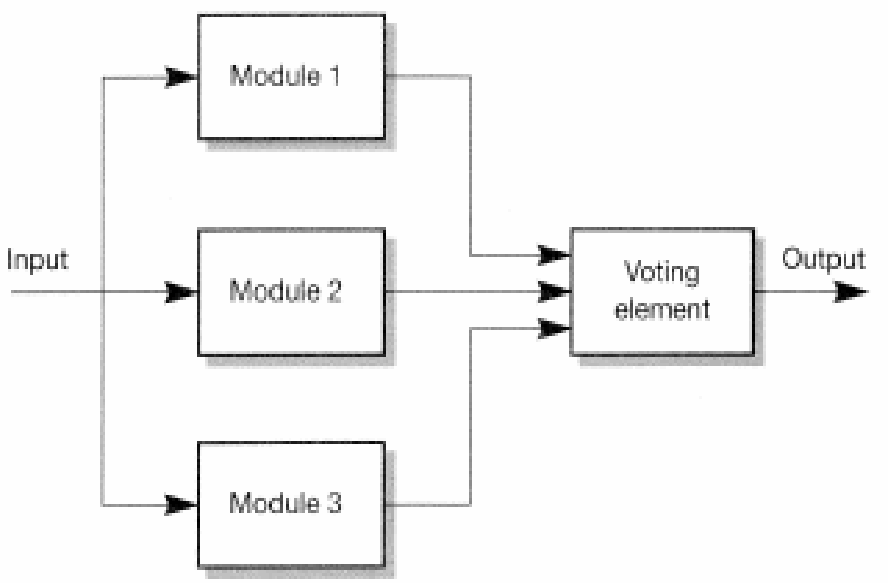
Conceitos Básicos de Tolerância a Falhas

- **Cobertura da Tolerância a Falhas**
  - As classes de falhas que podem ser toleradas dependem das hipóteses de falhas que são consideradas na fase de concepção.
    - » Um exemplo poderá ser considerar a tolerância a falhas físicas e a tolerância a falhas de concepção.
  - Esta Cobertura nunca é total devido 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);
    - » hipóteses de falhas que diferem das falhas que realmente acontecem durante a operação do sistema (falta de cobertura dos modos de falha).

- **6. Tolerância a Falhas**
  - Falhas, Erros e Avarias;
  - Modelos de Falhas;
  - Conceitos Básicos de Tolerância a Falhas;
  - Técnicas para Tolerância a Falhas (Hw/Sw);
    - » Redundância;
    - » Técnicas de Detecção de Falhas;
    - » Tolerância a Falhas em Hardware;
    - » Tolerância a Falhas em Software.

Redundância

- **Redundância de Hardware**
  - Utilização de módulos de hardware suplementares;
  - Exemplo: Arquitectura TMR (“Triple Modular Redundant”).



Redundância

- **Redundância de Software**
  - Utilização de módulos de software suplementares, para além dos que seriam necessários no caso de um funcionamento sem falhas.
- **Redundância de Informação**
  - Utilização de informação suplementar, destinada a detectar ou tolerar falhas;
    - » Exemplo, códigos de CRC, bits de paridade, “checksums”.

Redundância

- **Redundância Temporal**
  - Utilização de “tempo”, para além do requerido para implementar uma dada função, destinada a detectar ou tolerar falhas;
  - Pode ser implementada sob a forma de:
    - » repetição de cálculos e comparação de resultados;
    - » repetição de envio de mensagens e comparação de valores.

Redundância

- **Concepção Diversificada:**
  - A arquitectura TMR não é adequada à tolerância a falhas de concepção;
  - As diferentes vias têm de providenciar serviços idênticos através de concepções separadas, i.e., através de uma concepção diversificada (“design diversity”);
  - Uma concepção diversificada tem pelo menos duas variantes e um votador, que monitoriza os resultados da execução das variantes;

Redundância

- **Concepção Diversificada:**
  - No âmbito da tolerância a falhas de software, as variantes são também denominadas de “alternativas” (“alternates”), “réplicas” (“replicas”) ou de “versões” (“versions”).
  - Uma limitação à concepção diversificada são as inevitáveis correlações entre as variantes de software resultantes das diversas concepções.

Técnicas de Detecção de Falhas

- **Verificação de Funcionalidade**
  - Utilização de rotinas para verificar se o hardware está a funcionar correctamente.
    - » Teste de escrita e leitura na memória;
    - » Execução de sequências de teste e comparação dos resultados com os valores esperados;
    - » Testes de comunicação.

Técnicas de Detecção de Falhas

- **Verificação de Coerência ("*Consistency*")**
  - Utiliza conhecimento intrínseco sobre a informação para verificar a sua validade.
- **Redundância de Informação**
  - Utilização de redundância de informação para detectar falhas:
    - » Verificação de paridade;
    - » códigos de CRC;
    - » etc.
- **Temporizadores Cão-de-Guarda ("*Watchdog*")**
  - Detecção de avarias-por-paragem ("*crash*");
  - Um temporizador efectua a re-inicialização do sistema, caso este não apresente actividade durante um intervalo de tempo pré-determinado.

Técnicas de Detecção de Falhas

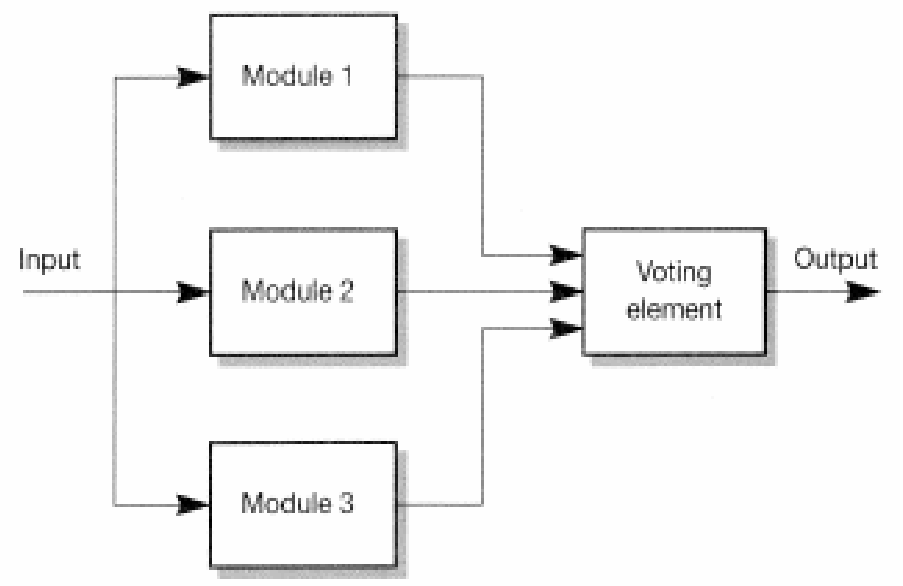
- **Detecção através de retorno ("Loopback")**
  - Verificar se o sinal à saída de um circuito a montante coincide com o sinal à entrada de um circuito a jusante.
- **Monitorização de Barramento**
  - Método eficaz para verificar a actividade de um processador, através da comparação da actividade no barramento de endereços com a gama de endereços autorizados.
- **Monitorização de Fonte de Alimentação**
  - Fundamental para o correcto funcionamento do sistema, visto que uma ligeira variação do valor da tensão de alimentação, pode provocar um comportamento erróneo do sistema.

Tolerância a Falhas em Hardware

- **1. Redundância estática**
  - Utiliza mascaramento de falhas (e não detecção) para tolerar falhas.
- **2. Redundância Dinâmica**
  - Utiliza detecção de falhas, tomando as acções necessárias para anular os seus efeitos.
- **3. Redundância Híbrida**
  - Combina os dois métodos anteriores. Utiliza mascaramento de falhas para evitar a propagação de erros no sistema, e detecção de falhas para remover os componentes avariados.

Tolerância a Falhas em Hardware

- **Redundância estática**
  - Utiliza módulos de votação para mascarar o efeito de falhas
  - Versão mais simples: TMR ("Triple Modular Redundancy").

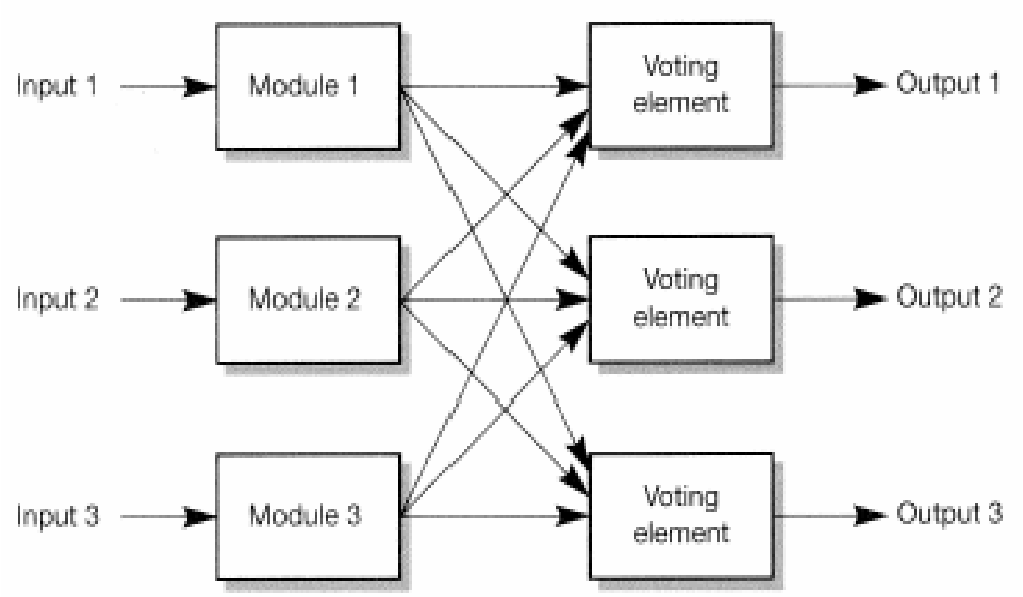


Tolerância a Falhas em Hardware

- **Redundância estática**
  - TMR (“Triple Modular Redundancy”);
  - Esta arquitectura é capaz de mascarar a falha de um dos três módulos.
  - Problema do TMR: 2 Pontos Singulares de Avaria:
    - » Sensor de entrada (é necessário a sua replicação)
    - » Elemento Votador (idem).

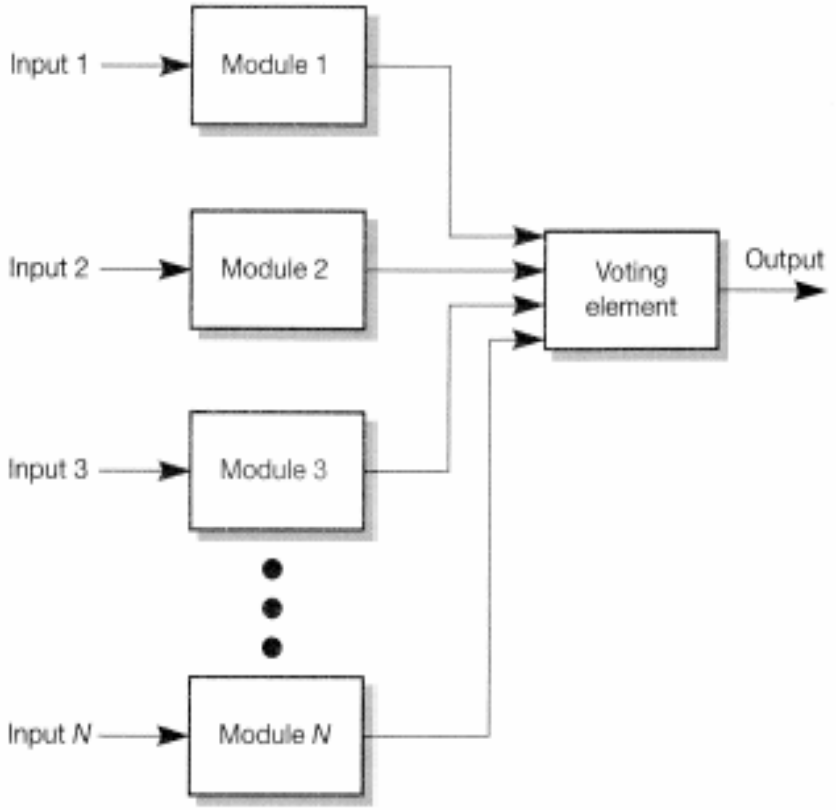
Tolerância a Falhas em Hardware

- **Redundância estática**
  - TMR com votação triplicada



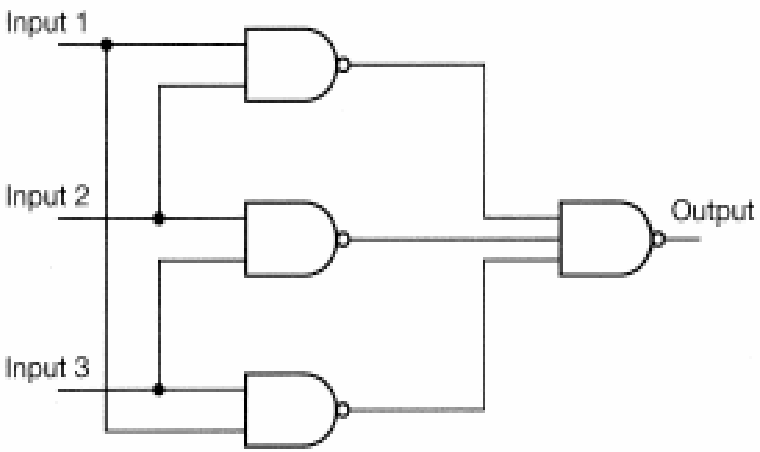
Tolerância a Falhas em Hardware

- **Redundância estática**
  - NMR: Redundância com N-módulos
  - Capaz de mascarar a falha de (N-1)/2 módulos;
  - Deverão ser utilizados múltiplos votadores.



Tolerância a Falhas em Hardware

- **Redundância estática**
  - Técnicas de Votação
    - » Exemplo: Votação em Hardware



Truth table

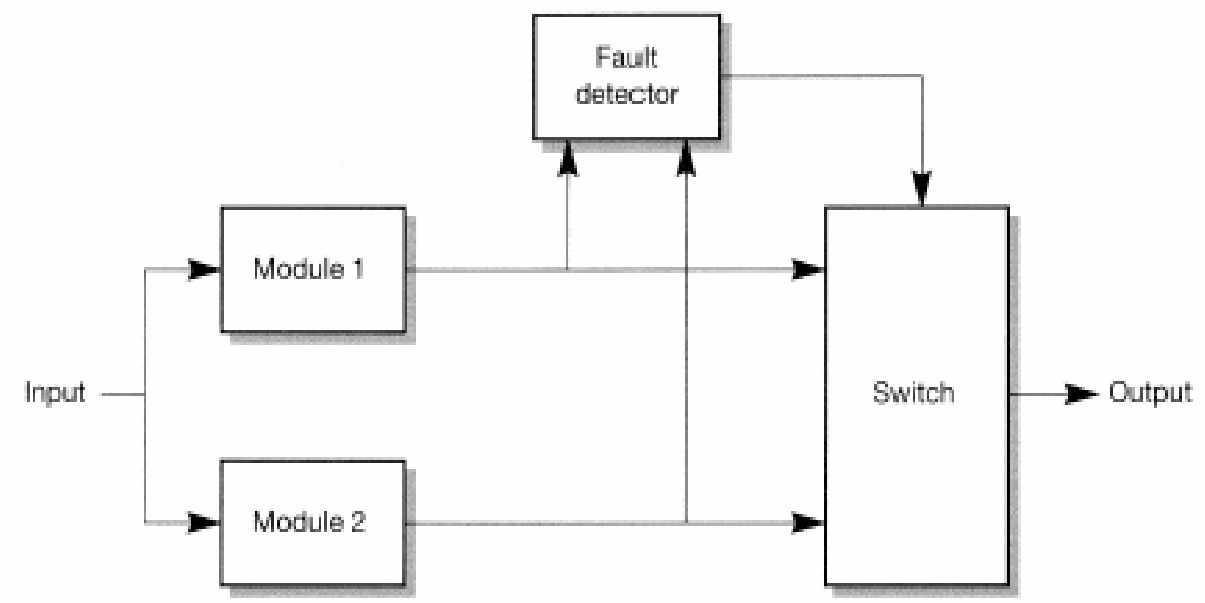
Inputs			Output
1	2	3	
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	1
1	1	1	1

Tolerância a Falhas em Hardware

- **Redundância dinâmica**
  - Utiliza detecção de falhas, tomando as acções necessárias para anular os seus efeitos;
  - Utiliza um módulo em funcionamento normal, mantendo um ou mais módulos sobresselentes que, caso seja detectada uma falha no módulo em funcionamento, tomarão o seu lugar;
  - Efectua a contenção de falhas (e não o seu mascaramento).

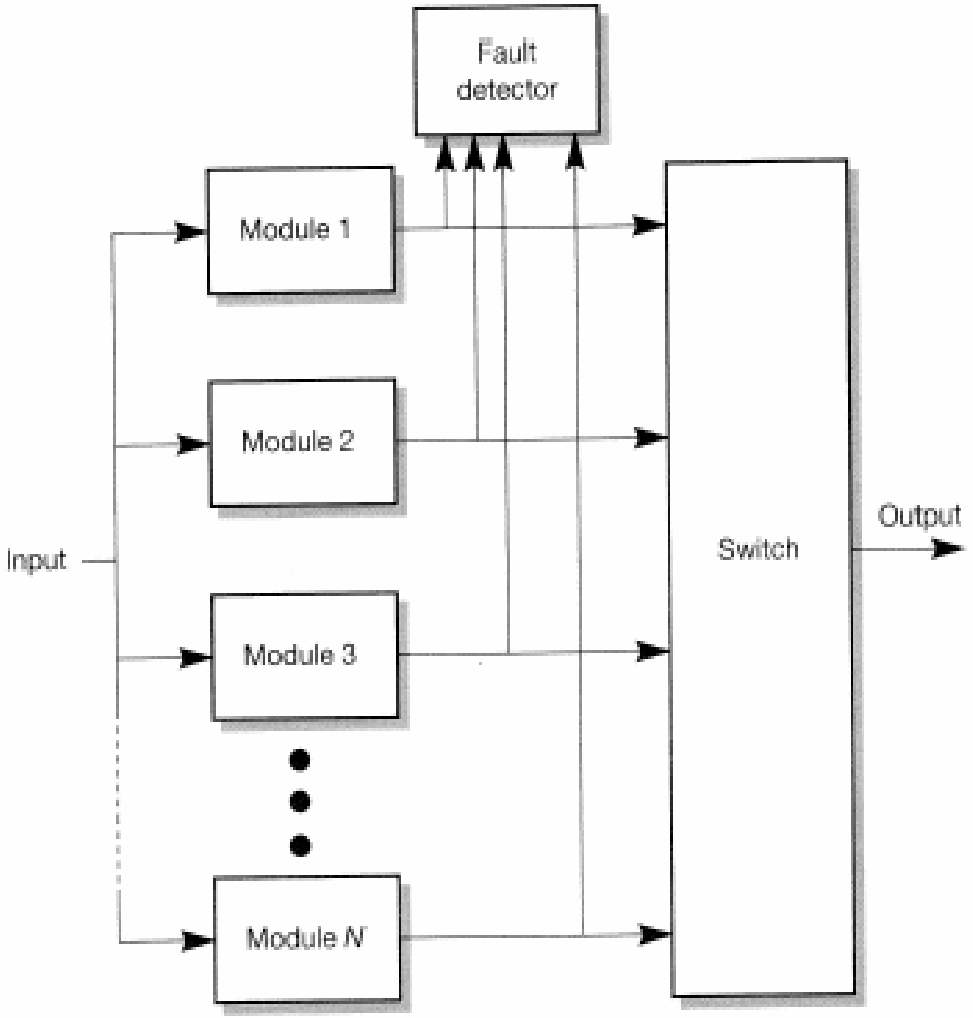
Tolerância a Falhas em Hardware

- **Redundância dinâmica**
  - "Hot Standby": o módulo sobresselente em funcionamento;
  - "Cold Standby": o módulo sobresselente inicializado caso seja detectada uma falha.



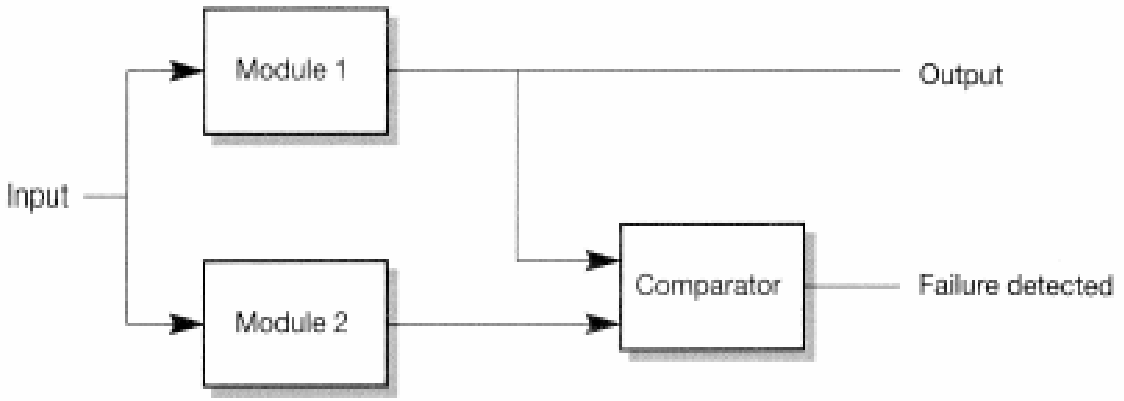
Tolerância a Falhas em Hardware

- **Redundância dinâmica**
  - Caso de N módulos



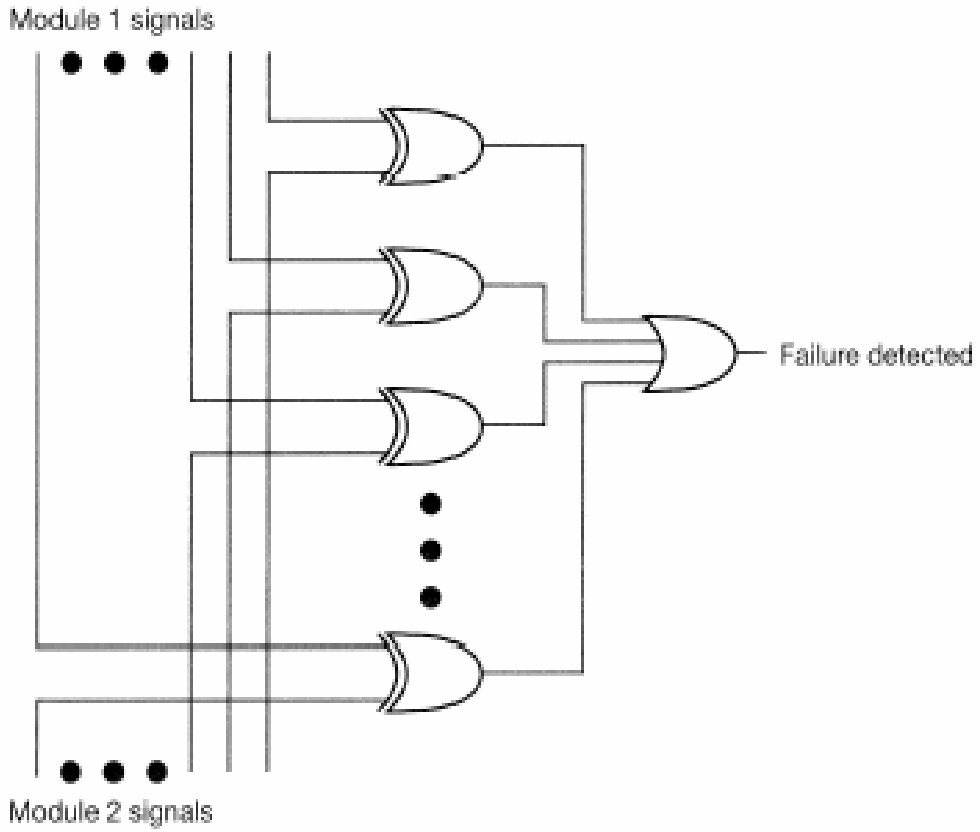
Tolerância a Falhas em Hardware

- **Redundância dinâmica**
  - "Self- Checking Pair"
  - Permite a detecção de falhas, informando o andar seguinte. Não é capaz de tolerar falhas.



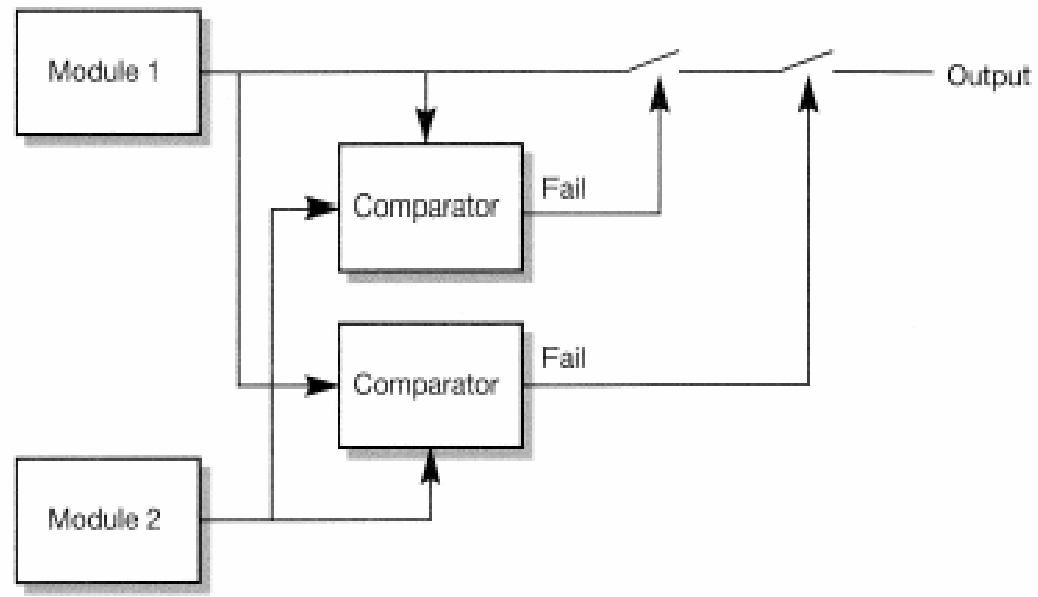
Tolerância a Falhas em Hardware

- **Redundância dinâmica**
  - "Self- Checking Pair"



Tolerância a Falhas em Hardware

- **Redundância dinâmica**
  - "Self- Checking Pair"



Tolerância a Falhas em Software

- **Recuperação por blocos**
  - Pretende fornecer uma funcionamento correcto na presença de um ou mais erros.
  - Recuperação para trás
    - » Quando um erro é detectado, o sistema é recolocado num estado seguro anterior;
    - » Implica a salvaguarda frequente do estado do sistema.
  - Recuperação para a frente
    - » Quando um erro é detectado, o sistema é forçado a evoluir para um estado posterior seguro;
    - » Método potencialmente interessante para sistemas de tempo-real.

- **Recuperação por blocos**
  - Recuperação de N-blocos
    - » Estão disponíveis N blocos (segmentos de programa) para executar a mesma função;
    - » Quando um erro é detectado durante a execução, um bloco alternativo será despoletado;
    - » Em caso de múltiplos erros, múltiplos blocos serão accionados.

Tolerância a Falhas em Software

- **Diversidade**
  - Diferentes algoritmos implementam a mesma função.
  - Os resultados da computação são comparados e, em caso de concordância ("agreement"), as acções subsequentes serão executadas;
    - Concordância por "maioria", por "unanimidade", ...
  - Em caso de não-concordância, serão desencadeados mecanismos de detecção e de recuperação de erros.

Tolerância a Falhas em Software

- **Independência**
  - Extensão ao conceito de diversidade;
  - O mesmo algoritmo é desenvolvido, implementado, verificado e validado por equipas diferentes;
  - Objectivo: diminuir a probabilidade de avarias de causa comum (CCF);

- **Encapsulamento de informação**
  - Pretende minimizar o acoplamento entre módulos e aumentar a coesão de cada módulo;
  - Objectivo: Reduzir as avarias de causa comum (CCF), minimizar o potencial para propagação de falhas e facilitar acções de manutenção.