

REQUISITOS DE SISTEMAS COMPUTACIONAIS

ESCOLA POLITÉCNICA DA UNIVERSIDADE DE SÃO PAULO

**DEPARTAMENTO DE ENGENHARIA DE COMPUTAÇÃO E SISTEMAS
DIGITAIS - PCS**

GRUPO DE ANÁLISE DE SEGURANÇA - GAS

PROF. JOÃO BATISTA CAMARGO JR.

JANEIRO - 2007

ÍNDICE

1. Apresentação	4
2. Evolução Tecnológica e Aplicações	7
3. Aplicações	9
3.1. Aplicações de Vida Longa	9
3.2. Aplicações críticas	9
3.3. Aplicações com Manutenção Adiantada/ Adiada	9
3.4. Aplicações de Alta Disponibilidade	10
4. Definições Fundamentais	11
4.1 Causas das Falhas	14
4.2. Filosofias de Projeto para Combater as Falhas	15
5. Técnicas de Projeto para Alcançar Tolerância a Defeitos	18
5.1. Redundância de Hardware	18
5.1.1. Redundância de Hardware Passiva	19
5.1.2. Redundância de Hardware Ativa	24
5.1.3. Redundância de Hardware Híbrida	29
5.2. Redundância de Informação	38
5.2.1. Código de Paridade	39
5.2.2. Códigos m de n	47
5.2.3. Códigos Duplicados	49
5.2.4. Checksums (Código Separável)	51
5.2.5. Códigos Cíclicos (Não Separável)	55
5.2.6. Códigos Aritméticos	60
5.2.7. Códigos Berger	64
5.2.8. Paridade Horizontal e Vertical	66
5.2.9. Código de Correção de Erro Hamming	66
5.2.10 Circuitos Integrados com Correção de Erro	68
5.3. Redundância por Tempo	69
5.3.1. Detecção de Defeito Transiente	70
5.3.2. Detecção do Defeito Permanente	70
5.3.2.1. Lógica Alternada	72
5.3.2.2. Recomputação com Operandos Deslocados	72
5.3.2.3. Recomputação com Operando Trocados	74
5.3.2.4. Recomputação com Duplicação e Comparação “Recomputing with Duplication with Compararison - REDWC”	76
5.3.2.5. Recomputação para Correção de Erro	77
5.4. Redundância por Software	79
5.4.1. Verificação de Consistência	79
5.4.2. Verificação de Capacidade	80
5.4.3. N_ Versões de Software	81
6. Técnicas de Avaliação de Sistemas Tolerantes a Defeito	82
6.1. Função Confiabilidade ($R(t)$) e Taxa de Falhas	82
6.2. Cálculo da Taxa de Falhas	85
6.3 Tempo Médio para Falhar – “Mean Time to Failure” – MTTF e a Confiabilidade de um Sistema	87
6.4. Tempo Médio para Reparo	90
6.5. Tempo Médio Entre Falhas	91
6.6 A Disponibilidade Assintótica de um Sistema	92
6.7. Cobertura de Defeitos	93
6.8. Modelo de Confiabilidade	96
6.8.1. Modelos Combinatórios	96
6.8.1.1. Sistema séries	97
6.8.1.2. Sistemas Paralelos	101
6.8.1.3. Cobertura de Defeito e seu Impacto na Confiabilidade	104
6.8.1.4. Sistema M de N	108
6.8.2 Modelos de Markov	110
6.8.2.1. Modelo de Markov com Cobertura de Defeitos	116

6.8.2.2. Modelo de Markov com Reparo.....	120
6.8.2.3. Modelo de Segurança.....	123
6.8.2.4 Comparação entre Sistemas.....	125
6.8.2.5. Modelo de Disponibilidade.....	127
6.8.2.6. Modelo de Manutenibilidade.....	130
7. Segurança.....	133
7.1. Aspectos Conceituais.....	133
7.2. Erro Humano.....	138
8. Metodologia de Análise de Risco Proposta.....	141
8.1. Gerenciamento da Segurança.....	142
8.2. Análise de Risco.....	143
8.3. Certificação.....	150
8.4. Perigo e Risco.....	153
8.5. Análise de Perigo.....	156
8.5.1. Métodos para Realizar Análise de Perigo.....	165
8.5.1.1. Lista de Verificação.....	165
8.5.1.2. Árvore de Falhas.....	166
8.5.1.3. Árvore de Eventos.....	168
8.5.1.4. Análise dos Efeitos dos Modos de Falhas - FMEA - Failure Modes and Effects Analysis.....	169
8.5.1.5. Análise Crítica dos Efeitos dos Modos de Falhas - FMECA - Failure Modes, Effects, and Criticality Analysis.....	170
8.5.1.6. Análise de Operação e Perigo - Hazard and Operability Analysis - HAZOP.....	170
8.5.1.7. Avaliação de Completeza de Especificações.....	173
8.5.1.8. Métodos Semi Formais.....	184
8.5.1.9. Métodos Formais.....	190
8.5.1.10. Avaliação Quantitativa da Segurança de Aplicações Microprocessadas.....	193
8.5.1.11. Injeção de Falhas.....	197
9. Referências Bibliográficas.....	198

1. Apresentação

Um sistema tolerante a defeito deve atender aos requisitos:

- Confiabilidade;
- Disponibilidade;
- Segurança;
- Desempenho;
- Dependabilidade;
- Manutenabilidade;
- Testabilidade.

Um sistema tolerante a defeito é aquele que continua desempenhado corretamente suas tarefas específicas na presença de defeito de hardware e de software.

A tolerância a defeito é um ATRIBUTO que permite ao sistema alcançar a “operação tolerante a defeito”.

Definições:

- **Confiabilidade** (“Reliability”): $R(t)$

O conceito de confiabilidade corresponde à probabilidade que um determinado sistema irá operar corretamente, ou seja, de acordo com sua especificação de requisitos, durante um determinado intervalo completo de tempo.

Desta forma, $R(t)$ é a probabilidade condicional que o sistema irá desempenhar corretamente no intervalo de tempo $[t_0, t]$, dado que o sistema estava desempenhado corretamente no instante t_0 . Trata-se de uma probabilidade condicional, pois ela depende do sistema estar operacional no início deste intervalo de tempo escolhido.

De forma complementar, pode-se definir o conceito de Não-confiabilidade

A Não-Confiabilidade (“Unreliability”) - $Q(t)$ é definida como a probabilidade que um sistema irá realizar as tarefas de maneira incorreta, ou seja, desrespeitando sua especificação de requisitos, durante um intervalo de tempo $[t_0, t]$, dado que o sistema estava desempenhando corretamente no instante t_0 . Este conceito também é conhecido como Probabilidade de Falha.

O conceito de Tolerância a Falha corresponde a uma técnica que pode aumentar a Confiabilidade de um sistema. No entanto, um sistema, com

características de tolerância a defeito, não apresenta, necessariamente, uma alta confiabilidade. Da mesma forma, pode-se afirmar que, um sistema, com alta confiabilidade, não necessariamente possui características de tolerância a defeito.

- **Disponibilidade (Availability- $A(t)$).**

O conceito de disponibilidade corresponde à probabilidade que um determinado sistema esteja operando corretamente e disponível para realizar suas funções num instante de tempo t . Assim, a disponibilidade difere da confiabilidade de forma que, a confiabilidade depende de um *INTERVALO* de tempo enquanto que a disponibilidade é avaliada em um *INSTANTE* de tempo. A disponibilidade depende então não apenas o quão freqüente um sistema torna-se inoperante, mas também o quão rápido o sistema pode ser reparado. Num caso prático, um processador reserva pode desempenhar as funções do sistema enquanto que o processador principal está sendo reparado, conservando desta forma o sistema disponível para uso final.

- **Segurança (Safety- $S(t)$).**

É a probabilidade que um sistema irá desempenhar suas funções corretamente ou descontinuar suas funções de uma forma que não interrompa a operação de outros sistemas de segurança nem comprometa a segurança de pessoas. A segurança é uma média da capacidade do sistema ser “Fail- Safe”.

Se o sistema não opera corretamente, você quer que ele falhe, pelo menos, de uma maneira segura.

- **Desempenho - (“ Performability - $P (L, t)$ ”).**

Em muitos casos é possível projetar sistemas que podem continuar a realizar as funções corretamente após falhas de HW e erros no SW, mas não no mesmo nível de desempenho.

- **Definição:**

Probabilidade que o desempenho de um sistema estará maior ou igual a um nível L , no instante t . É a média de probabilidade que um subconjunto de funções está sendo realizado corretamente.

- **Manutenabilidade (“Maintainability - $M (t)$ ”).**

É a facilidade com a qual um sistema pode ser reparado, uma vez que ele falhou.

É a probabilidade que um sistema em falha será reparado para o estado operacional dentro de um período de tempo t .

Muitas técnicas de tolerância a defeito podem ser utilizadas para detectar e localizar problemas no sistema, com o propósito de manutenção.

Os diagnósticos automáticos podem auxiliar muito neste aspecto.

- **Testabilidade (“ Testability”).**

É a habilidade de testar certos atributos dentro de um sistema. Mede a facilidade com a qual certos testes podem ser realizados.

Muitas técnicas que são vitais no sentido de se alcançar “tolerância a defeito” podem ser utilizadas para detectar e localizar problemas aumentando a Testabilidade. Está intimamente relacionada com a manutenabilidade, devido ao fato de diminuir o tempo requerido para identificar e localizar o problema.

- **Dependabilidade (“Dependability”).**

Aglutina os conceitos de Confiabilidade, Disponibilidade, Segurança, Manutenabilidade, desempenho e testabilidade.

2. Evolução Tecnológica e Aplicações

A evolução tecnológica dos Sistemas de Proteção pode ser esquematizada em quatro etapas:

- Sistemas a Relês;
- Sistemas com Tecnologia a Estado Sólido;
- Sistemas CLT redundantes;e
- Sistemas Tolerantes a Defeitos com Redundância Triplicada.

Nos sistemas implementados a Relês tem-se um aumento significativo da sua complexidade á medida que se expande a funcionalidade, tornando, desta forma, a manutenção, a localização de defeitos e prováveis modificações, difíceis de serem realizadas. Convém destacar também os altos custos deste tipo de sistema. Apesar dos relês apresentarem um modo de falha bastante conhecido, este sistemas trabalham apenas com entrada e saídas digitais.

Com relação aos Sistemas Baseados em Lógica do Estado Sólido, estes apresentam boa testabilidade e localização de defeitos. Por outro lado, estes sistemas apresentam uma capacidade limitada de diagnóstico e pouca flexibilidade para modificações, aceitam somente entradas e saídas digitais, além de utilizarem componentes de difícil reposição.

Em função dessas dificuldades e da evolução tecnológica, passou - se a utilizar sistemas com Controladores Lógicos Programáveis - CLT ‘ s redundantes, implementados através de dois processadores paralelos sendo um deles selecionado através de chaveamento. Esses processadores além de aceitarem entrada e saídas analógicas, melhoraram a capacidade de diagnóstico, podendo ser usados justamente com um “Bus de dados”. Nesses sistemas torna-se difícil definir qual o processador que está correto, o chaveamento entre ambos não é seguro, há a necessidade de duplicação das entradas e saídas dos processadores e alto risco em mudança nas suas programações. Neste sentido, vale também destacar a dificuldade em se verificar os programas contidos nos processadores.

No sentido de superar todas dificuldades e com o aumento da complexidade dos Sistemas de Proteção, chegou - se aos Sistemas Tolerantes a Defeito com redundância Modular Triplicada. Dentro desta filosofia destacam-se duas

linhas de trabalho, com ênfase em Hardware e Software, sendo a última adotada com mais frequência.

Alguns dos requisitos desejáveis desta nova arquitetura são:

- uma falha simples no sistema não deve gerar entradas ou saídas erradas nem deve evitar que o sistema funcione conforme projetado;
- qualquer falha deve gerar alarme e uma indicação do local da ocorrência;
- qualquer falha simples deve ser reparada “on -line” sem interrupção da operação do Sistema de Segurança.

Como consequência das necessidades do mundo moderno e da complexidade crescente dos Sistemas de proteção exige-se um trabalho de análise bastante amplo na verificação dos Requisitos Gerais de Segurança, no sentido de caminhar-se em direção a uma melhor “Segurança” e “Qualidade” desses sistemas.

Desta forma deve-se Ter como principais metas nestes Sistemas de Segurança as seguintes características:

- Alta Disponibilidade;
- Baixo tempo Médio para Reparo (“Mean Time to Repair”- MTTR);
- Alto Tempo Médio entre Falhas (“Mean Time Between Failures”- MTBF);
- Alto Tempo Médio entre Falhas Inseguras (“Mean Time Between Unsafe Failures”- MTBUF);
- Metodologia de Programação que dê ênfase á segurança;
- Diagnóstico Exaustivo;
- Interface Amigável;
- Comunicação Segura via Rede; e
- Excelente Documentação.

A Tolerância a Defeito é atualmente requerida, pois os novos sistemas eletrônicos são menos confiáveis.

3. Aplicações

As aplicações de computação tolerante a defeitos podem ser categorizadas em 4 áreas:

- Vida longa
- Computação crítica
- Aspecto de manutenção adiantada/adiada
- Alta disponibilidade

3.1. Aplicações de Vida Longa

Sistemas a bordo de naves e satélites.

Requisitos típicos desta aplicação é Ter uma probabilidade de 0,95 de estar operacional no fim de um período de 10 anos.

No entanto, podem suportar pequenos períodos fora do ar (1 semana).

3.2. Aplicações críticas

Relacionadas com Vidas Humanas, meio-ambiente, ou proteção de equipamento.

- Controladores de Vôo
- Sistemas militares
- Certos controles industriais
- Controle metrô - Ferroviário, etc...
- Indústria química

Requisito típico: Confiabilidade $> 0,97$ no fim de um período de três horas, no entanto os requisitos podem variar em função das particularidades do sistema.

3.3. Aplicações com Manutenção Adiantada/ Adiada

Aplicáveis em sistemas que as operações de manutenção são extremamente custosas, inconvenientes ou difíceis de se realizar.

Exemplo:

- Estações de processamento remoto. (custo) (Centrais Telefônicas)
- Certas aplicações espaciais (local).

Visitas periódicas que realizam uma manutenção preventiva muito forte.

Evitar manutenção não programada.

Durante os períodos entre visitas, o sistema manipula as falhas e serviços errados de maneira autônoma.

3.4. Aplicações de Alta Disponibilidade

Bons exemplos: Sistemas Bancários e "Time-Shared"

Os usuários destes sistemas querem Ter uma probabilidade alta de receber o serviço quando requisitado.

4. Definições Fundamentais

Algumas definições fundamentais na área de confiabilidade e disponibilidade devem ser formalizadas visando um maior esclarecimento da terminologia sendo utilizada. A figura 4.1 apresenta a relação entre os termos *Falha*, *Erro* e *Disfunção*.

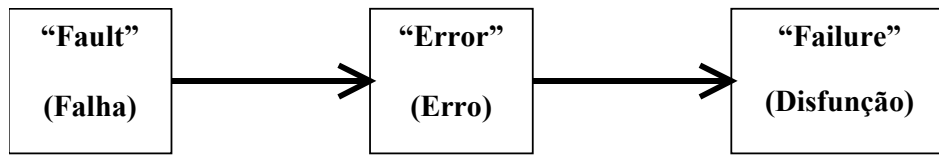


Figura 4.1. Relação entre Falha, Erro e Disfunção

O termo *Falha* corresponde a um problema intrínseco de um componente. Pode corresponder a uma *falha física* ou *imperfeição*, que ocorre dentro de um componente de hardware ou software.

Como exemplo de *falha* pode ser citado um condutor aberto ou fechado, uma imperfeição física num dispositivo semicondutor ou, um laço infinito num programa de computador.

O termo *Erro* corresponde à manifestação interna de uma *falha*. Especificamente um *erro* é um desvio da precisão ou correção esperada de uma informação interna ao sistema.

Uma *Disfunção* ocorre quando o sistema desempenha incorretamente uma de suas funções.

É também denominada como mau funcionamento. Dessa forma, uma *disfunção* corresponde também ao desempenho de alguma função numa quantidade ou qualidade abaixo do normal. A *disfunção* é uma exteriorização do *erro*.

Seja o exemplo de um Votador dois de três, apresentado na figura 4.2.

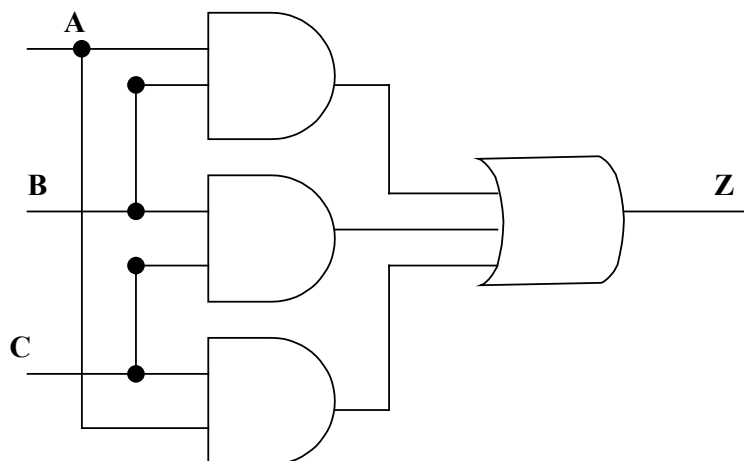


Figura 4.2 – Circuito Votador

A Tabela da Verdade deste circuito está apresentada na tabela 4.1, que se segue.

A	B	C	Z
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

Tabela 4.1 – Tabela de Verdade do Circuito Somador

Supondo a ocorrência de uma *Falha* intrínseca no componente Votador, como por exemplo, um curto entre o ponto A e o V_{cc} , a nova Tabela da Verdade passa a apresentar a seguinte constituição apresentada na Tabela 4.2.

A	B	C	Z	
0	0	0	0	
0	0	1	1	(*)
0	1	0	1	(*)
0	1	1	1	
1	0	0	0	
1	0	1	1	
1	1	0	1	
1	1	1	1	

Tabela 4.2. Nova Tabela da Verdade do Circuito Somador

Pode-se observar que os valores com (*) correspondem às saídas *Z* que contém *erros* na informação interna do sistema sendo considerado. Se a saída *Z* estiver controlando um circuito externo, como por exemplo, um circuito a Relê, toda vez que os dado contaminado de entrada *A* for utilizado, o *erro* na informação *Z* será exteriorizado, transformando-se em uma *disfunção*.

A *Falha* está relacionada com o *Universo Físico*, representado através dos dispositivos semicondutores, fontes, além de outras entidades físicas de hardware e componentes de software.

O *Erro* está relacionado com o *Universo de Informação*, podendo afetar sua precisão ou conteúdo. Os termos aplicáveis a este universo podem incluir *erro de paridade*, *erro de bit*, *etc...*

A *Disfunção* está relacionada com o *Universo Externo*, ou seja, aquele em que o usuário final enxerga o sistema e o seu mau funcionamento.

Com relação aos instantes de ocorrência entre uma *Falha*, um *Erro* e uma *Disfunção*, pode-se definir o conceito de *Latência*.

A *Latência* de uma *Falha* corresponde ao intervalo de tempo entre a ocorrência de uma *Falha* e o aparecimento do correspondente *Erro*.

A *Latência* de um *Erro* corresponde ao intervalo de tempo entre a ocorrência de um *Erro* e o aparecimento da correspondente *Disfunção*.

4.1 Causas das Falhas

As *Falhas* podem ser resultado de uma variedade de eventos que ocorrem internamente a um determinado componente eletrônico ou computacional, externamente ao componente ou durante o projeto do componente ou sistema.

As causas possíveis das *Falhas* podem, desta forma, estar associadas com os seguintes aspectos:

- Problema na Especificação;
- Problema na Implementação;
- Desgaste do Componente;
- Fatores Externos.

A figura 4.3. a seguir ilustra a relação entre as causas fundamentais das *Falhas* e a relação com os *Erros* e *Disfunções*.

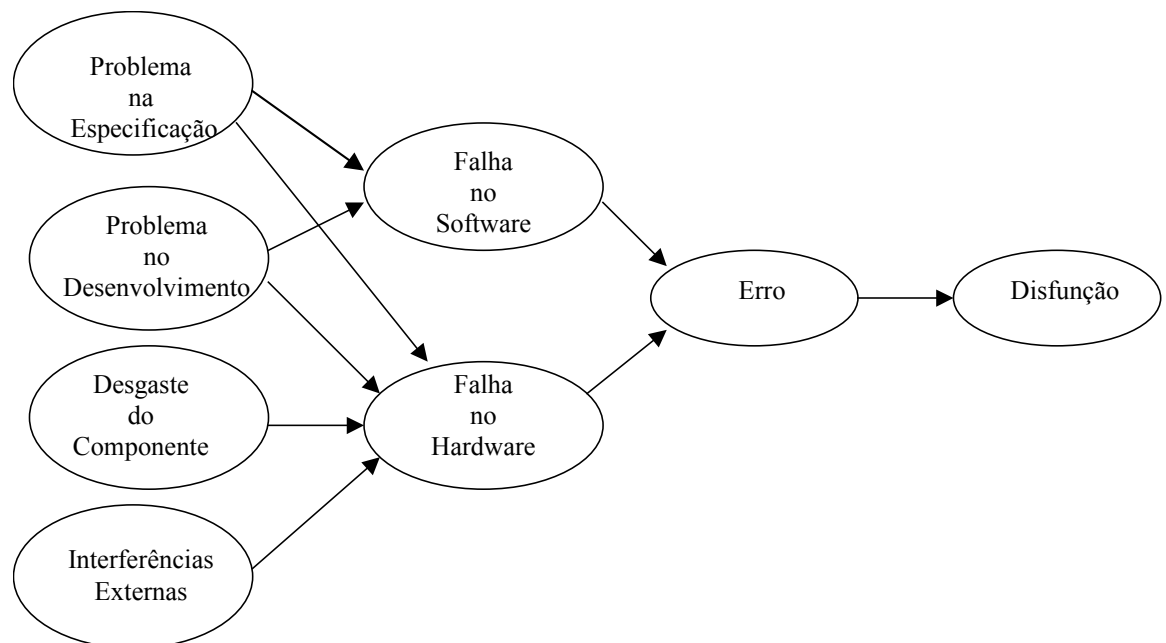


Figura 4.3 – Causas das Falhas, Erros e Disfunção

As Falhas podem ser caracterizadas de acordo com sua Causa, Natureza, Duração, Extensão e Valor. O detalhe desta relação é apresentado de forma esquemática na tabela 4.3.

Características das Falhas	Causa	Problema na Especificação	
		Problema na Implementação	
		Desgaste	
		Interferências Externas	
	Natureza	Hardware	Analógico
			Digital
		Software	
	Duração	Permanente	
		Intermitente	
		Transiente	
	Extensão	Local	
		Global	
	Valor	Determinado	
		Indeterminado	

Tabela 4.3 – Características das Falhas

4.2. Filosofias de Projeto para Combater as Falhas

Conforme apresentado anteriormente, podem existir *Falhas*, *Erros* e *Disfunções*. Existem técnicas que podem ser utilizadas para evitar as Falhas, outras para mascarar as Falhas, correção dos Erros e tolerâncias a Falhas/Erros. Estas técnicas são apresentadas na figura 4.4.

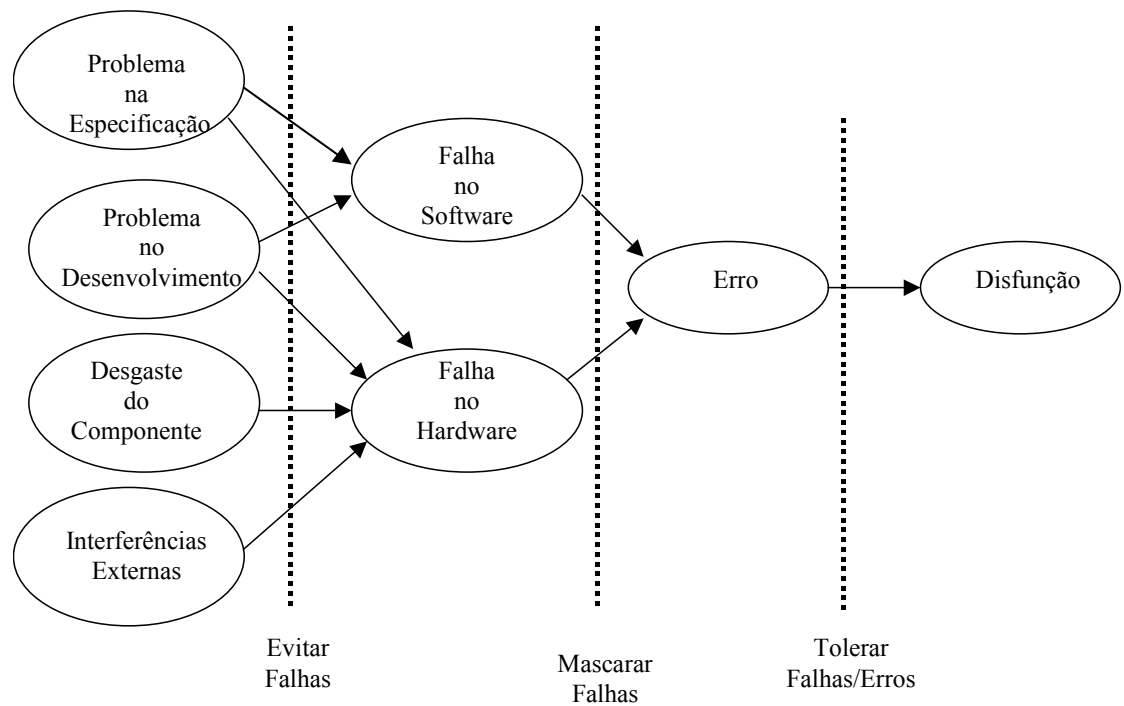


Figura 4.4. – Técnicas para Combater as Falhas/Erros

Para se Evitar as Falhas podem ser citadas as seguintes técnicas:

- Revisões de Projeto;
- Testes;
- Métodos de Controle de Qualidade;
- Métodos Preventivos.

Para se Mascara as Falhas podem ser realizadas as seguintes ações:

- Correção dos Erros;
- Votação Majoritária, entre outros.

Com relação à técnica de Tolerância à Falhas/Erros, o sistema deve apresentar a habilidade de continuar desempenhando corretamente suas funções mesmo após a ocorrência da Falha/Erro. Nesse sentido devem ser tomadas algumas atitudes para que essa tolerância seja alcançada. Podem ser adotadas as seguintes tarefas:

- Mascaramento das Falhas
- Reconfiguração do sistema: deve-se detectar e localizar a Falha/Erro e reconfigurar o sistema visando a remoção do componente em falha.
 - A detecção da Falha irá determinar a necessidade das outras ações. Está relacionada com o Fator de Cobertura de Defeitos.
 - A localização da Falha irá direcionar a forma de recuperação
 - Isolamento da Falha, prevenindo que seus efeitos se propaguem através do sistema.
- Recuperação da Falha, fazendo com que o sistema permaneça operacional através de uma ação de reconfiguração.

5. Técnicas de Projeto para Alcançar Tolerância a Defeitos.

O conceito de Redundância

No início o conceito de redundância estava intimamente relacionado com a repetição de bloco de hardware.

De acordo com Johnson: “Redundância é simplesmente a adição de informação, recursos ou tempo além do que é necessário para a operação normal do sistema”.

FORMAS DE REDUNDÂNCIA

1. Redundância de Hardware: adição extra de hardware, geralmente com o objetivo de detenção ou tolerância a defeito.
2. Redundância de Software: adição extra de software, além do necessário para realizar uma função para detectar ou possivelmente tolerar defeitos.
3. Redundância de Informação: adição extra de informação além do requerido para implementar uma dada função; ex.: código de detecção de erro.
4. Redundância de tempo: usa tempo adicional para realizar as funções de um sistema de maneira que a detecção e a tolerância a defeitos possam ser alcançadas. (Ex: detectar erros transientes).

O uso de redundância pode fornecer capacidade adicional num sistema. Na realidade, se tolerância a defeito ou detecção de defeito são requeridos, alguma forma de redundância é requerida.

Mas a redundância pode ter um impacto importante no desempenho do sistema, tamanho, peso, consumo de energia, custo e confiabilidade.

5.1. Redundância de Hardware.

Há três tipos básicos de redundância de hardware:

- Passiva: usa o conceito de mascaramento de defeitos para esconder a ocorrência de defeitos e prevenir que defeitos resultem em erros. Não requer nenhuma ação do sistema ou do operador.
- Ativa: conhecido como método dinâmico, alcança tolerância a defeito por detecção de defeitos e realizando alguma ação para remover o hardware com defeito. (reconfiguração).

Deteção de defeitos → localização do defeito → recuperação do defeito.

- Híbrida: combina técnicas passivas e ativas.

5.1.1. Redundância de Hardware Passiva.

Baseia-se no mecanismo de votação (votação majoritária), para mascarar a ocorrência de defeitos.

Redundância de Hardware Passiva mais comum:

“TMR - Triple Modular Redundancy”

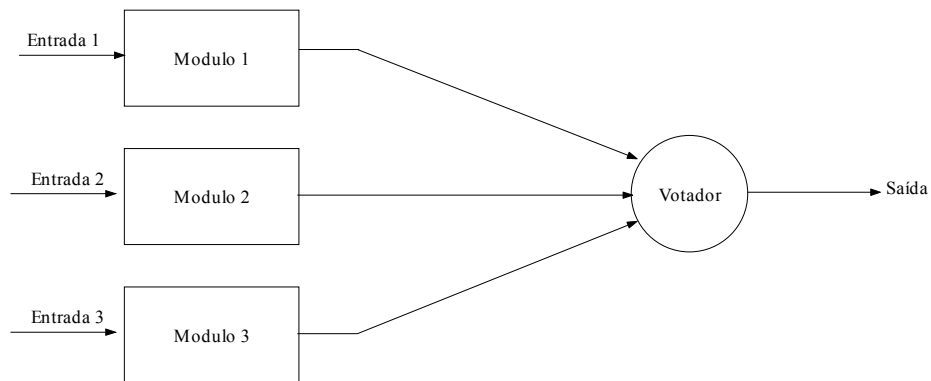


Figura 5.1

O TMR pode ser aplicado á software onde três versões diferentes do programa são usadas para proteger contra defeito no software em uma das três versões.

A maior dificuldade no TMR é o votador; se o votador falha, o sistema completo falha.

Em outras palavras, a confiabilidade do TMR simples não pode ser melhor do que a confiabilidade do votador. (“fail-safe”).

Muitas técnicas podem ser usadas para recuperar a falha no votador.

Uma abordagem é triplicar o votador e fornecer três resultados independentes.

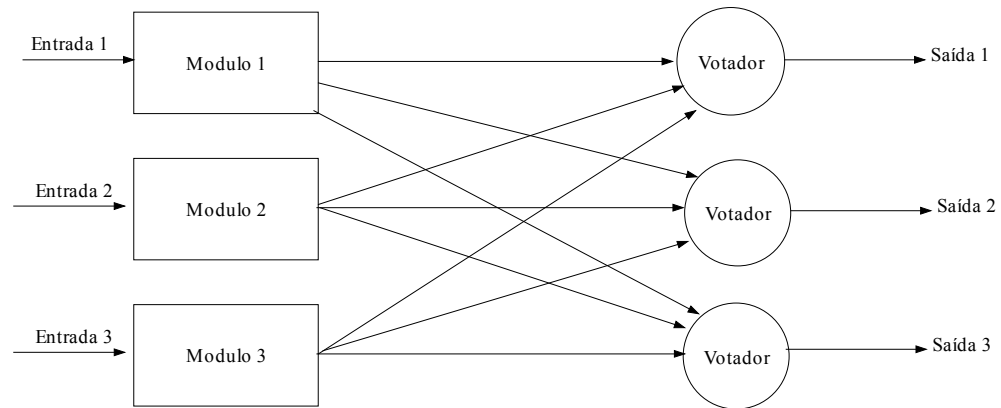


Figura 5.2

Muitos estágios do TMR podem ser interconectados usando a seguinte abordagem:

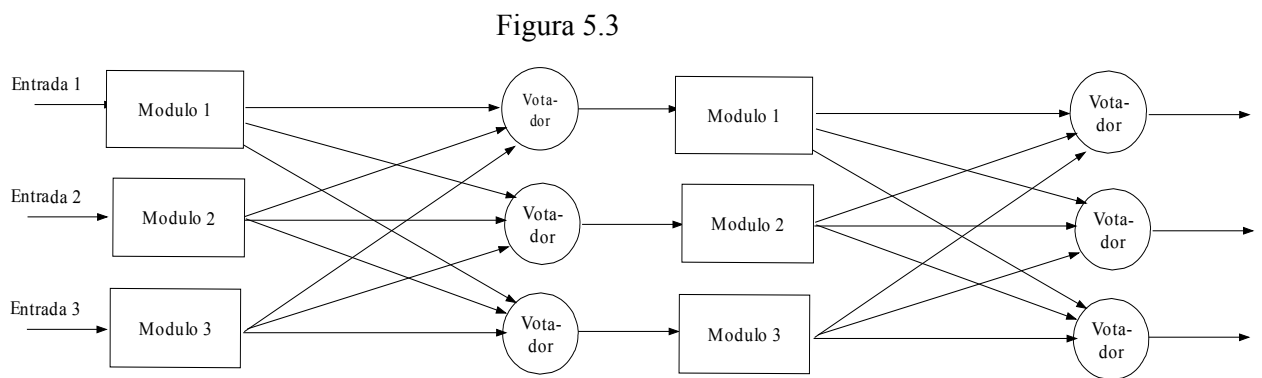


Figura 5.3

Redundância N-Modular

Generalização do TMR. A diferença é que são utilizados N módulos.

Normalmente N é um número ímpar, para efeitos de votação majoritária.

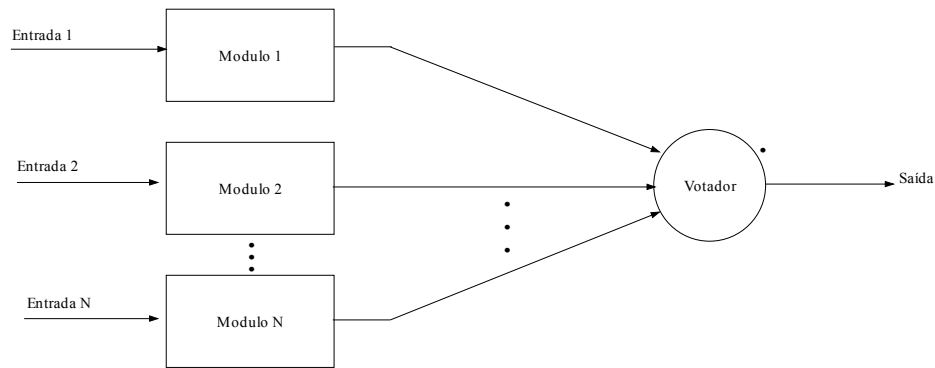


Figura 5.4

Neste caso mais do que um módulo com defeito podem ser tolerados.

Em muitas aplicações críticas, dois defeitos devem ser tolerados para permitir alcançar a confiabilidade requerida e a capacidade de tolerar defeitos.

Limitante de N: consumo, peso, custo, confiabilidade do próprio módulo e tamanho.

Técnicas de votação

A votação não apenas engloba decisões de projeto (em que instante votar), mas também impõe muitas considerações como, votação por hardware ou por software.

Votador em hardware para dados digitais são relativamente simples de projetar.

O circuito a seguir implementa um votador majoritário

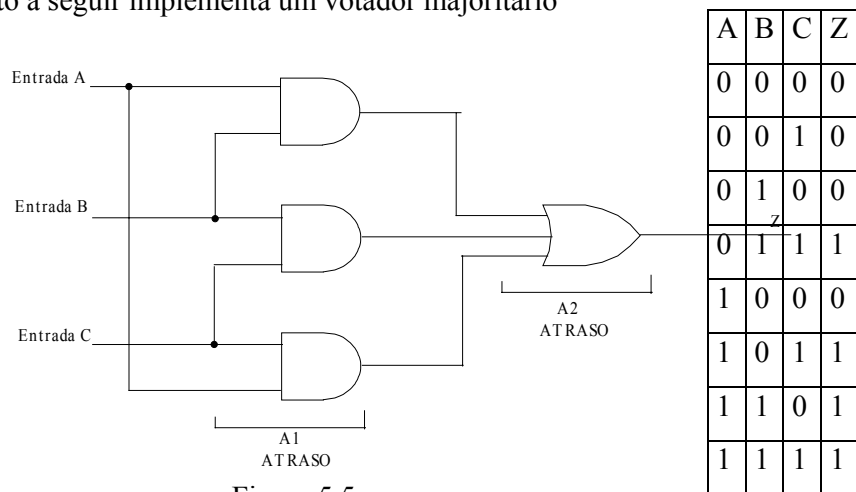


Figura 5.5

Na maioria das aplicações práticas, o “timing” é um aspecto crítico no procedimento de votação. Se os valores chegam no votador em tempos poucos diferentes, resultados incorretos podem ser gerados temporariamente. Para contornar estes problemas, flip-flop’s podem ser utilizados nas entradas dos votadores para sincronizar o processo de votação.

O circuito a seguir procura resolver este problema.

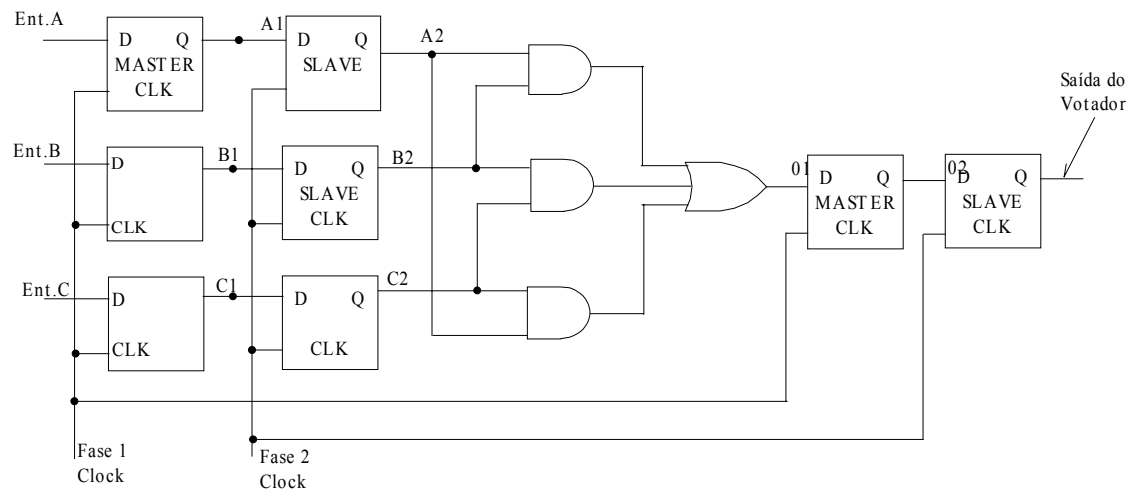


Figura 5.6

CARTA DO TEMPO

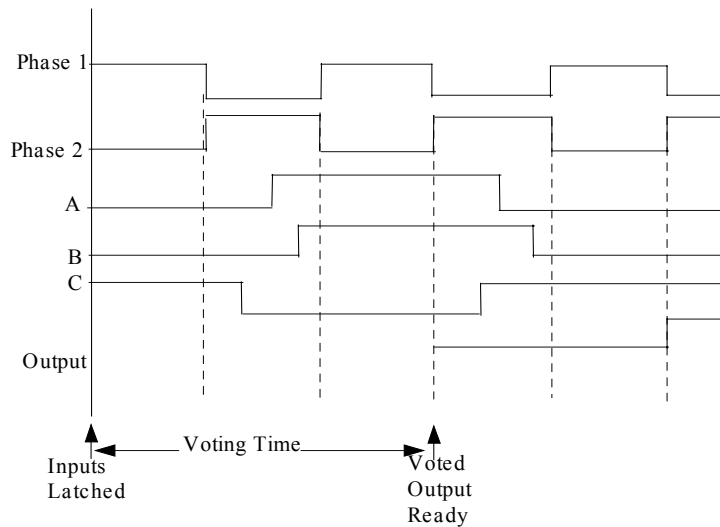
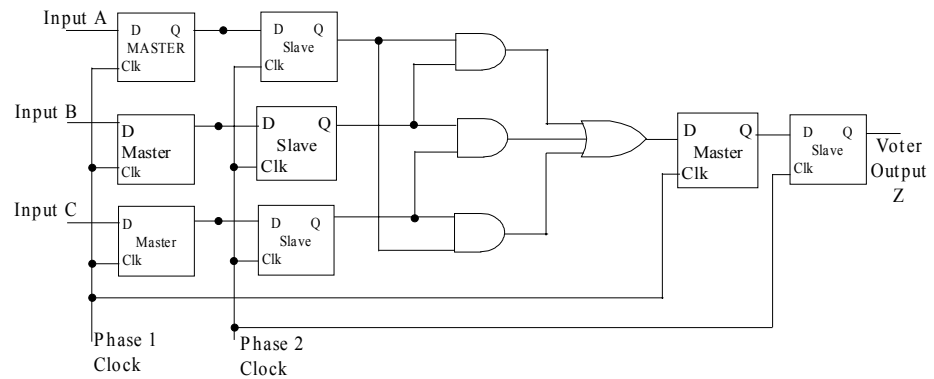


Figura 5.7

Quando se tem votação por hardware:

- atraso depende dos atrasos das pastilhas/componentes usados para construir o circuito.
- Número de componentes bastante grande aumento do consumo, peso, tamanho.

Quando se tem votação por software:

- mínimo hardware;
- simples modificar o software;
- pode requerer mais tempo que um hardware dedicado.

As decisões pela votação em hardware ou software devem levar em considerações os seguintes aspectos:

- 1) Disponibilidade do processador para realizar a votação
- 2) Velocidade que a votação deve ser realizada
- 3) Criticidade quanto a espaço, potência e peso.
- 4) número total de votadores
- 5) flexibilidade para futuras mudanças.

Um outro problema em votação é que os valores de uma votação podem não concordar perfeitamente devido a precisão e tolerância.

Desta forma, uma saída é através da técnica “Mid-Value” (entre o valor correto sem erro e o com erro em votação com número ímpar de módulos).

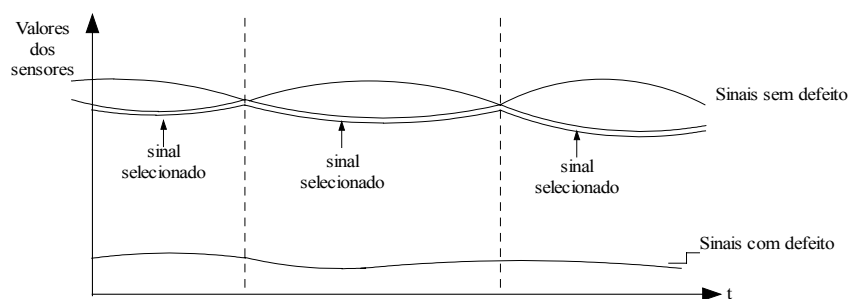


Figura 5.8

Outra maneira é ignorar os últimos bits menos significativas da informação em função da precisão dos componentes sendo utilizados. Em outras palavras, a votação é realizada sobre os R bits mais significativos.

5.1.2. Redundância de Hardware Ativa.

Esta técnica atinge a tolerância a defeito através da detecção do defeito, localização do defeito e recuperação do defeito. (detecção do erro, localização do erro e recuperação do erro).

Neste tipo de redundância o erro é tolerado temporariamente até que o sistema seja reconfigurado.

Exemplos dessas técnicas:

a) Duplicação com Comparação

Este tipo de técnica é apresentado na figura a seguir:

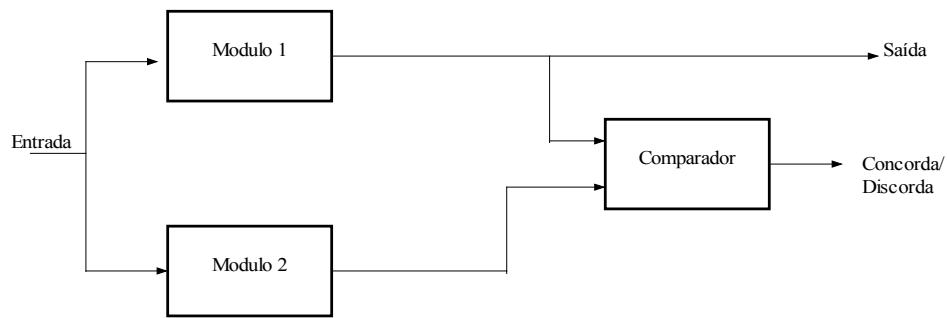


Figura 5.9

Cada módulo realiza as mesmas funções em paralelo. Se ocorrer uma discordância, mensagem de erro é gerada.

Na maioria dos casos é detectada a existência do defeito, mas não é tolerado, pois não há meios de saber qual módulo está com defeito. Mas é um meio de se fazer detecção de defeito.

Problemas:

- Entradas com defeito, resultados errados em ambos os módulos;
- Comparador, pode não executar sua função exatamente (precisão);
- Falha no Comparador: não indicar erros quando tem.

A técnica que resolve alguns dos problemas anteriores é:

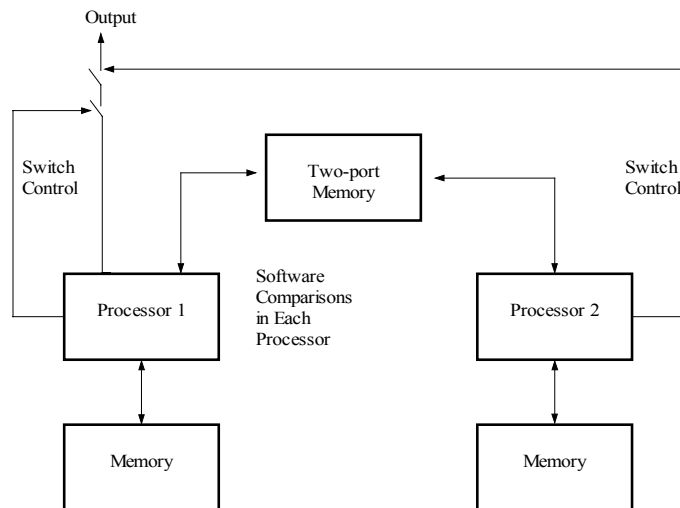


Figura 5.10

Os processadores realizam funções idênticas.

Se a memória compartilhada falha, ambos os processadores detectam a discrepância de valores com os valores de sua própria memória.

Se um processador falhar, a falha é detectada pelo outro processador pela discrepância de valores.

As chaves podem ser implementadas na forma “digital” ou “analogicamente” dependendo da aplicação.

Ambos os processadores devem concordar antes que o sistema permita produzir uma saída.

b) Unidade “Standby”.

A Configuração está apresentada na figura a seguir.

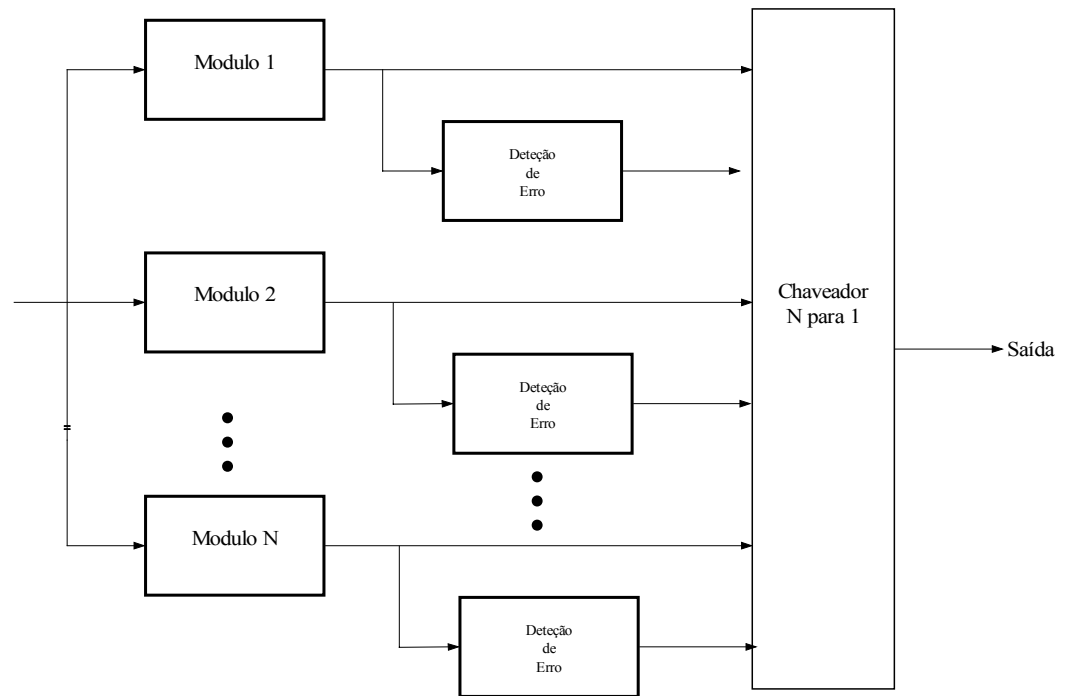


Figura 5.11

Nesta técnica um módulo é operacional e um ou mais módulos servem como reservas.

A reconfiguração neste caso pode ser vista conceitualmente como uma chave cuja saída é selecionada de um dos módulos bons. Se todos estão bons, pode-se fixar um esquema de prioridades.

Qualquer módulo com erro é eliminado desta consideração.

O sistema precisa tolerar uma interrupção durante o processo de reconfiguração.

Em função do tempo necessário pode-se ter “hot-standby” (opera em sincronia Ex: Controle de reação química) ou “ cold standby”(não alimenta - Ex: satélite) reserva.

No caso “Hot Stand By” utiliza-se de unidades sobressalentes energizadas.

No caso “Cold Standy By” utilizam-se de unidades sobressalentes não energizadas.

Uma “questão chave” nesta abordagem é o esquema de Detecção de Erro/Detecção de Defeito usado para identificar o módulo em falha.

Duas técnicas são apresentadas a seguir:

b.1) “Pair-and-a-Spare”

A arquitetura está apresentada a seguir:

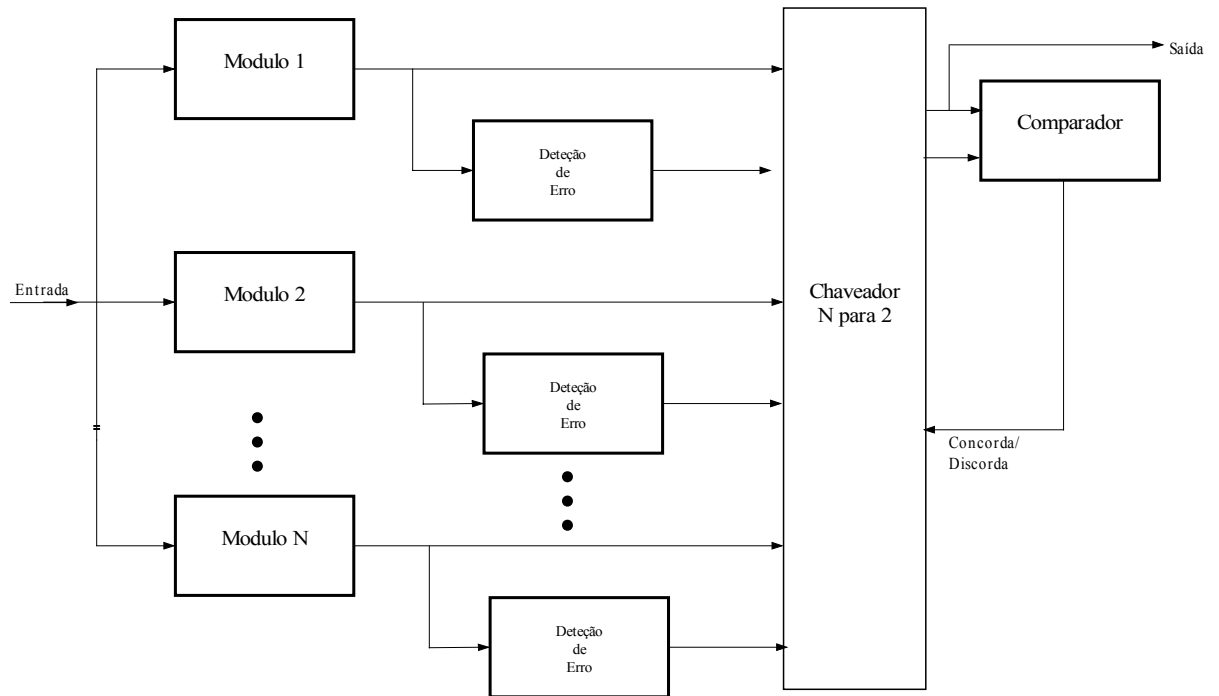


Figura 5.12

Nesta técnica dois módulos estão operando em paralelo durante todo o tempo e seus resultados são comparados para detectar erro. O sinal de erro da comparação é utilizado para iniciar o processo de reconfiguração que vai remover o módulo em falha e o substituir á pelas reservas.

A chave usa a informação de erro do comparador e do módulo individual para manter dois módulos livres de erro operando em duplicação.

Outra saída seria usar os “módulos em par” e quando houver erro descarta o “par de módulos”, em função da saída da comparação.

b.2) Temporizador “ **watchdog**” (comparação com carteiro)

Neste caso é necessária uma ação para indicar o estado “livre de defeito”.

A ausência da ação indica o defeito/erro.

O “watchdog” é um temporizador que deve ser resetado numa base repetitiva.

A falha do sistema em resetar resultará no sistema ser “resetado” ou “desligado” antes que outra falha aconteça no sistema.

A hipótese fundamental aqui é que o sistema seja livre de defeito se ele tem a capacidade de repetir periodicamente uma função como “resetar” um temporizador.

A frequência que o temporizador deve ser “resetado” depende do sistema.

Exemplo:

- Sistema de controle de Avião < 100 ms;
- Sistema bancário < 1seg;
- Controle de metrô < 2 seg.

5.1.3. Redundância de Hardware Híbrida.

Combina as características da Passiva e Ativa. Usa-se Mascaramento de Defeitos e Reconfiguração (Detecção, Localização e Recuperação do Defeito).

É mais utilizada quando exige-se um alto grau de integridade computacional. (Confiabilidade).

a) Redundância N-Modular Reservas.

N módulos arrumados numa configuração com votação. Além disso, sobressalentes são disponíveis para substituir as unidades em falha.

Esta técnica está apresentada na figura a seguir:

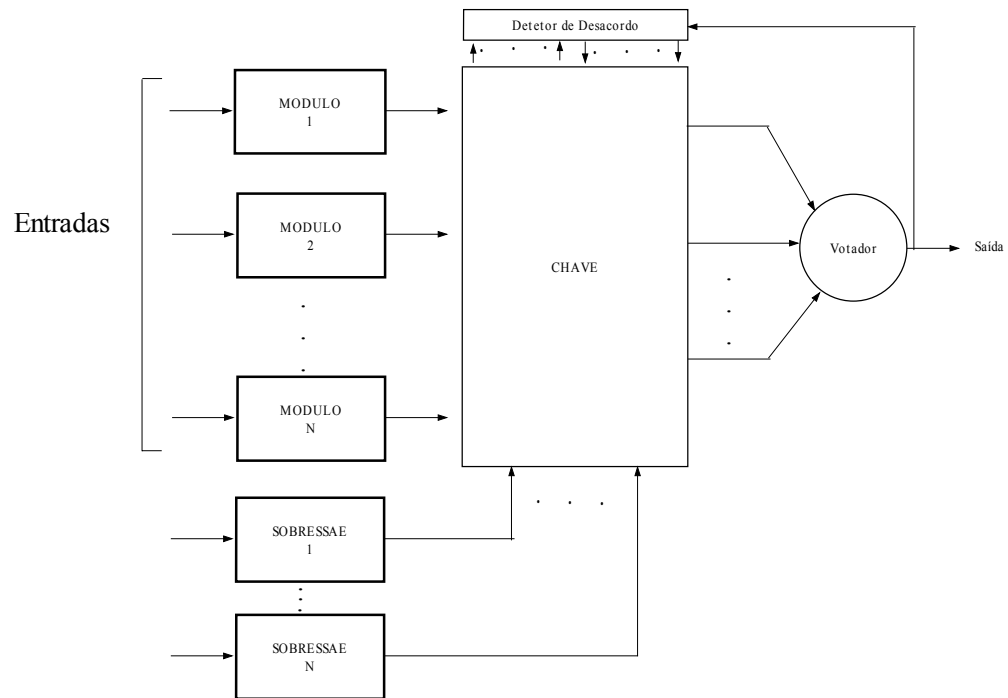


Figura 5.13

A configuração inicial permanece até que o detector de desacordo determine que existe uma unidade falha. Uma abordagem é comparar a saída do votador com as saídas dos módulos individualmente.

Neste caso a Unidade Reserva é recolocada no lugar do módulo falho.

b) Redundância com Auto-Remoção (“Self-Purging”)

A diferença é que neste caso todas as unidades participam ativamente, conforme ilustra a figura a seguir:

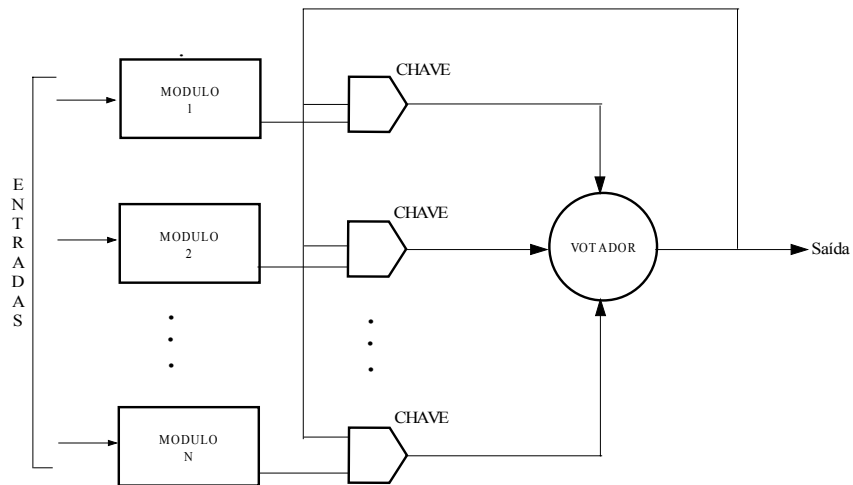


Figura 5.14

Cada módulo tem a capacidade de se auto-remover do sistema no caso de sua saída discordar da saída do votador.

Neste caso o votador usa a técnica “threshold gate”. (através de componentes analógicos, na maioria).

x_i ... entradas binárias n ...nº de entradas

z ... saída binária

w_i .. pesos

T ... valor de “threshold” especificado.

$$Z = \begin{cases} 1, \text{ se } \sum_{i=1}^n w_i \cdot x_i \geq T \\ 0, \text{ se } \sum_{i=1}^n w_i \cdot x_i < T \end{cases}$$

Exemplos “Threshold gate” de 3 entradas.

$$w_i = 1 \text{ e } T = 2$$

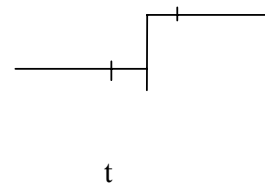
Entradas

X1	X2	X3	Σ	Saída
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	2	1
1	0	0	1	0
1	0	1	2	1
1	1	0	2	1
1	1	1	3	1

Uma abordagem é força para zero o peso de todos os módulos em falha, não contribuindo neste caso para o valor de saída. A chave pode controlar o valor deste peso, associando o valor “zero” se a saída do módulo discorda da saída do sistema.

Implementação da chave utilizando Flip-Flop tipo JK:

J	K	$Q(t+1) *$	Estado $t+1$
0	0	$Q(t)$	mantém
0	1	0	reset
1	0	1	set
1	1	$Q(t+1)$	muda



A figura a seguir apresenta o esquema da chave:

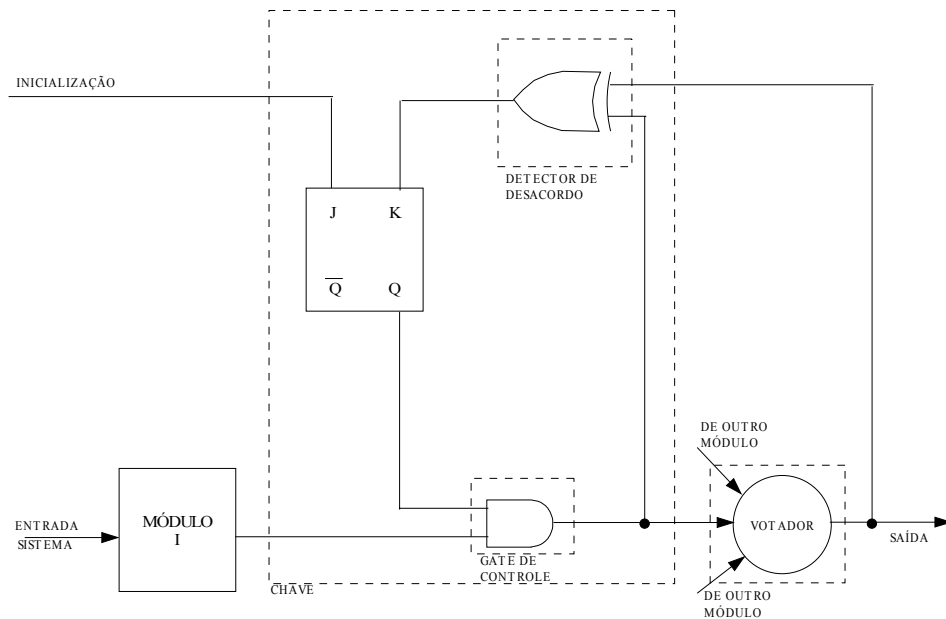


Figura 5.15

O mecanismo de iniciação permite um módulo, desabilitado uma vez, contribuir novamente no processo de votação, se ele for substituído por um sem falha.

- Exemplo: “Somador de 1bit”

Três somadores são usados para redundância tripla. Neste caso são necessários dois votadores: votador do bit soma e votador do bit carry. (vai um).

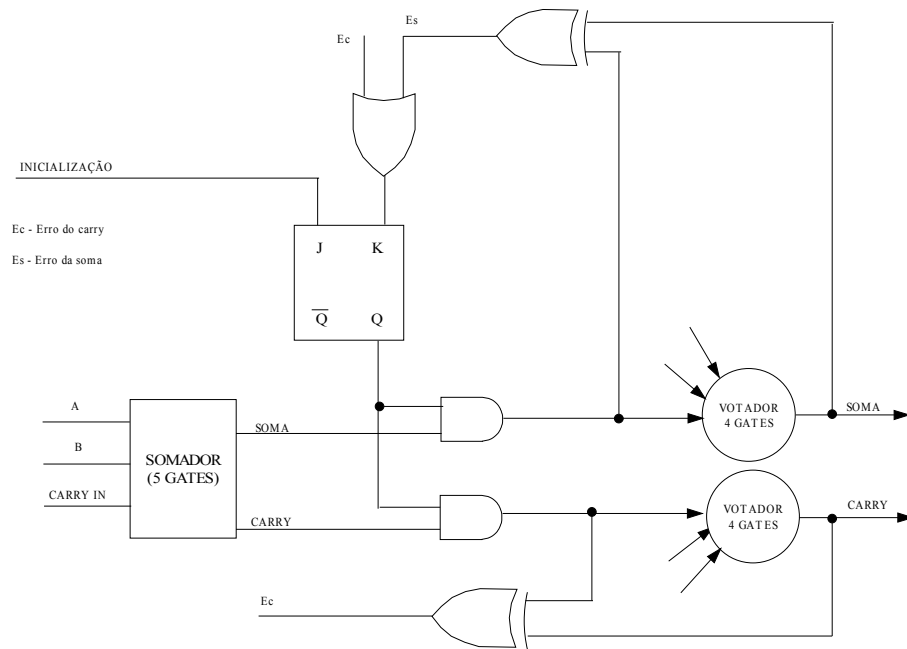


Figura 5.16

c) Redundância Modular “Sift-Out”

A estrutura básica está apresentada a seguir composta por comparadores, detectores e coletores.

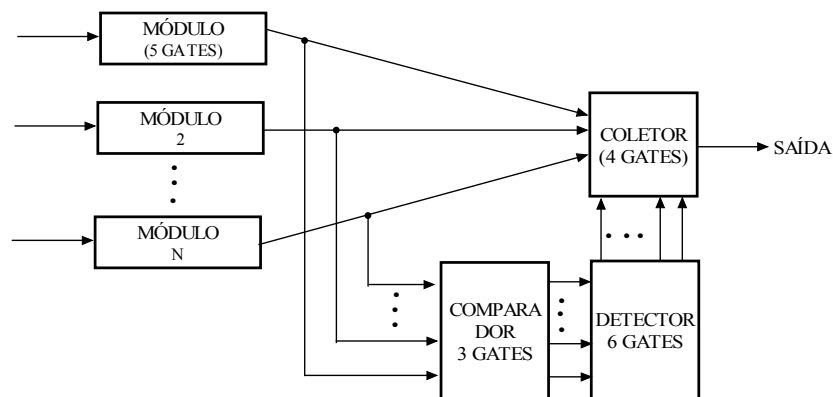


Figura 5.17

- Comparador: compara a saída de cada módulo com os demais. É produzido um sinal para cada comparação.

$C_n, 2 = - - n ! - -$ comparações para n módulos.

$$2 ! (n-2) !$$

A saída é “1” se as duas saídas discordam;

A saída é “0” se as duas saídas concordam.

Exemplo para 3 módulos:

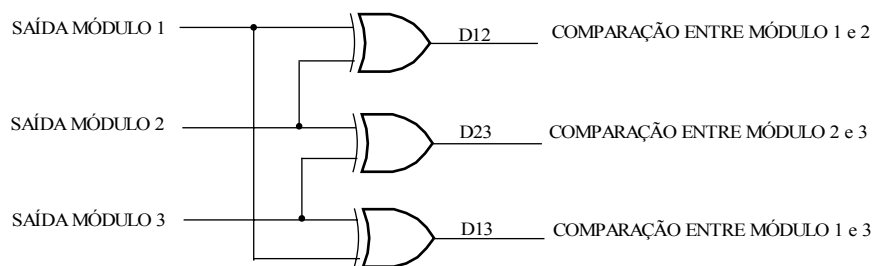


Figura 5.18

- Detetor: determina quais comparações discordam e desabilita a unidade (módulo) que discorda com a maioria dos módulos. A saída é para cada um dos módulos:

- “1”: o módulo discorda com a maioria dos módulos
- “0” o módulo concorda com a maioria dos módulos.

Usa o mesmo mecanismo de flip-flops tipo JK. Exemplo para 3 módulos está na figura 5.19.

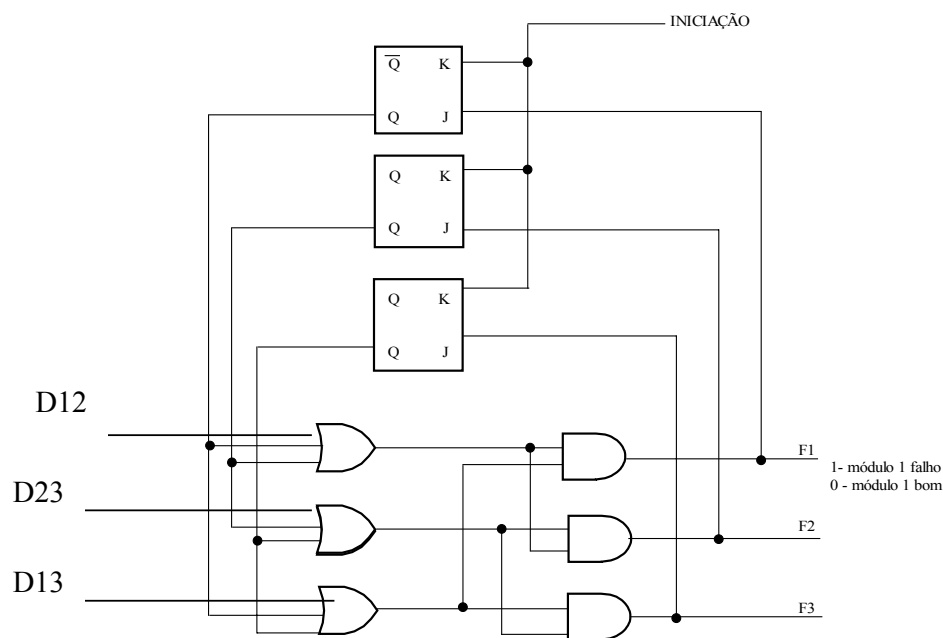


Figura 5.19 - Exemplo: Somador de 1 bit

O módulo que é indicado como falho não influencia na saída do sistema.

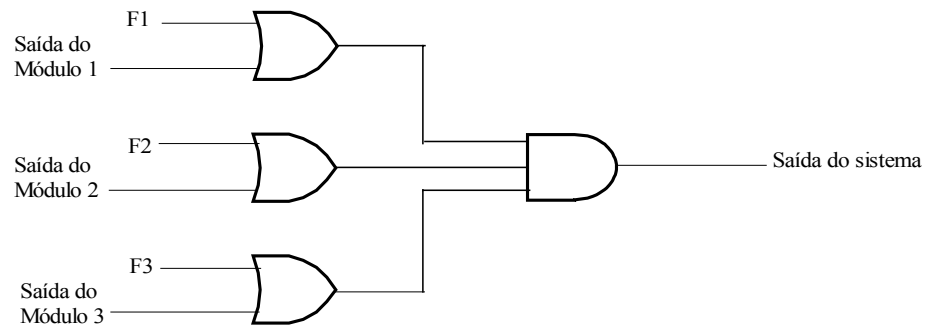


Figura 5.20 – Coletor

O esquema do coletor é apresentado na figura 5.20.

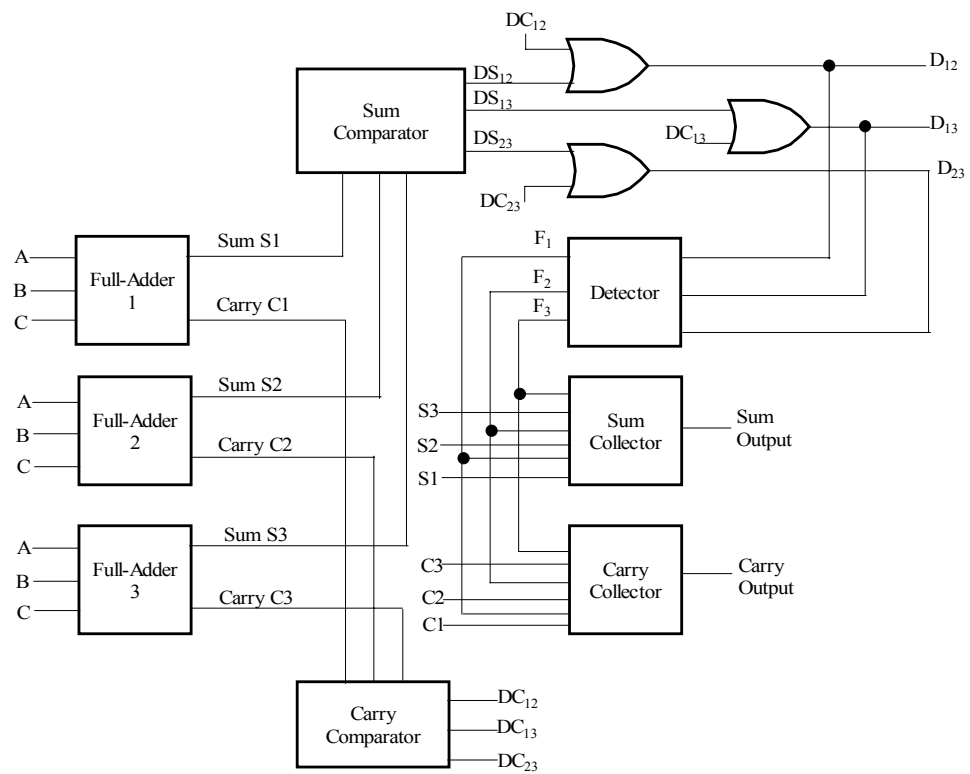


Figura 5.21

COMPARAÇÃO	“AUTO REMOÇÃO”	SIF-OUT”
No de gates:	38	38 * ← interessante
no. de flip flops:	3	3

d) Arquitetura Tripla-Duplex.

Combina redundância triplicada com comparação.

O TMR permite mascarar defeitos. A duplicação com comparação permite defeitos serem detectados e módulos com defeitos removidos da votação.

O esquema desta solução é apresentado a seguir:

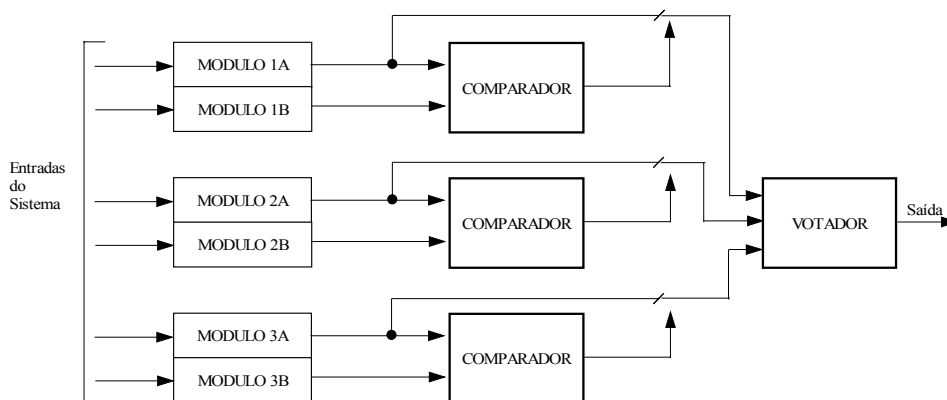


Figura 5.22

5.2. Redundância de Informação

É adição de informação para permitir detecção de defeito, mascaramento de defeito ou possivelmente tolerância ao defeito.

EX.: código de detecção e correção de erros.

Algumas definições:

- Código: meio de representar informação, ou dados, usando um conjunto bem definido de regras.
- Palavras de Código: coleção de símbolos usados para representar um pedaço particular de dados baseados num código específico.
- Código Binário: os símbolos são “0”s e “1”s.

Um conceito fundamental na caracterização dos códigos de detecção e correção de erros é a “Distância Hamming”.

A “Distância Hamming” entre duas palavras binárias é o número de posições de bits na qual as duas palavras diferem.

$$0000 \text{ x } 0101 \quad \text{dist. hamming} = 2.$$

Distância de código.

É a mínima distância hamming entre duas palavras de código válidas.

Exemplo:

Distância de código = 3

Qualquer erro em único bit pode ser corrigido, pois vai ter uma distância de hamming de 1 do código correto e uma distância de hamming de pelo menos 2 dos demais códigos válidos.

Regra geral:

c... No de bits que podem ser corrigidos de erros.

d... No de bits que podem ser detectados erros.

Hd- distância de Hamming

$$2c + d + 1 < H_d$$

Código Separável: para obter o código original basta retirar os bits de código ou de verificação.

Código não Separável: para obter o código original é necessário um processamento mais complexo.

5.2.1. Código de Paridade

Código de paridade de um bit requer a adição de apenas um bit á palavra.

Se o bit extra resulta num número total de “1” s ser ímpar, o código é referenciado como paridade ímpar.

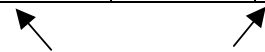
Se o resultado do No de “1” s é par, o código é de paridade par.

Ambos tem uma distância de Hamming de 2 permitindo a detecção de erro em 1 bits, mas não a correção.

Trata-se de um código separável.

Exemplo:

Digito Decimal	BCD	BCD c/ Paridade	Impar	BCD c/ Paridade	Par
0	0000	0000	1	0000	0
1	0001	0001	0	0001	1
2	0010	0010	0	0010	1
3	0011	0011	1	0011	0
4	0100	0100	0	0100	1
5	0101	0101	1	0101	0
6	0110	0110	1	0110	0
7	0111	0111	0	0111	1
8	1000	1000	0	1000	1
9	1001	1001	1	1001	0


bit de paridade

Aplicação: Memórias em sistemas computacionais.

O esquema de aplicação é mostrado a seguir:

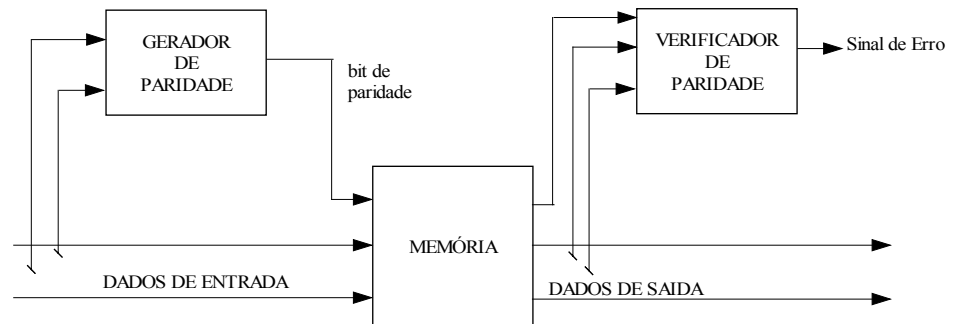


Figura 5.23

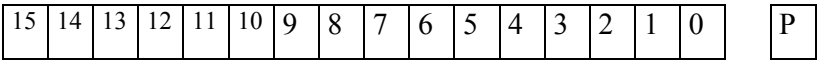
Existem outros tipos de paridade para detectar mais erros no código:

- 5. Paridade bit-por-palavra;
- 5. Paridade bit-por-byte;
- 5. Paridade bit-por-chips;
- 5. Paridade bit-por-multiplos-chips;
- 5. Paridade interlaçada.

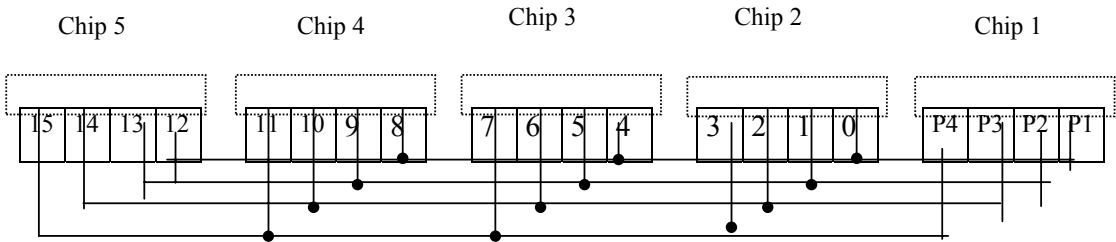
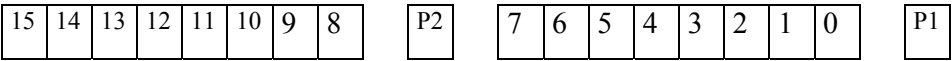
A figura a seguir apresenta o esquema geral destes bits de paridade

Odd or
Even

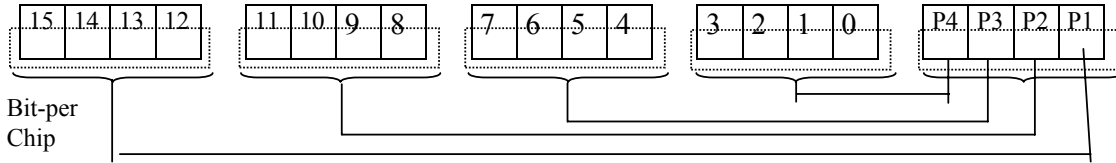
Bit-per Word



Bit-per Byte



Bit-per
Multiple-
Chips



Bit-per
Chip

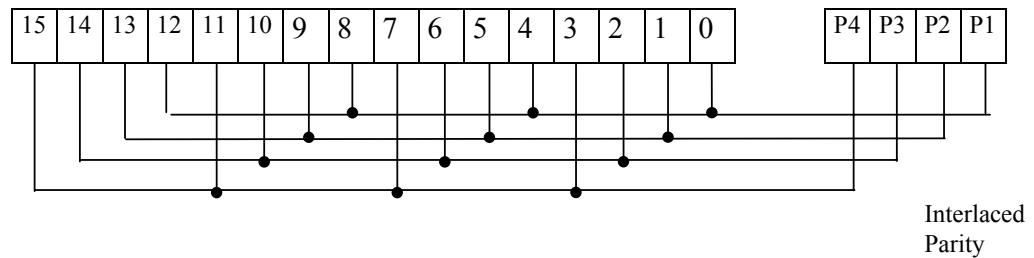


Figura 5.24

a) Bit por palavra

Quando o bus falha em “1”, inclusive o bit de paridade, a paridade Impar não detecta.

Quando o bus falha em “0”, inclusive o bit de paridade, a paridade par não detecta.

b) Bit por Byte -1 grupo paridade impar.

-1 grupo paridade par.

Falha tudo 1 → paridade par detecta

Falha tudo 0 → paridade impar detecta.

Ambos, “bits por palavra” e “bit por Byte”, não detectam erros múltiplos, no caso de falha de uma pastilha de memória contendo bits de um dado, pois a distância de código é um.

Para resolver este fato tem-se:

c) Bit por múltiplos Chips

Detecta erro numa pastilha, total da memória todos os bits de paridade detectam este erro (podem detectar) usando tipos de paridade diferentes.

Pode ser difícil de detectar os Chips com defeito.

Surgiu então,

d) Bit por chip tem mesmo problema de bit por palavra.

e posteriormente surgiu,

e) Bit com Paridade Interlaçada (“overlapping”).

Os grupos não estão relacionados com as pastilhas físicas. Muito usado quando erros em bits adjacentes são mais prováveis. Ex.: “bus” paralelo.

A paridade interlaçada apresenta grupos que são formados com cada bit aparecendo em mais de um grupo.

Neste caso o erro pode ser localizado além de ser detectado.

Este tipo de paridade é o conceito do código de correção de erro Hamming.

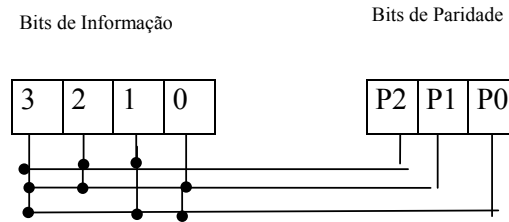
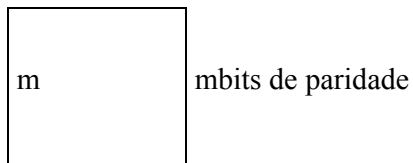
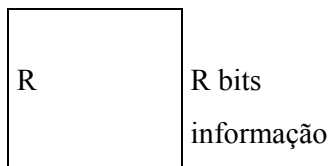


Figura 5.25

Erro no bit		Erro no bit de paridade
3	→	P2 P1 P0 111 7
2	→	P2 P1 110 6
1	→	P2 P0 101 5
0	→	P1 P0 011 3
P2	→	P2 100 4
P1	→	P1 010 2
P0	→	P0 001 1
sem erro	→	000 0

4 bits → 3 bits de paridade.

Pensamento;



$\left. \begin{array}{l} \text{nº de erros: } R+m \\ \text{sem erro : 1} \end{array} \right\} R+m+1 \leq 2^m$

n° de bits de Paridade/ n° de bits de informação
--

n° de bits de Informação	n° de bits de Paridade	Porcentagem de Redundância (%)
2	3	150
4	3	75
6	4	66,7
8	4	50,0
10	4	40,0
12	5	41,7
16	5	31,25
24	5	20,8
32	6	18,75
64	7	10,9

5.2.2. Códigos m de n

Definir palavras que tem n bits com exatamente m 1's.

→ altera

\forall erro → n° DE 1'S → M+1 OU M-1

↓
Em um único bit.

Dificuldades: processo de codificação, decodificação e detecção de erro.

Exemplo: 3 de 6 (i de 2 i) (Redundância neste caso → 100%)

Informação	Código 3 de 6	
000	000	111
001	001	110
010	010	101
011	011	100
100	100	011
101	101	010
110	110	001
111	111	000



Informação adicional

Vantagem Código Separável

Distância é 2

Este código detecta erros simples em bits e erros múltiplos unidirecionais

(1 0; ou 0 1).

Exemplo: 2 de 5 para dados BCD (não separável)

	BCD	2 de 5	
0	0000	00011	Qual a regra geral?
1	0001	11000	
2	0010	10100	
3	0011	01100	
4	0100	10010	
5	0101	01010	
6	0110	00110	
7	0111	10001	
8	1000	01001	
9	1001	00101	
	↓	↓	
	d3 d2 d1,d0	b4 b3 b1 b0	

Exemplo:

$$\begin{aligned}
 d_0 &= b_4.b_3 + b_3.b_2 + b_3.b_1 + b_4.b_\phi = b_2.b_\phi \\
 &= b_3 (b_4+b_2) + b_\phi (b_4+b_2) + b_3.b_1 \\
 &= \mathbf{(b_4+b_2).(b_3+b_\phi)+b_3.b_1}
 \end{aligned}$$

$$d_1 = b_4.(b_2+b_\phi) + b_2.(b_3+b_1)$$

5.2.3. Códigos Duplicados

Duplica o código original para formar a palavra codificada.

- . Vantagem: simplicidade
- . Desvantagem: n° de bits extra

Aplicação: sistema de memória e alguns sistemas de comunicação.

Em sistema de comunicação o conceito de duplicação é frequentemente aplicado na transmissão de toda a informação duas vezes. Se as cópias concordam, a informação é assumida estar correta.

O prejuízo é o decréscimo na taxa de informação.

Outra variação é complementar a porção duplicada da palavra codificada.

Palavra N
Palavra N

Palavra 1
Palavra 1

usado também em sistemas de comunicação que tem o mesmo meio físico para transmissão das informação

Outra variação é a duplicação (inverte e compara) “swap and compare”. Neste método são mantidas as duas cópias da informação original, mas trocam-se as metades da palavra codificada.

•	•
•	•
•	•
L2	U2
U2	L2
L1	U1
U1	L1

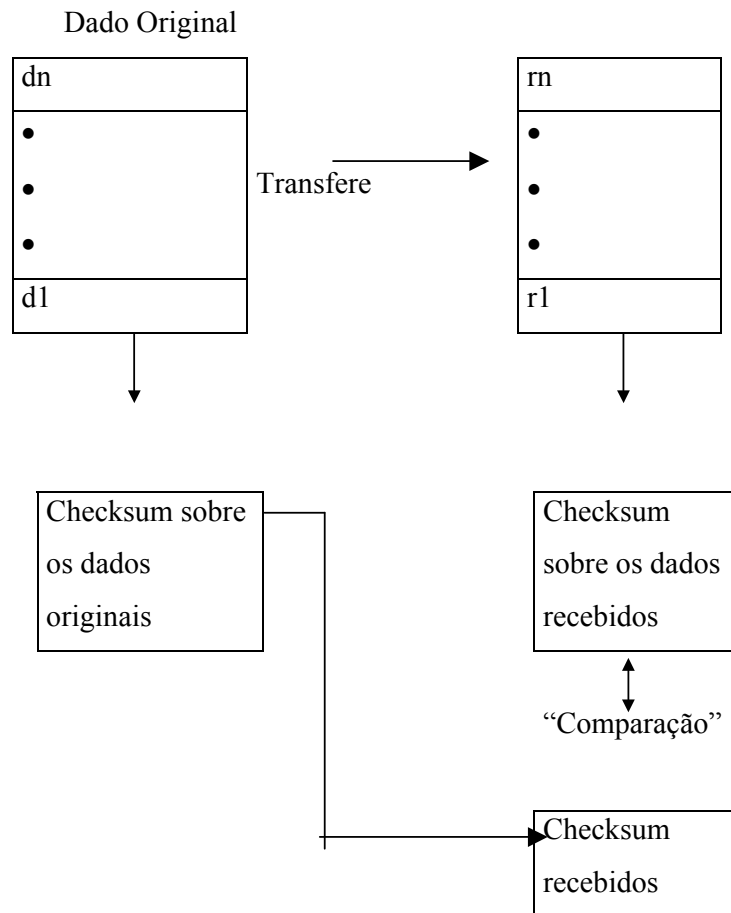
Ui - Metade Superior

Li - Metade Inferior

5.2.4. Checksums (Código Separável)

Aplicável quando blocos de dados devem ser transmitidos de um ponto para outro.

Conceito básico: Soma dos dados originais (variações na forma que a soma é feita).



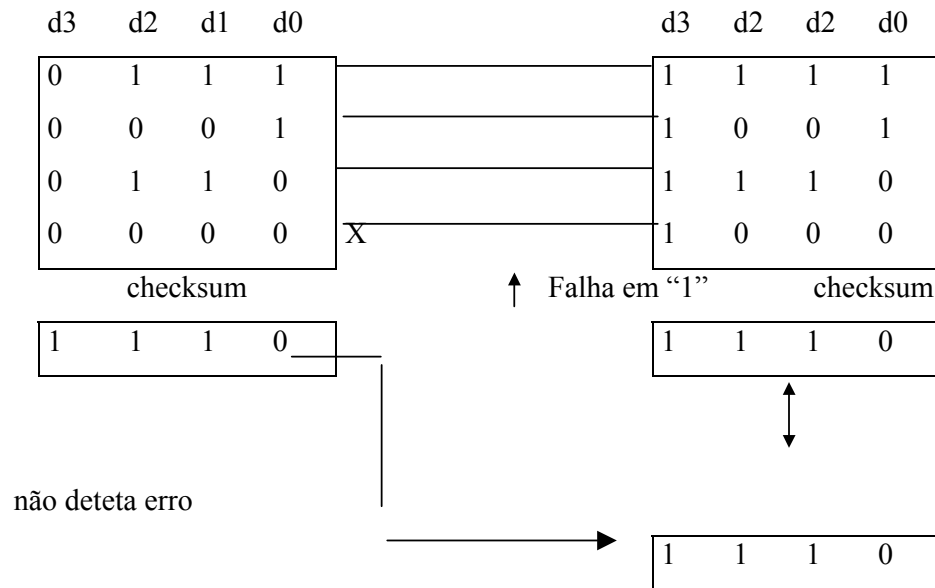
Há quatro tipos primários de checksum:

- Precisão simples;
- Precisão dupla;
- Honeywell;
- Residual.

a) Precisão Simples

Ignora o overflow do checksum.

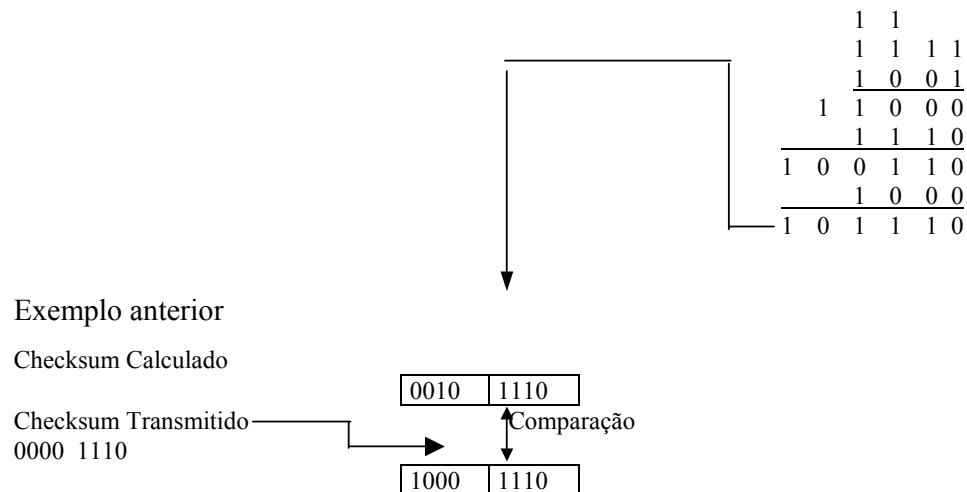
A primeira dificuldade é que se perde a habilidade de detectar erros



B) Precisão Dupla

Computar um checksum de $2n$ bits para um bloco de palavras com n bits.

Ainda é possível overflow, mas, mais difícil.

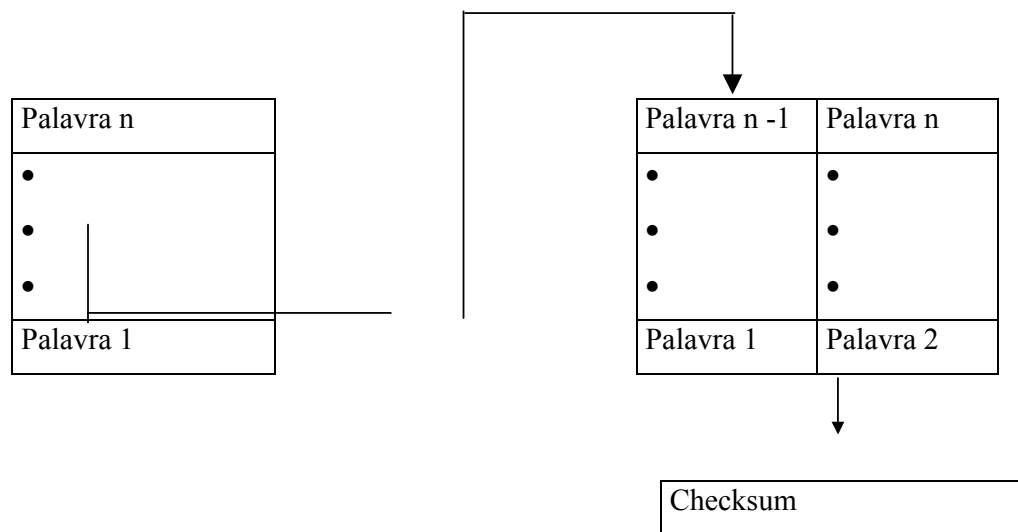


c) Honeywell

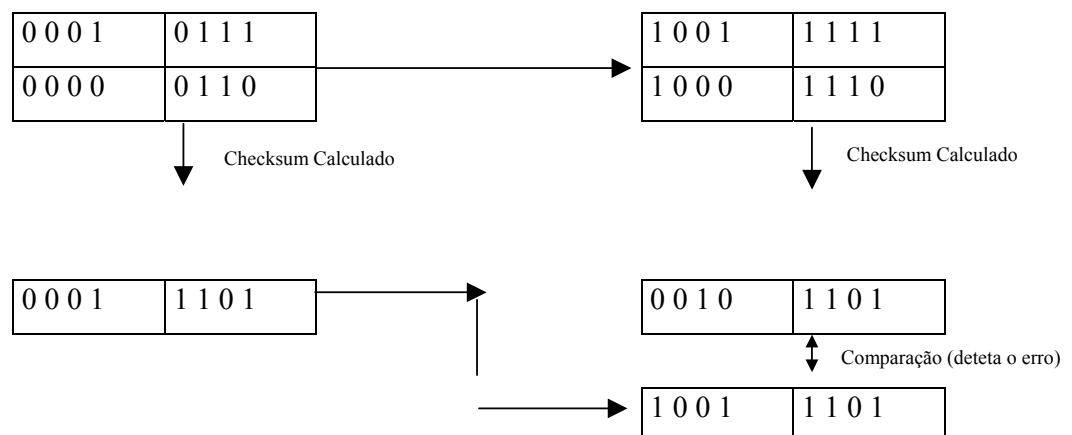
Concatena palavras consecutivas para formar uma coleção de palavras de comprimento duplo.

O Checksum é calculado sobre esta nova estrutura.

Vantagem: erro num bit de todas as palavras provocará erro pelo menos em 2 bits do Checksum.



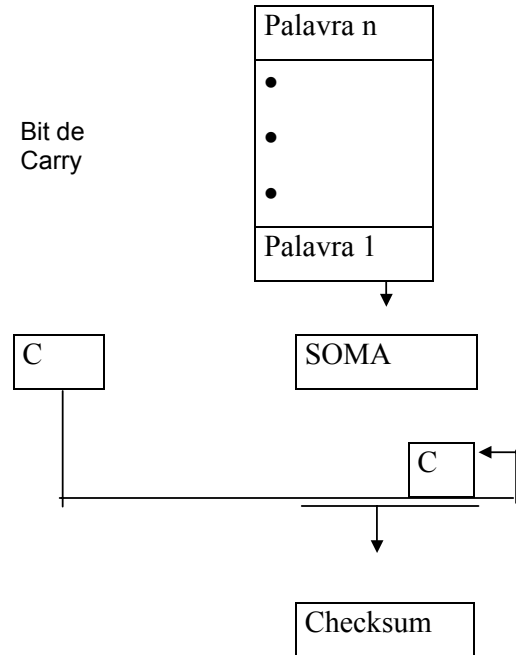
Exemplo anterior:

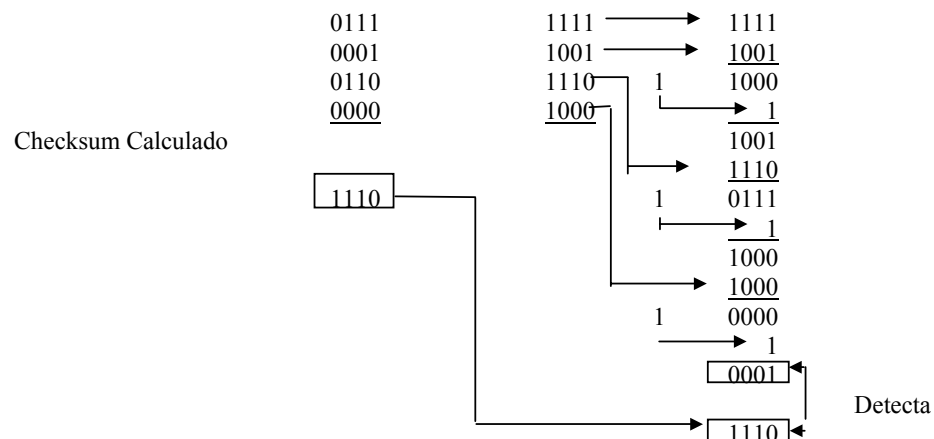


d) Residual

Mesmo conceito da precisão simples exceto que o bit carry não é ignorado, mas, é adicionado de volta ao checksum.

Exemplo:





Através do checksum não há condições para se determinar onde o erro ocorreu.

5.2.5. Códigos Cíclicos (Não Separável).

É caracterizado pelo seu polinômio gerador $G(x)$ de grau maior ou igual a $(n-R)$ onde n é o número de bits contidos na palavra codificada produzida por $G(x)$, e R é o número de bits na informação original a ser codificada.

Tais códigos têm a propriedade de serem capaz de detectar todos erros simples e múltiplos, adjacentes, que afetam menos do que $(n-R)$ bits. (Para (n,R) código $G(x)$ grau $(n-R)$).

A propriedade de detecção de erro dos códigos cíclicos é particularmente importante nas aplicações de comunicações onde erros transitórios podem ocorrer.

No código cíclico a codificação representa os coeficientes do polinômio.

Exemplo: Código = $(v_0, v_1, \dots, v_{n-1})$ corresponde ao polinômio

$$V(x) = v_0 + v_1 \cdot X + v_2 \cdot X^2 + \dots + v_{n-1} \cdot X^{n-1}. \text{ (Polinômio Codificado)}$$

$$V(X) = D(X) \cdot G(X). \text{ Polinômio de dados é } D(x).$$

Qualquer adição requerida durante a multiplicação de dois polinômios é desempenhada usando módulo-2.

Exemplo: $G(X) = 1+X+X^3$

$D(X) = 1+X+X^2+X^3$

$v(x) = ?$ 1011

1111

$v(x) = 1+X^3+X^5+X^6$

1011

1011

(1001011)

1011

↓

1011

forma de representar

1101001

Information (d_0, d_1, d_2, d_3)	Code ($v_0, v_1, v_2, v_3, v_4, v_5, v_6$)
0000	0000000
0001	0001101
0010	0011010
0011	0010111
0100	0110100
0101	0111001
0110	0101110
0111	0100011
1000	1101000
1001	1100101
1010	1110010
1011	1111111
1100	1011100
1101	1010001
1110	1000110
1111	1001011

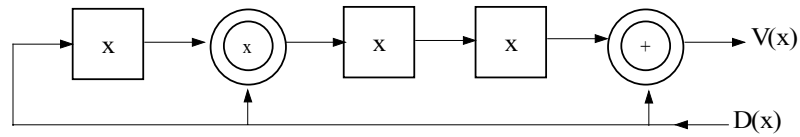
Distância de código = 3




* Qualquer erro em 2 bits é detetado *

O circulo lógico é obtido da seguinte forma:

Ex.: $G(X) = 1 + X^2 + X^3$; $D(X)$
 $V(X) = G(X) \cdot D(X) = (1 + X^2 + X^3) D(X) =$
 $(D(X) + D(X) X^2 + D(X) X^3 =$
 $(D(X) + (D(X) + D(X) X) X^2 =$



 = Multiplicação por X

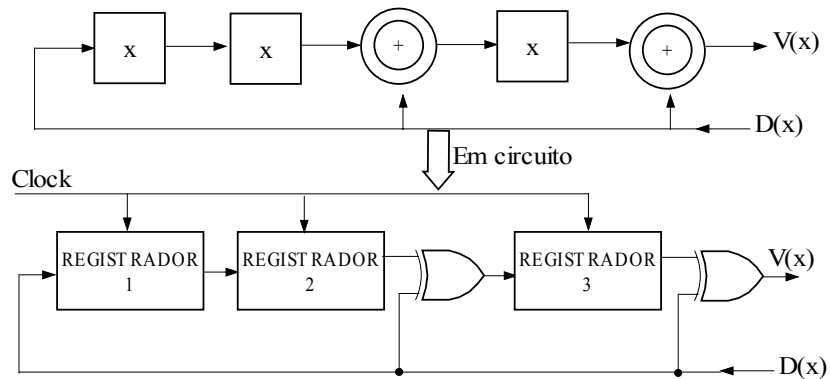
 = Adição modulo 2. (XOR-OU EXCLUSIVO)

Exemplo:

$$G(x) = 1 + X + X^3$$

$D(x)$

$$\begin{aligned} V(x) &= D(x) + (1 + X + X^3) \\ &= D(x) + (D(x) \cdot X + D(x) \cdot X^3) \\ &= D(x) + (D(x) + D(x) \cdot x^2) \cdot X \end{aligned}$$



Processo de Codificação Registradores: $(D(X)=(1101))$

Registadores

Clock	1	2	3	D(X)	V(X)
0	0	0	0	1	1
1	1	0	1	1	0
2	1	1	1	0	1
3	0	1	1	1	0
4	1	0	0	0	0
5	0	1	0	0	0
6	0	0	1	0	1
7	0	0	0	0	0



1011

1011

1011

1011

1011

1000101

Processo de Decodificação

Se $R(X)$ é um código válido recebido então:

$$R(X) = D(X) \cdot G(X)$$

Podemos escrever que

\mapsto Síndrome

$$R(X) = D(X) \cdot G(X) + S(X)$$

e que $S(X)$ deve ser zero se o polinômio $R(X)$ é um código válido.

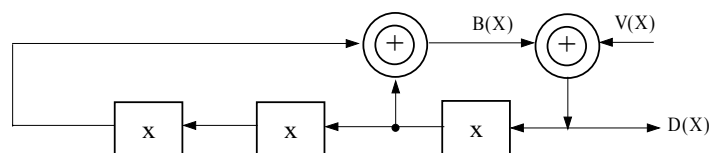
$$V(X) = G(X) \cdot D(X)$$

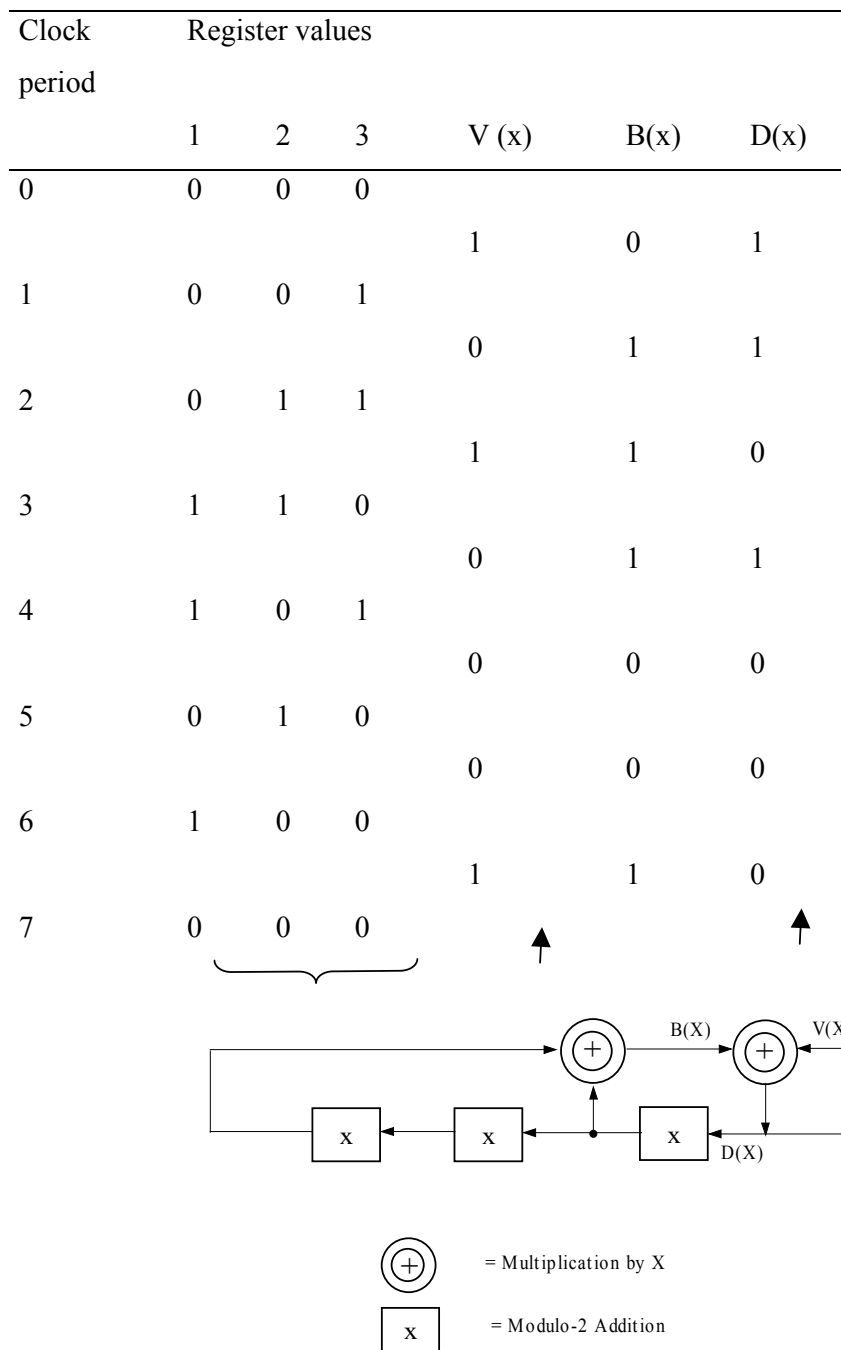
Exemplo: $G(X) = 1 + X + X^3$
 $D(X) = V(X) + B(X)$
 $V(X) = (1 + X + X^3) \cdot D(X) + S(X)$
 $V(X) = D(X) + D(X)(X + X^3)$

Em módulo-2 Soma e Subtração são iguais

- $D(X) = V(X) + D(X)(X + X^3)$
 $B(X) = D(X)(X + X^3)$
 $= D(X) \cdot X + D(X) \cdot X^3$

Circuito:





5.2.6. Códigos Aritméticos.

São úteis para verificar operações aritméticas (+, *,).

Propriedade: $A(b*c) = A(b) * A(c)$.

Exemplos de códigos aritméticos: AN, Residual, Inverso - residual e Sistema de Números Residuais.

a) Códigos AN

O mais simples código aritmético que é formado multiplicando cada dado por uma constante A.

Soma: $A \cdot 2^a$ (Não pode ser potência de 2).

No lado da recepção verifica se o código é divisível por A.

$(a_{n-1} a_{n-2} \dots a_2 a_1 a_0)$.

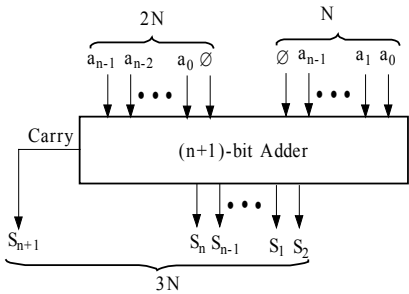
$$a_{n-1} \cdot 2^{a+n-1} + \dots + a_2 \cdot 2^{a+2} + a_1 \cdot 2^{a+1} + a_0 \cdot 2^a + 0 \cdot 2^{a-1} + \dots + 0 \cdot 2^0.$$



mudando este bit continua sendo divisível por 2^a

Exemplo: $3N$ código = $(N+2N)$

Original Information	3/V code word
0000	000000
0001	000011
0010	000110
0011	001001
0100	001100
0101	001111
0110	010010
0111	010101
1000	011000
1001	011011
1010	011110
1011	100001
1100	100100
1101	100111
1110	101010
1111	101101



b) Códigos Residuais (Separável)

Adiciona um número residual ao número.

O número residual é simplesmente o resto quando o número é dividido por um inteiro.

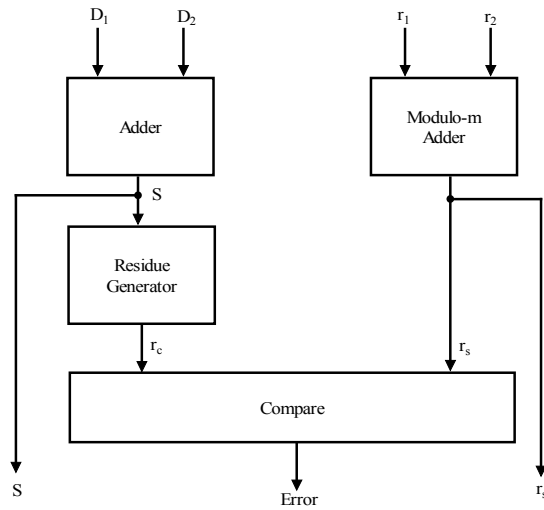
Ex.: Inteiro N, inteiro m

$$\begin{cases} N = Im + r \\ 0 \leq r < m. \end{cases}$$

base (módulo)

Information	Residue	code word	
0000	0	0000	00
0001	1	0001	01
0010	2	0010	10
0011	0	0011	00
0100	1	0100	01
0101	2	0101	10
0110	0	0110	00
0111	1	0111	01
1000	2	1000	10
1001	0	1001	00
1010	1	1010	01
1011	2	1011	10
1100	0	1100	00
1101	1	1101	01
1110	2	1110	10
1111	0	1111	00

Processo de detecção de erro na recepção.



5.2.7. Códigos Berger

Adicionar um conjunto de bits verificadores, caracterizando um código separável.

O comprimento é $_N$ e tem $_I$ bits de informação e $_R$ bits de verificação.

$$R = \log_2(I+1) \text{ e } n = I + R$$

Exemplo:

(0111010) ← Informação $I=7$

$$R = \log_2(7+1) = 3 \longrightarrow \Downarrow$$

O no. de "1"s é 4 → (100)

complemento
 \Downarrow

(011)

código resultante
 \Downarrow
 (0111010011)

Quando o n de bits da Informação é baixo, a redundância é alta.

N de bits de Informação (I)	N de bits de verificação (R)	%
4	3	75
8	4	50,00
16	5	31,25
32	6	18,75
64	7	10,94

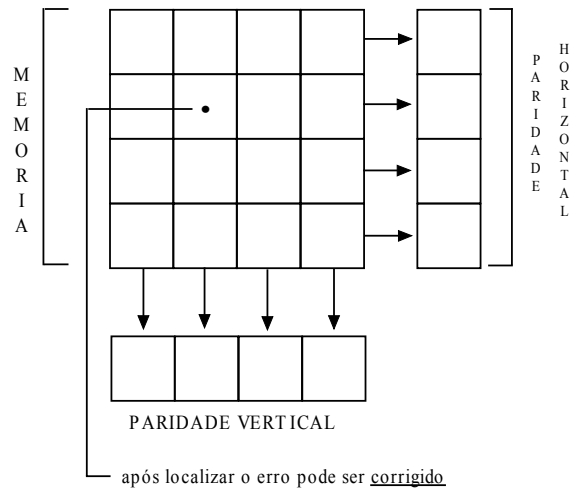
Original Information	Berger code	
0000	0000	111
0001	0001	110
0010	0010	110
0011	0011	101
0100	0100	110
0101	0101	101
0110	0110	101
0111	0111	100
1000	1000	110
1001	1001	101
1010	1010	101
1011	1011	100
1100	1100	101
1101	1101	100
1110	1110	100
1111	1111	011

Exemplo: I=4 e R=3

1011 → no. de “1” e = 3 → (011)

(1011100) ← (100)

5.2.8. Paridade Horizontal e Vertical



Quando o erro é múltiplo, é possível detectar, mas não corrigir o erro.

5.2.9. Código de Correção de Erro Hamming

Requer de 10 a 40% de redundância. A codificação e decodificação acrescentam um atraso relativamente pequeno.

Este código utiliza c bits de verificação de paridade para proteger R bits de informação.

Relação: $2^c \geq c + R + 1$

Comprimento total: $n = c + R$

O código Hamming é formado particionando os bits de informação em grupos de paridade e especificando um bit de paridade para cada grupo. (par ou ímpar).

A habilidade de localizar qual bit é errado é obtida através de superposição dos grupos de bits.

Exemplo:

bits de informação: ($d_3 d_2 d_1 d_0$)

$$2^c \geq c + 4 + 1 = C+5$$

$$2^c \geq c + 5 \rightarrow C = 3 (C_1 C_2 C_3)$$

Bit errado	bit de Verificação Afetado
d0	C1, C2
d1	C1, C3
d2	C2, C3
d3	C1, C2, C3
C1	C1
C2	C2
C3	C3

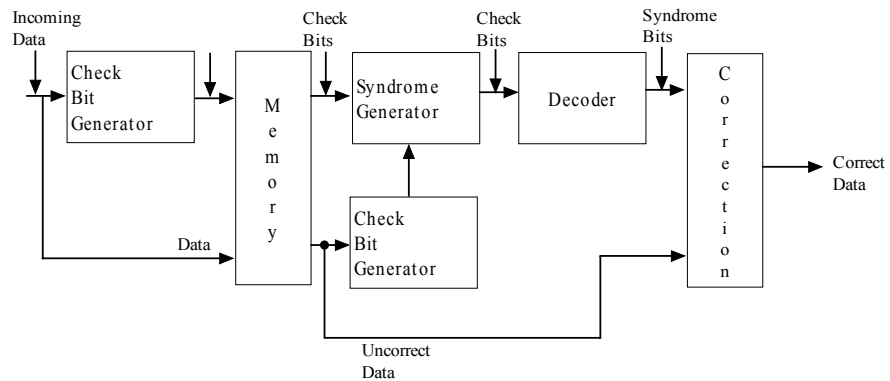
Na recepção calcula-se o código de verificação e compara com o recebido.

É definido SINDROME como a comparação entre o código calculado e o recebido (XOR).

Se for igual a “1” indica incorreção.

(Um Bit errado)	C1, C2, C1, C3
	Sindrome
d0	110
d1	101
d2	011
d3	111
c1	100
c2	010
c3	001

Esquema geral utilização



Para 2 bits errados acrescenta-se mais um bit de paridade da palavra. (Parid. Global).

Síndrome	Paridade Global	correção
não zero	incorreto	um bit errado → correção
não zero	correto	Dois bits errados →
zero	correto	Correto

5.2.10 Circuitos Integrados com Correção de Erro

Muitos circuitos integrados estão disponíveis para detectar, localizar e corrigir erros. Como exemplo, pode-se citar o 8206 da Intel, MC68540 da Motorola, AM2960 e AMZ8160 da Advanced Micro Devices, DP8400 da National Semiconductor e MB1412A da Fijitsu.

Estes circuitos integrados são projetados para gerarem os bits de verificação, gerar a “síndrome”, decodificar a “síndrome” e realizar a correção dos dados. Unidades típicas são organizadas para suportar dados de 16 bits, mas podem ser expansíveis para detecção e correção de palavras de tamanho maior.

Quase todos os CI's usam o Código de Hamming modificado para detectar erros duplos e corrigir erros simples.

O Diagrama em blocos da estrutura fundamental de um Circuito Integrado deste tipo é mostrada a seguir. Uma palavra vindo do sistema vai diretamente à memória e a um gerador de bits de verificação. Ambos são armazenados na memória. Na leitura de uma palavra da memória ocorrem os seguintes eventos:

1. O dado é usado para regenerar os bits de verificação através de um Gerador de Bits de Verificação.
2. É criada a “síndrome” comparando os bits de verificação original com os regenerados.
3. A “Síndrome” é usada para identificar o bit errado.
4. Os dados são passados através de uma unidade de correção.
5. Os dados corrigidos são colocados no bus.

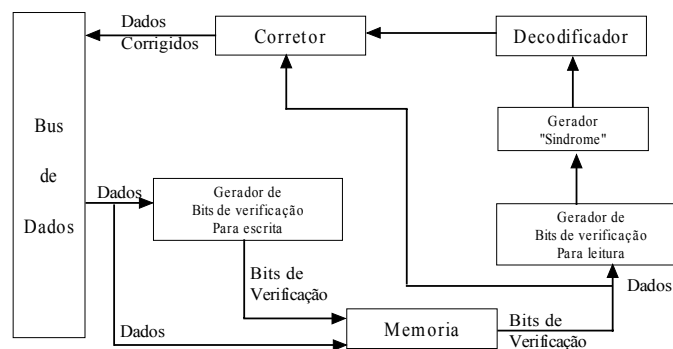
O número de ocorrência de erros é usado para indicar um defeito permanente na memória.

Para Intel 8206: (16 bits)

- tempo de detecção do erro: $\leq 52\text{ns}$
- tempo de detecção e correção: $\leq 67\text{ns}$.

Para efeito de rapidez, em códigos separáveis, a informação já pode ser usada enquanto se realizam as verificações necessárias.

Circuito Integrado de Correção de Erro



5.3. Redundância por Tempo.

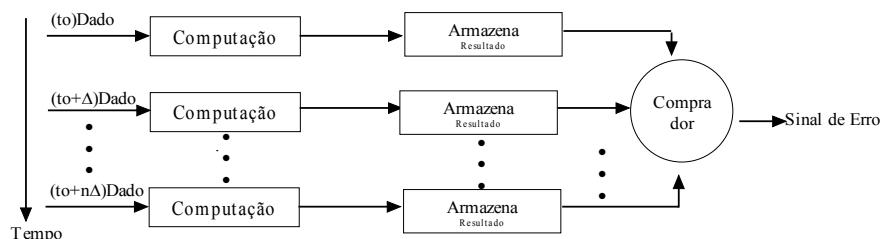
Tanto a redundância de hardware como a de informação, requer um hardware extra na sua implementação.

A Redundância por tempo, por sua vez, não requer hardware extra.

A escolha adequada deve ser tomada em função dos requisitos de cada aplicação.

5.3.1. Detecção de Defeito Transiente.

O conceito básico da redundância temporal é a repetição da computação de forma a permitir que o defeito seja detectado. A forma básica da redundância temporal é apresentada a seguir:



Se um erro é detectado, a computação pode ser realizada novamente para verificar se a discordância continua ou desaparece. Tal abordagem é interessante para detecção de erros resultantes de defeito transitórios, mas não protege contra erros de defeitos permanentes.

Pode-se incrementar o sistema de forma que quando o erro for detectado, existirem duas condições a serem analisadas:

- Um defeito permanente produziu o erro, e aquela parte do sistema deve ser isolada (reconfigurada);
- Um defeito transiente aconteceu e produziu o erro, mas o hardware continua sendo utilizável, sem necessidade de isolar parte do sistema.

Isto pode ser realizado em função do número de computações repetidas após a detecção do primeiro erro. (Depende dos requisitos do sistema).

5.3.2. Detecção do Defeito Permanente.

Um dos grandes potenciais da técnica da redundância temporal é a habilidade de detectar defeitos permanentes usando um mínimo de hardware extra.

Quatro abordagens são consideradas:

- * Lógica Alternada;
- * Recomputação com Operandos Deslocados;

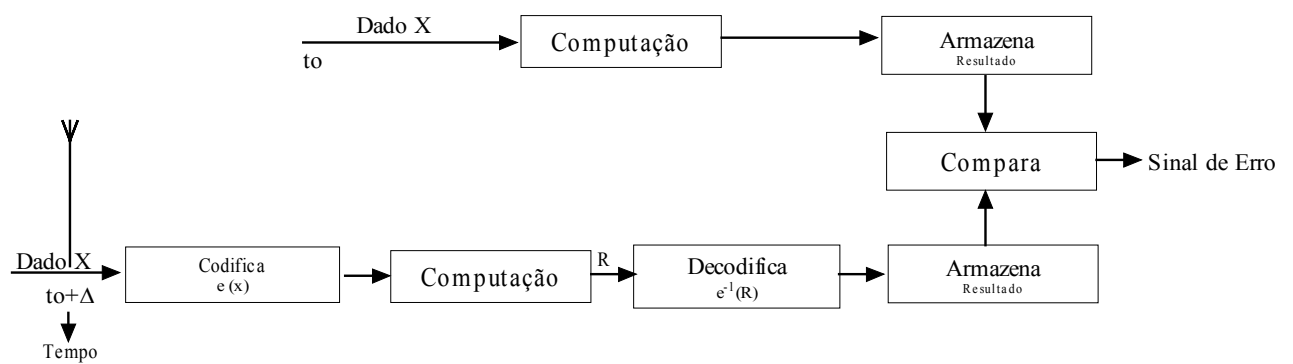
(Recomputing with Shifted Operands - Reso);

* Recomputing with Swapped Operands (RESWO)

Recomputação com Operandos trocados

* Recomputação com Duplicação com Comparação. Recomputing with duplication with comparison (REDWC).

O conceito fundamental por trás de cada abordagem é apresentado a seguir:

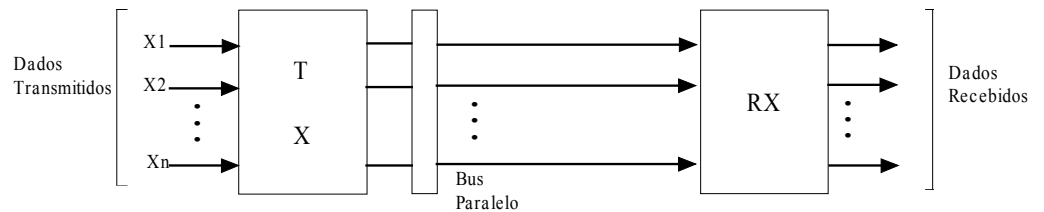


A seleção da função de codificação $e(x)$ é feita para permitir que defeitos no hardware sejam detectados.

5.3.2.1. Lógica Alternada

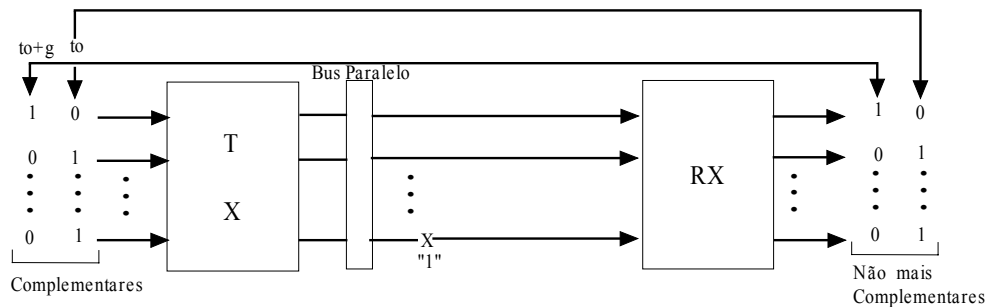
Aplicada na transmissão de dados digitais em cabos e detecção de defeitos em circuitos digitais.

Seja a figura a seguir, de transmissão de dados sobre uma via paralela.



No instante t_0 transmite-se os dados originais e no instante $t_0 + g$ transmite-se os dados complementados.

Suponha um defeito numa linha, grampeando-a em nível “1”.



O Defeito pode ser detectado, alternando a lógica:

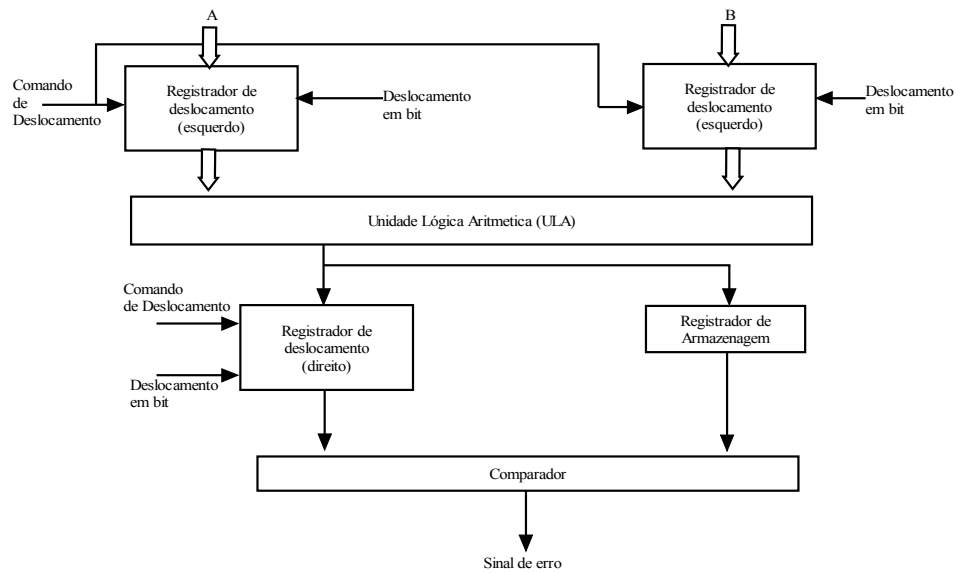
O conceito da Lógica Alternada pode ser aplicado, em geral, a circuitos lógicos que tem a propriedade de “self-duality”, ou seja:

$$f(x) = -f(-x)$$

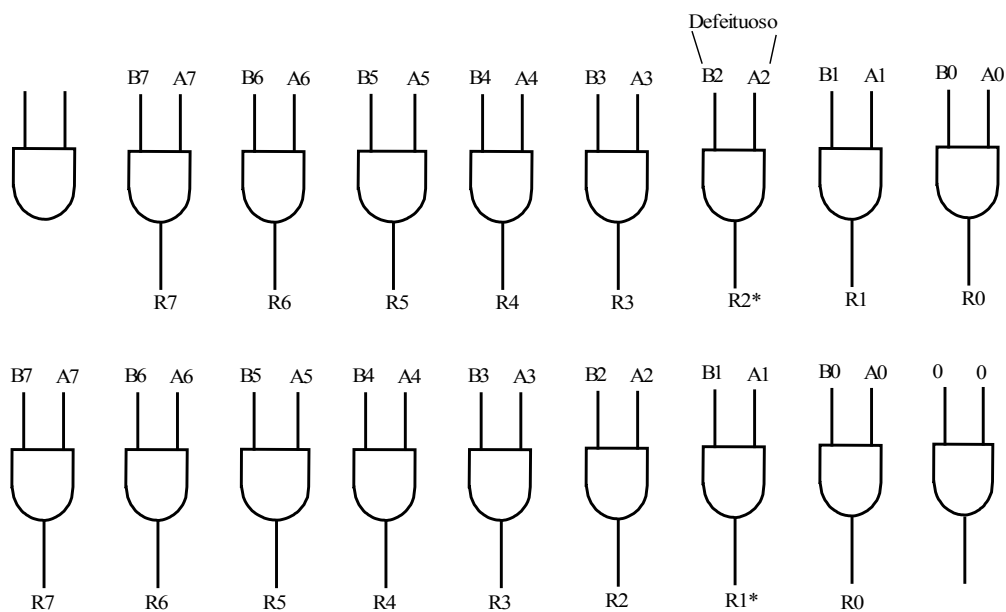
5.3.2.2. Recomputação com Operandos Deslocados

“Recomputing with Shifted Operands-RESO”. Utilizado como método de fornecer detecção de erro em ULA.

A função de Codificação é selecionada como operação de deslocamento para a esquerda e a de decodificação para a direita. (deslocamento lógico ou aritmético).



Como exemplo, considere a operação lógica AND realizada sobre 8 operandos, como mostrado a seguir. Se um “bit slice” está como defeito e não tem efeito sobre os demais, um único deslocamento para à esquerda detecta o erro que ocorre na operação lógica.



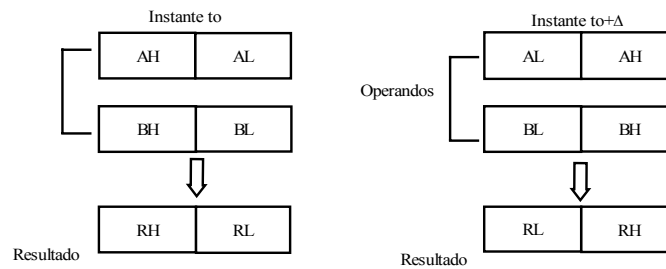
{	Comparação dos	R7	R6	R5	R4	R3	R2*	R1	R0
	Resultados	R7	R6	R5	R4	R3	R2	R1*	R0

A estrutura de ULA que usa esta técnica é apresentada a seguir. O Hardware adicional são 3 Deslocadores, um Registrador para armazenar o resultado da primeira computação e um comparador.

O problemas primários com esta arquitetura são o hardware adicional necessário, a falta de cobertura para defeitos nos deslocadores, e o requisito que o comparador seja totalmente auto-verificável de maneira que defeitos no comparador não comprometa a eficiência da abordagem.

5.3.2.3. Recomputação com Operando Trocados

RESWO - “Recomputing with Swapped Operands”. O conceito básico desta arquitetura é mostrado a seguir:



Durante a Segunda computação, a metade superior e inferior são trocadas de forma que o bit slice com defeito atua nas duas metades durante a primeira e a Segunda computação.

A estrutura da ULA projetada para acomodar a arquitetura RESWO é

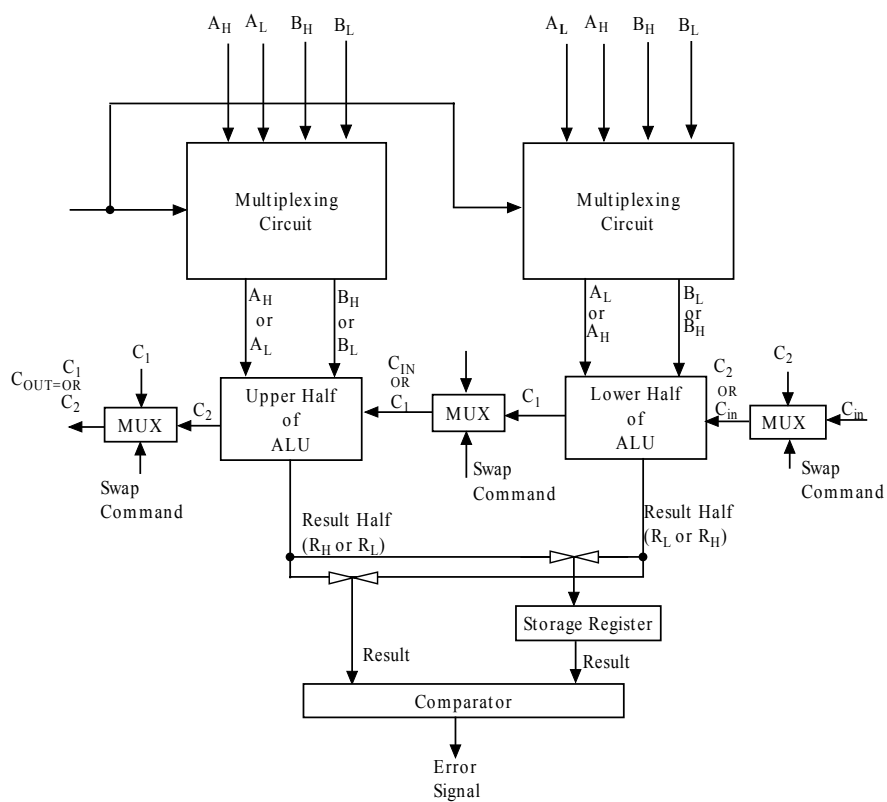
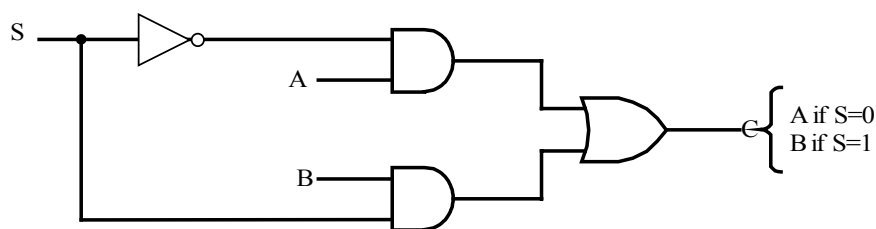


Figura 5.64

mostrada a seguir:

O multiplex é um circuito relativamente simples apresentado a seguir:



S = Swap Comand

A } Inputs
B }

Figura 5.65

5.3.2.4. Recomputação com Duplicação e Comparação “Recomputing with Duplication with Compararison - REDWC”

A Redundância no tempo é utilizado para completar o cálculo e obter o resultado final.

Esta arquitetura, no caso do somador, realiza duas computações. Na primeira as metades inferiores dos operandos são somadas nas duas metades do somador. Os resultados são então comparados e um resultado é armazenado para representar a metade inferior.

A Segunda computação realiza o mesmo para as metades superiores.

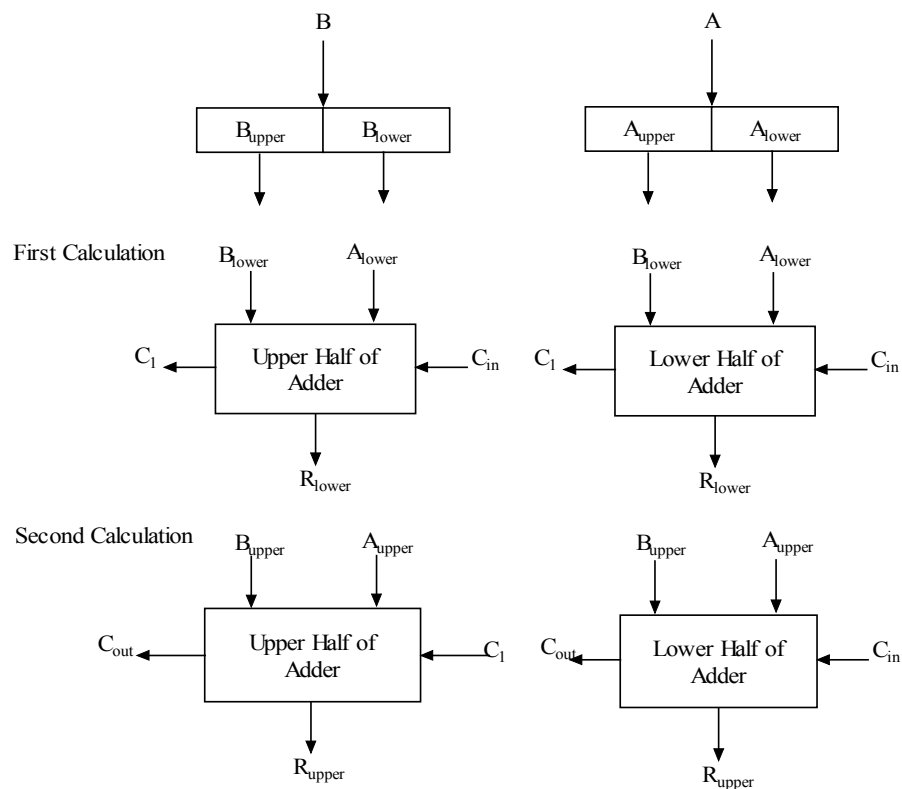


Figura 5.66

O diagrama em bloco desta arquitetura é apresentado na figura que se segue:

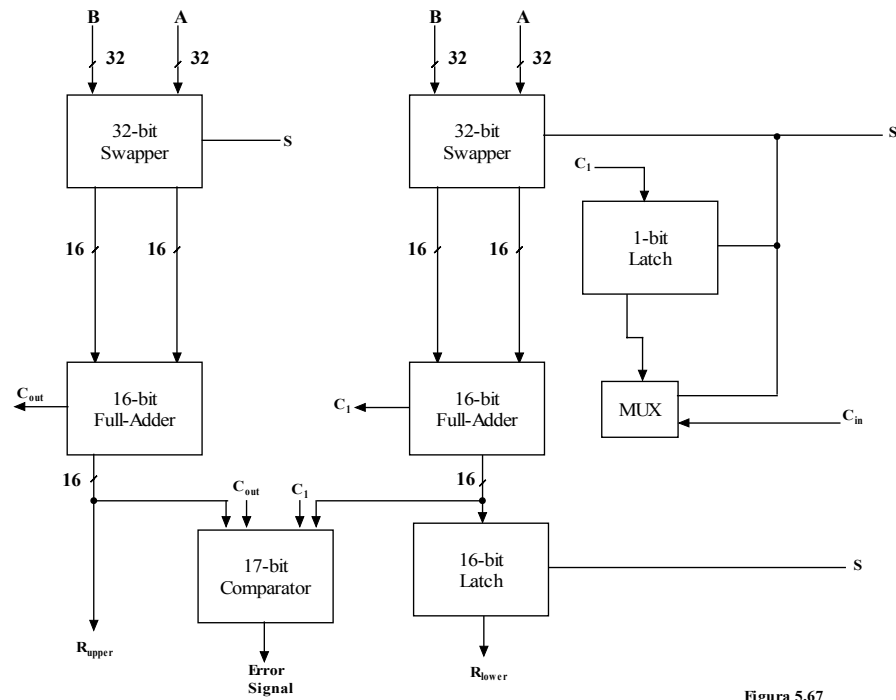


Figura 5.67

Figura 5.67

5.3.2.5. Recomputação para Correção de Erro

A Redundância pro tempo pode ser utilizada também para correção de erro se repetida 3 ou mais vezes.

Considere a operação lógica AND ilustrada na figura que se segue:

Perform
Operation on
Unshifted
Operands

0	0	a ₇	a ₆	a ₅	a ₄	a ₃	a ₂	a ₁	a ₀
0	0	b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀

0	0	r ₇	r ₆	r ₅	r ₄	r ₃	r ₂	r ₁	r ₀
---	---	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

Repeat
Operation
on Operands
After a 1-Bit

0	a ₇	a ₆	a ₅	a ₄	a ₃	a ₂	a ₁	a ₀	0
0	b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	0

0	r ₇	r ₆	r ₅	r ₄	r ₃	r ₂	r ₁	r ₀	0
---	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	---

Repeat
Operation
on Operands
After a 2-bit

a ₇	a ₆	a ₅	a ₄	a ₃	a ₂	a ₁	a ₀	0	0
b ₇	b ₆	b ₅	b ₄	b ₃	b ₂	b ₁	b ₀	0	0

r ₇	r ₆	r ₅	r ₄	r ₃	r ₂	r ₁	r ₀	0	0
----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------	---	---

Faulty
Bit

Perform
Bit-by-Bit
Vote to Correct
Erroneous Bits

0	0	r ₇	r ₆	r ₅	r ₄	r ₃	r ₂	r ₁	r ₀
0	0	r ₇	r ₆	r ₅	r ₄	r ₃	r ₂	r ₁	r ₀
0	0	r ₇	r ₆	r ₅	r ₄	r ₃	r ₂	r ₁	r ₀



Corrected Result

0	0	r ₇	r ₆	r ₅	r ₄	r ₃	r ₂	r ₁	r ₀
---	---	----------------	----------------	----------------	----------------	----------------	----------------	----------------	----------------

Suponha que a operação é realizada três vezes: a primeira, sem deslocamento dos operandos; a Segunda com o deslocamento de 1 bit dos operandos; a terceira, com deslocamento de 2 bits dos operandos.

Desta forma, um bit diferente no resultado será afetado pelo defeito no “bit slice”. Se os bits em cada posição são comparados, o resultado devido ao defeito, pode ser corrigido realizando-se uma votação majoritária.

Esta solução não funciona para operações aritméticas, pois os bits adjacentes não são independentes. Um defeito em um “bit slice” pode afetar mais de um bit no resultado final.

5.4. Redundância por Software

Em aplicações que utilizam computadores, muitos meios de detecção de defeitos e técnica de tolerância a defeitos podem ser implementados por software. O hardware redundante necessário para implementar esta capacidade pode ser mínimo, mas o software redundante é bastante substancial.

A Redundância de Software pode aparecer como muitas linhas extras do código usadas para verificar a magnitude de um sinal ou uma pequena rotina usada para testar periodicamente a memória (escrevendo e lendo em posições específicas).

São consideradas aqui, três técnicas de redundância de software:

- Verificação de Consistência;
- Verificação de Capacidade;
- N_ Versões de software.

5.4.1. Verificação de Consistência

A verificação de consistência usa o conhecimento a priori sobre as características da informação a ser verificada a correção. Por exemplo, em algumas aplicações, é conhecido que uma grandeza nunca deve exceder uma certa magnitude. Se o sinal exceder esta magnitude, indica a presença de algum erro.

A verificação da consistência pode ser implementada em hardware, mas é mais realizada em software.

Um exemplo de verificação de consistência que pode ser realizado em hardware é a detecção de códigos de instrução inválidos em computadores. Muitos computadores usam n bits para representar 2^k instruções possíveis, onde $K < n$. Em outras palavras, há $2^n - 2^k$ instruções possíveis, ilegais. Cada instrução pode ser verificada que não se trata de um código ilegal. Se um código ilegal ocorre, o processador pode ser “parado” evitando operação errada de ocorrer. Esta técnica detecta erro no processador de interpretar dados como instrução.

Outro exemplo é na transferência de pacotes de dados. No início de cada pacote pode haver um “header” que contém o número de palavras naquele pacote.

Na recepção é detectado um desacordo entre o número de palavras realmente recebidas e o número especificado no “header”.

5.4.2. Verificação de Capacidade

Esta verificação constata ou não que o sistema possui uma capacidade esperada.

Um primeiro exemplo é o teste de memória simples. O processador pode simplesmente escrever padrões específicos em posições de memória e lê-los para verificar que o dado armazenado é recuperado. Este teste pode ser um complemento ao bit de paridade contra defeitos na memória.

Outro teste é da ULA. Periodicamente o processador executa certas instruções em dados específicos e compara os resultados com resultados esperados armazenados em ROM.

Este tipo de teste, verifica tanto a ULA como a memória em que os resultados foram armazenados. As instruções executadas podem consistir de adição, multiplicação, operações lógicas e transferência de dados.

Outro tipo desta verificação consiste na verificação se todos os processadores estão se comunicando corretamente. Neste método são enviadas periodicamente informações específicas de um processador para outro.

5.4.3. N_ Versões de Software

Defeitos no software são conceitos não usuais. Técnicas para detectar estes defeitos devem detectar desvios no projeto. Uma simples duplicação e procedimento de comparação não detectam defeitos no software se os módulos de software duplicados são idênticos.

O conceito básico em N_ Versões de software é projetar e codificar o software N vezes e comparar os N resultados produzidos por cada um destes módulos. Cada um desses módulos é projetado e codificado por um grupo separado de programadores. Cada grupo projeta o software a partir de uma mesma especificação, de maneira que cada módulo desempenha a mesma função.

É esperado que N projetos independentes não irão gerar erros iguais.

Entretanto quando ocorre um erro, não ocorre em todos os módulos, ou ocorre de maneira diferente em cada módulo, de forma que os resultados finais serão diferentes.

Existem certas polêmicas com relação à N_ Versões de software.

A primeira é que os projetistas e codificadores tendem a fazer erros similares. Além disso, não se garante que versões independentes de um programa não terão defeitos idênticos.

Finalizando, como as N versões se originam a partir de uma mesma especificação, não se detecta erros na especificação.

Se o software é desenvolvido corretamente, não há a necessidade de se utilizar técnicas de tolerância a defeitos de software.

6. Técnicas de Avaliação de Sistemas Tolerantes a Defeito

- Taxa de Falhas
- Tempo médio para Falhar (MTTF)
- Tempo médio entre Falhas (MTBF)
- Cobertura de Defeitos
- Análise de Confiabilidade
- Análise de Segurança
- Análise de Disponibilidade
- Análise de Manutenibilidade
- Taxas de Redundância
- Custos

6.1. Função Confiabilidade ($R(t)$) e Taxa de Falhas

Para se definir a Função Confiabilidade $R(t)$ é necessário conceituar o termo taxa de falhas. Intuitivamente a taxa de falha é o número esperado de falhas de um tipo de sistema num determinado período de tempo.

Como exemplo, pode-se considerar que um computador falha, em média, uma vez a cada 2000 horas. Neste caso o computador apresentará uma taxa de falhas de 1/2000 falhas/hora.

A taxa de falha é denotada pela letra grega λ .

Quando algum tipo de redundância for incorporado como meio de alcançar tolerância a defeito, a taxa de falhas desse novo sistema redundante deve ser menor do que a taxa da falha do sistema similar, não redundante.

Para podermos representar formalmente o conceito de Confiabilidade pode-se seguir o seguinte raciocínio.

Vamos utilizar N componentes idênticos.

Suponha que estão todos operacionais no instante t_0 e que se registre o número de componentes falhos e componentes corretos no instante t .

$N_f(t)$: Número de componentes que falharam até o instante t .

$N_o(t)$: Número de componentes operacionais até o instante t .

A confiabilidade $R(t)$ neste exemplo pode ser calculada como a divisão entre o número de componentes operacionais, dividido pelo número total de componentes.

$$R(t) = \frac{N_o(t)}{N} = \frac{N_o(t)}{N_o(t) + N_f(t)}$$

Esse valor corresponde a probabilidade de um componente funcionar no intervalo entre $[t_o, t]$.

Por outro lado, a Não-Confiabilidade $Q(t)$ de um componente funcionar no intervalo $[t_o, t]$ pode ser calculada como:

$$Q(t) = \frac{N_f(t)}{N} = \frac{N_f(t)}{N_o(t) + N_f(t)}$$

Desta forma:

$$R(t) = 1 - Q(t) = 1 - \frac{N_f(t)}{N}$$

$$\frac{dR(t)}{dt} = -\frac{1}{N} \cdot \frac{dN_f(t)}{dt}$$

$$\frac{dN_f(t)}{dt} = -N \cdot \frac{dR(t)}{dt}, \text{ considerada como taxa de falha instantânea.}$$

A Função Taxa de Falha $\lambda(t)$ ou “*Hazard Function*” ou “*Hazard Rate*” é definida como:

$$\lambda(t) = \frac{\frac{dN_f(t)}{dt}}{N_o(t)} \quad (\text{falhas por unidade de tempo})$$

$$\lambda(t) = \frac{1}{N_o(t)} \cdot \left(-N \cdot \frac{dR(t)}{dt} \right) = -\frac{N}{N_o(t)} \cdot \frac{dR(t)}{dt}$$

$$\lambda(t) = -\frac{\frac{dR(t)}{dt}}{R(t)} = \frac{\frac{dQ(t)}{dt}}{1-Q(t)}$$

Sendo que $f(t) = \frac{dQ(t)}{dt}$ correspondente à função densidade de falha.

A função taxa de falhas $\lambda(t)$ depende do tempo, e verifica-se experimentalmente que para componentes eletrônicos e computacionais existe um período em que $\lambda(t)$ é aproximadamente constante. Este fato está relacionado com a denominada Curva da Banheira, conforme mostra a figura 6.1.

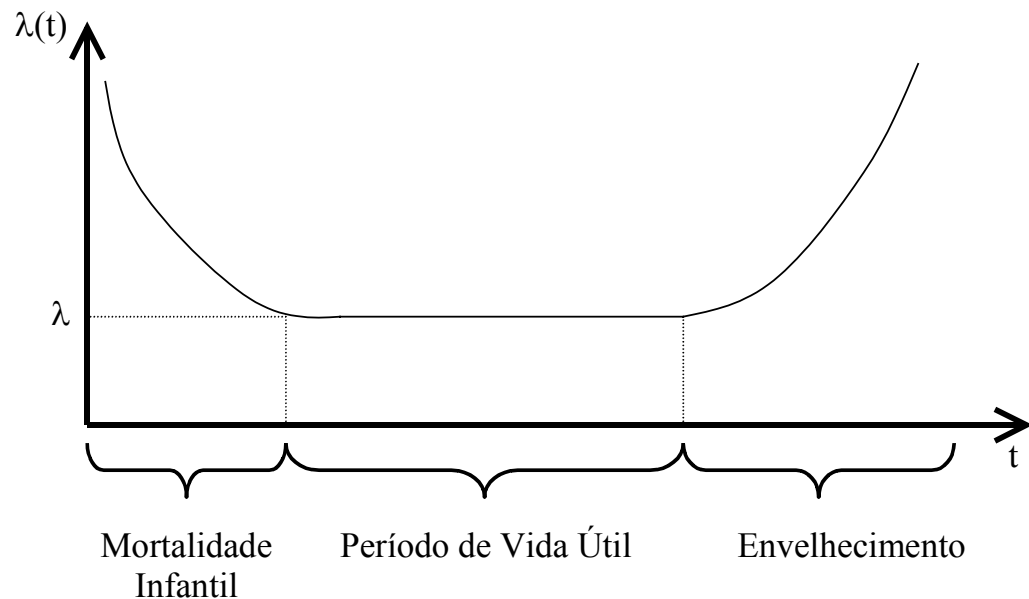


Figura 6.1 – Curva da Banheira

A região de Mortalidade Infantil corresponde à região em que aqueles componentes produzidos com algum defeito constitucional falham. Para se eliminar esses componentes com uma maior rapidez, pode-se utilizar técnicas de Teste de Burn-in, onde os componentes são submetidos a um stress maior do que aquele em que é submetido quando em operação normal. O Período de Vida Útil corresponde ao período de tempo em que a taxa de falhas do

componente se mantém relativamente estável. O período de envelhecimento corresponde ao período em que a taxa de falhas do componente começa a crescer vertiginosamente, aumentando drasticamente a probabilidade do componente falhar, caso não tenha ainda falhado.

Já havíamos concluído que:

$$\lambda(t) = -\frac{\frac{dR(t)}{dt}}{R(t)}$$

$$\frac{dR(t)}{dt} = -\lambda(t).R(t)$$

Se admitirmos que o sistema está no período de vida útil, a função taxa de falha tem um valor constante, denominado λ .

$$\frac{dR(t)}{dt} = -\lambda.R(t)$$

A solução para essa equação é:

$$R(t) = e^{-\lambda.t}$$

que corresponde a uma curva de distribuição exponencial. Vale ressaltar que essa expressão só é válida para componentes eletrônicos e computacionais.

Em certas aplicações, entretanto, não se pode assumir que a função de taxa de falha apresenta valor constante. Entre estas aplicações pode-se citar sistemas mecânicos, pneumáticos, hidráulicos e o próprio componente Software. Nestes casos outros modelos devem ser empregados.

6.2. Cálculo da Taxa de Falhas

Um aspecto importante na análise de sistema é a estimativa de taxa de falhas de componentes específicos. A técnica mais comum é do Departamento de Defesa Americano (USDOD) cuja norma MIL - HDBK - 217 contém esses dados a partir de valores experimentais.

Apresenta - se a seguir alguns parâmetros.

A medição de falha de um CI é dada por:

$\lambda = \pi L \cdot \pi Q (C1 \cdot \pi_t + C2 \cdot \pi_E) \cdot \pi_p$ falhas por milhões de horas.

πL ... Fator de aprendizado: representa a maturidade do processo do fabricante utilizado na produção do CI (Varia de 1 a 1ϕ)

πQ ... Fator de qualidade: representa o conjunto de teste que o componente sofre antes de ser vendido por um fornecedor (1 a $3\phi\phi$)

Existem 4 níveis básicos: A, B, C, D.

Classes A e B => aplicações militares

A => $\pi Q = 1$

B => $\pi Q = 2$

Classe C => componentes comerciais de alta qualidade.

$\pi Q = 16$

Classe D => componentes comerciais hermeticamente selados.

$\pi Q = 150$

πT ... Fator de temperatura. É função da tecnologia, temperatura de operação, tecnologia de empacotamento e dissipação de potência. (0, 1 a 1000).

πE ... Fator ambiental. Baseado no ambiente operacional.

Ex.: Componentes em sala de computador - 0, 2

componentes em mísseis - 10, 0

πp ... Função do No de pinos do CI.

1, 0 => No de pinos ≤ 25

1, 1 => $26 < n < 64$

1, 2 => $n > 64$

$C1, C2$... Fator de Complexidade; função do número de portas em circuito lógicos, número de transistores para circuitos lineares e No de bits para memórias.

Number of logic gates	Failure rate (Failures per milion hours)
(a) logic circuits	
50	0.1527
100	0.2312
200	0.3655
500	1.4483
1000	14.4880

Memories (RAM) Number of bits	Failure rate (Failures per milion hours)
1024 (1K)	0.8837
2048 (2K)	1.3491
8192 (8K)	3.1453
16.384 (16K)	4.8033
32.768 (32K)	7.3362

6.3 Tempo Médio para Falhar – “Mean Time to Failure” – MTTF e a Confiabilidade de um Sistema

O Tempo Médio para Falhar - *MTTF* corresponde ao tempo médio esperado em que o sistema irá operar antes que a primeira falha ocorra.

Num exemplo prático, se tivermos N sistemas idênticos colocados em operação no instante $t = 0$ e medirmos o tempo que cada um desses N sistemas operem antes da ocorrência de uma falha, pode-se determinar o *MTTF* como:

$$MTTF = \frac{\sum_{i=1}^N t_i}{N}$$

Dada teoria de probabilidade, pode-se determinar o valor médio de uma variável x em função de sua densidade de probabilidade $f(x)$ através da seguinte formulação:

$$E[x] = \int_{-\infty}^{+\infty} x \cdot f(x) \cdot dx$$

Considerando-se na expressão anterior a variável x como sendo *tempo operacional até uma falha*, e a função $f(x)$ como a *função de densidade de falha*, o valor de $E(x)$ corresponderá ao valor de *MTTF*. Desta forma têm-se:

$$MTTF = \int_0^{+\infty} t \cdot f(t) \cdot dt$$

Como trata-se da variável tempo, não tem sentido valores negativos, razão pela qual a integral é realizada nos intervalos de 0 a ∞ .

A *função densidade de falha* $f(t)$ pode ser calculada através da variação da Não - Confiabilidade:

$$f(t) = \frac{dQ(t)}{dt}$$

Substituindo a função $f(t)$ na expressão do *MTTF* têm-se:

$$MTTF = \int_0^{\infty} t \cdot \frac{dQ(t)}{dt} \cdot dt = - \int_0^{\infty} t \cdot \frac{dR(t)}{dt} \cdot dt$$

O principal objetivo neste momento é procurar obter uma relação entre a confiabilidade $R(t)$ e o *MTTF*. Para tal, utilizaremos a teoria de cálculo integral.

Dada duas funções, pode-se calcular a derivada da multiplicação através da seguinte formulação:

$$\frac{df(t) \cdot g(t)}{dt} = f(t) \cdot \frac{dg(t)}{dt} + g(t) \cdot \frac{df(t)}{dt}$$

Realizando-se a integração entre os instante 0 e t de ambos os lados têm-se:

$$\int_0^t \frac{df(t) \cdot g(t)}{dt} \cdot dt = \int_0^t f(t) \cdot \frac{dg(t)}{dt} \cdot dt + \int_0^t g(t) \cdot \frac{df(t)}{dt} \cdot dt$$

Pode-se então escrever:

$$[f(t).g(t)]_0^t = \int_0^t f(t). \frac{dg(t)}{dt}.dt + \int_0^t g(t). \frac{df(t)}{dt}.dt$$

Considerando $f(t) = R(t)$ e $g(t) = t$ têm-se:

$$[R(t).t]_0^t = \int_0^t R(t).dt + \int_0^t t. \frac{dR(t)}{dt}.dt$$

Quando t tende a infinito pode-se escrever:

$$[R(t).t]_0^\infty = \int_0^\infty R(t).dt - MTTF$$

Como pode considerar que o valor de $R(t)$ tende a zero antes que t tenda a infinito, pode-se afirmar que:

$$(R(t).t)^{t \rightarrow \infty} - (R(t).t)^{t \rightarrow 0} = 0$$

Assim têm-se:

$$0 = \int_0^\infty R(t).dt - MTTF$$

Desta forma pode-se concluir uma expressão geral que represente o valor de $MTTF$ em função da Confiabilidade.

$$MTTF = \int_0^\infty R(t).dt$$

Caso a função confiabilidade $R(t)$ obedeça à lei de falha exponencial, característica plausível para componentes elétricos/eletrônicos/computacionais, têm-se:

$$MTTF = \int_0^\infty e^{-\lambda.t}.dt = \left[\frac{e^{-\lambda.t}}{-\lambda} \right]_0^\infty = \frac{1}{\lambda}$$

Pode-se concluir que o valor do $MTTF$, para componentes que apresentem uma função de confiabilidade exponencial, corresponde ao valor inverso da taxa de falhas.

Se formos calcular a confiabilidade de um sistema no instante $t = MTTF$, com distribuição exponencial de falhas, tem-se:

$$R(MTTF) = e^{-\lambda.MTTF} = e^{-\lambda.\frac{1}{\lambda}} = e^{-1} = 0,3678$$

Em outras palavras um sistema obedecendo a lei de falha exponencial tem uma probabilidade de 0,3678 de não experimentar uma falha antes do tempo igual a $MTTF$, dado que o sistema estava em correto funcionamento no início do período de tempo.

6.4. Tempo Médio para Reparo

O Tempo Médio para Reparo – $MTTR$ corresponde simplesmente ao tempo médio requerido para reparar um determinado sistema, que se encontrava falho.

Neste sentido, pode-se avaliar este tempo inserindo-se N defeitos em N sistemas idênticos e medindo-se o tempo que a equipe de manutenção leva para repará-los. Evidentemente este tempo de reparo irá depender da qualidade do alarme sendo emitido, destacando sua precisão quanto à localização exata da falha origem. Outro aspecto importante é que os N defeitos introduzidos sejam representativos dos possíveis de acontecerem, podendo ser considerados como uma boa representação do universo de defeitos possíveis. Neste livro não se entrará em maiores detalhes sobre a representativa dessa amostra e do grau de confiança do resultado final. O objetivo principal é se chegar a uma expressão geral do conceito de $MTTR$.

Tendo feito essas considerações pode-se considerar o $MTTR$ como:

$$MTTR = \frac{\sum_{i=1}^N t_i}{N}$$

Quando se trata de uma manutenção em laboratório, em que a rapidez e a qualidade da manutenção depende muito do nível técnico do profissional

envolvido na atividade, este valor do $MTTR$ pode ser extremamente influenciado pelo tipo de profissional e, portanto, deve ser levado em consideração. Tendo em vista este novo aspecto e considerando M técnicos envolvidos no trabalho, o novo $MTTR$ pode ser calculado como:

$$MTTR = \frac{\sum_{i=1}^M MTTR_i}{M}$$

onde $MTTR_i$ corresponde ao valor médio calculado anteriormente sobre N tipos de defeitos inseridos.

O $MTTR$ pode também ser representado através do inverso da taxa de reparo (μ), que é o número médio de reparos que ocorrem num determinado período de tempo.

6.5. Tempo Médio Entre Falhas

O Tempo Médio entre Falhas - $MTBF$ pode ser calculado através da media de tempo entre falhas consecutivas, incluindo o tempo para reparar o sistema e colocá-lo de volta ao estado operacional.

Para a realização prática desta medida pode-se colocar N sistemas idênticos em funcionamento por um período de tempo T , e anotar o número total de falhas em cada um desses sistemas (n_i) considerando-se, evidentemente, a realização de manutenções.

Desta forma pode-se estabelecer que o número médio de falhas seja avaliado como:

$$N_{Médio} = \frac{\sum_{i=1}^N n_i}{N}$$

Como esse número médio de falhas foi detectado ao longo de um período de tempo T , pode-se calcular o $MTBF$ como:

$$MTBF = \frac{T}{N_{Médio}}$$

Para se compreender a relação entre o *MTBF*, o *MTTF* e o *MTTR* pode-se observar a figura 6.2, que ilustra os seus instantes de ocorrência.

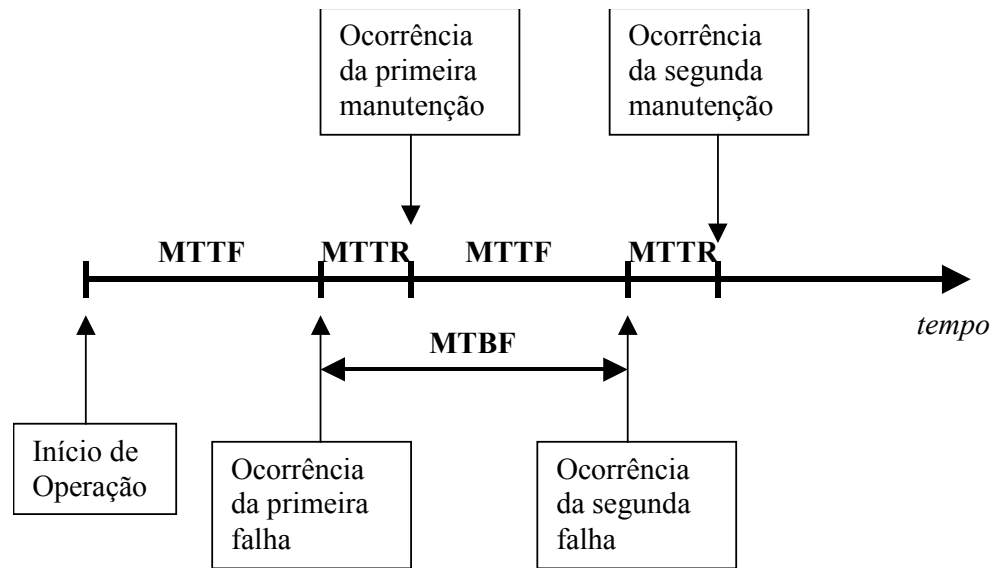


Figura 6.2 – Relação entre MTTF, MTTR e MTBF

Com base na figura pode-se concluir que:

$$MTBF = MTTF + MTTR$$

Na maioria dos sistemas o valor do *MTTR* corresponde a uma pequena fração do valor do *MTBF*. Este fato torna os valores do *MTTF* e do *MTBF* bastante próximos. Do ponto de vista conceitual eles são valores bastantes diferentes, diferença essa que irá influenciar na determinação da disponibilidade final do sistema.

6.6 A Disponibilidade Assintótica de um Sistema

Conforme já foi comentado anteriormente, a disponibilidade de um sistema corresponde à probabilidade dele estar funcionando corretamente em um determinado instante de tempo.

No entanto, esta curva de disponibilidade tende a um valor estável, denominado disponibilidade assintótica.

Este valor de disponibilidade assintótica pode ser calculado da seguinte forma:

$$A_{assintótica} = \frac{TempoOperacional}{TempoTotal} = \frac{\sum_{i=1}^n (MTTF_i)}{\sum_{i=1}^n (MTTF_i + MTTR_i)}$$

Dessa forma, têm-se:

$$A_{assintótica} = \frac{MTTF}{MTTF + MTTR} = \frac{MTTF}{MTBF}$$

Por exemplo, um sistema computacional com *MTTF* igual a 10.000 horas e um valor de *MTTR* igual a 1 hora, apresenta uma disponibilidade assintótica de 0,9999.

6.7. Cobertura de Defeitos.

A Cobertura de Defeito num sistema pode ter um tremendo impacto na Confiabilidade, Segurança e outros atributos do sistema.

Há muitos tipos de cobertura, dependendo se o projetista está querendo detecção do defeito, localização do defeito, (local e não global) contenção do defeito ou recuperação (manter o estado operacional) do defeito.

A Cobertura da Detecção do Defeito é a medida da habilidade do sistema em detectar defeitos e assim nos outros casos.


Na maioria dos casos, “cobertura de defeito” implica em “Cobertura da Recuperação do Defeito”.

Matematicamente, é a probabilidade condicional dada a existência de um defeito, do sistema recuperar:

$$C = P (\text{fault recovery} / \text{fault existence}).$$

A cobertura da Recuperação dos Defeitos é calculada como a função das falhas, que podem ser recuperadas, dividida pelo número total de falhas.

Exemplo:

Seja o circuito a seguir: (15 pontos de erro).

 preso em “0”

→ preso em “1”

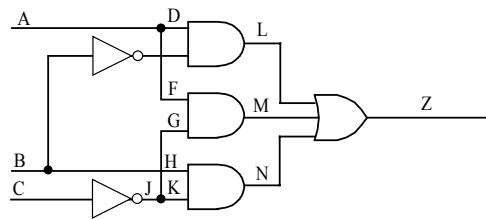


Figura 5.79

Figura 6.3

Como resultado, tem-se que a cobertura da detecção da falha é:

$$C = \frac{30 - 3 \text{ (Nº de modos de falhas detetáveis)}}{30 \text{ (Nº de modos de falhas possíveis)}} = 0,9 \longrightarrow 90\%$$

A tabela a seguir, apresenta todos os defeitos possíveis e seus respectivos vetores de teste de detecção.

Fault	Number of test vectors	Test vectors ABC
A ₀	2	100, 101
A ₁	2	000, 001
B ₀	2	010, 111
B ₁	2	000, 101
C ₀	2	011, 111
C ₁	2	010, 110
D ₀	1	101
D ₁	2	000, 001
E ₀	1	101
E ₁	1	111
F ₀	0	
F ₁	2	000, 001
G ₀	0	
G ₁	1	111
H ₀	1	010
H ₁	1	000
I ₀	1	111
I ₁	1	101
J ₀	2	010, 110
J ₁	2	011, 111
K ₀	1	010
K ₁	2	011, 111
L ₀	1	101
L ₁	4	000, 001, 011, 111
M ₀	0	
M ₁	4	000, 001, 011, 111
N ₀	1	010
N ₁	4	000, 001, 011, 111
Z ₀	4	010, 100, 101, 110
Z ₁	4	000, 001, 011, 111

6.8. Modelo de Confiabilidade

Um problema de se medir este atributo de um sistema é o número de sistemas necessários para se atingir um resultado bom. Isto é problemático quando existe a limitação do número de sistemas.

Outro problema é que os sistemas tem atingido índice de confiabilidade de 0,997 depois de 10 horas de operação, que corresponde a uma taxa de falhas, de 10^{-8} falhas/h.

$$R(t) = e^{-\lambda t} 0,9999999 = e^{-\lambda t}$$

$$\ln(0,9999999) = -\lambda \cdot t = 10^{-8} \text{ falhas/h}$$

1 falhas a cada 11416 anos.

As técnicas de análise de confiabilidade mais usuais são as abordagens analíticas. As que mais se destacam são os Modelos Combinatórias e por Markov.

6.8.1. Modelos Combinatórios

Esses modelos usam técnicas probabilísticas que enumeram os diferentes caminhos no qual o sistema pode permanecer operacional.

As probabilidades dos eventos que fazem com que o sistema permaneça operacional são calculadas para formar uma estimativa da confiabilidade do sistema.

A confiabilidade de um sistema é geralmente derivada em termos de confiabilidade dos componentes individuais do sistema. Os dois modelos mais comuns na prática são: **série** e **paralelo**. Num modelo **série**, cada elemento do sistema é requerido operar corretamente para que o sistema opere corretamente. No modelo **paralelo**, por outro lado, apenas um dos diversos elementos deve estar operacional para que o sistema desempenhe suas funções corretamente.

Na prática, os sistemas são típicas combinações de subsistemas série e paralelo.

Serão discutidas agora as técnicas de modelagem de sistema **série e paralelo**.

6.8.1.1. Sistema séries

O Sistema série é a melhor abordagem de um sistema que não contém redundância, ou seja, cada elemento do sistema é necessário para fazê-lo funcionar corretamente.

Uma maneira de representar o sistema série é através do DIAGRAMA DE BLOCO DE CONFIABILIDADE.

Este diagrama pode ser pensado como o diagrama de fluxo desde a entrada do sistema até a sua saída. Cada elemento do sistema é um bloco no diagrama de série. Os blocos são colocados em série para indicar o caminho da entrada para a saída. O fluxo é quebrado se um dos elementos falhar.

Um Diagrama de Bloco de Confiabilidade geral é apresentado a seguir é apresentado a seguir:

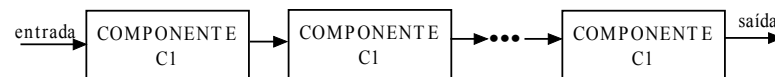


Figura 6.1

Figura 6.4

A confiabilidade do sistema série pode ser calculada como a probabilidade de nenhum dos elementos falharem. Outra maneira é que a confiabilidade do sistema série é a probabilidade de todos os elementos funcionarem corretamente.

Vamos supor que $C_{iW}(t)$ represente o evento do componente C_i estar funcionando corretamente no instante t , e $R_{series}(t)$ é a confiabilidade do sistema série.

A confiabilidade do sistema série, no instante t é:

$$R_{series}(t) = P(C_{1W}(t) \cap C_{2W}(t) \cap \dots \cap C_{NW}(t))$$

Assumindo que os eventos $C_{iW}(t)$ são independentes:

$$R_{\text{series}}(t) = R_1(t) \cdot R_2(t) \cdot R_3(t) \dots R_N(t)$$

$$R_{\text{series}}(t) = \prod_{i=1}^N R_i(t)$$

Uma relação interessante existe num sistema série se cada componente individual satisfaz a lei de falhas exponencial.

Suponha que cada componente tenha uma taxa de falha λ_i , e que $R_i(t) = e^{-\lambda_i t}$.

A confiabilidade do sistema será:

$$R_{\text{series}}(t) = e^{-\lambda_1 t} \cdot e^{-\lambda_2 t} \dots e^{-\lambda_N t}$$

$$= e^{-t \cdot \sum_{i=1}^N \lambda_i}$$

$$R_{\text{series}}(t) = e^{-\lambda_{\text{sist}} \cdot t}$$

$$\lambda_{\text{sistema}} = \sum_{i=1}^N \lambda_i$$

que corresponde á taxa de falha do sistema.

Exemplo: (Sistema de controle de avião)

Veja o sistema apresentado a seguir:

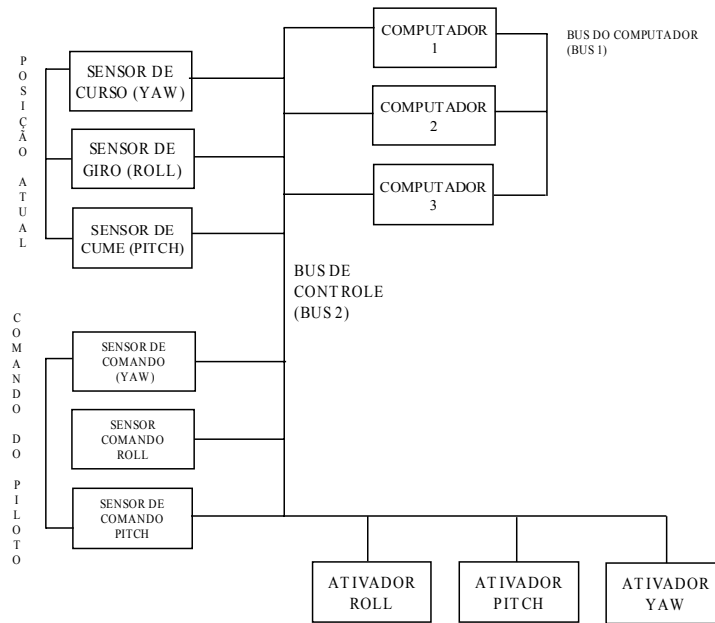


Figura 6.2

Os computadores recebem dos sensores de comando posições desejadas, comparam com as atuais, processam e enviam aos atuadores através do BUS2. Um bus de alta velocidade interconecta os computadores com o propósito de transferência de dados entre os computadores.

Não há redundância, e cada elemento do sistema é requerido para que o sistema desempenhe corretamente sua função.

O Diagrama em bloco de Confiabilidade é apresentado a seguir:

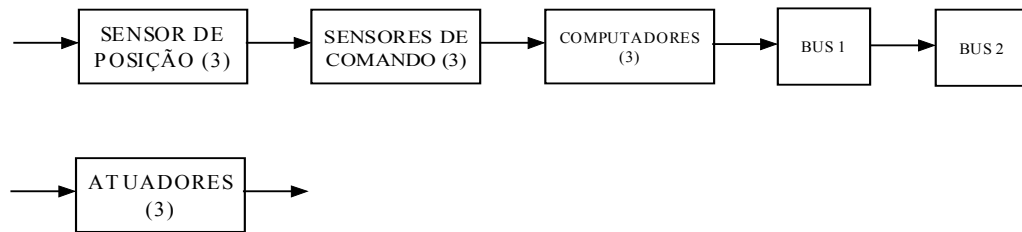


FIGURA 6.3

Figura 6.6

Por simplicidade assume-se que todos os seis sensores tem a mesma confiabilidade $R_s(t)$, cada atuador tem a confiabilidade $R_{act}(t)$, o bus1 tem confiabilidade $R_{bus1}(t)$ e o bus 2 de controle $R_{bus2}(t)$

O computador tem confiabilidade $R_{act}(t)$.

$$\Rightarrow R_{\text{sistema}}(t) = R_s(t)^6 \cdot R_{\text{act}}(t)^3 \cdot R_c(t)^3 \cdot R_{\text{bus1}}(t) \cdot R_{\text{bus2}}(t)$$

$$\Rightarrow \lambda_{\text{sistema}} = 6\lambda_s + 3\lambda_{\text{act}} + 3\lambda_c + \lambda_{\text{bus1}} + \lambda_{\text{bus2}}$$

\uparrow \uparrow \uparrow
 taxa de falha do sensor taxa de falha do atuador taxa de falha do computador

As taxas de falha de cada componente são:

$$\lambda_s = 1 \times 10^{-6} \text{ falhas por hora}$$

$$\lambda_{\text{act}} = 1 \times 10^{-5} \text{ falhas por hora}$$

$$\lambda_c = 4 \times 10^{-4} \text{ falhas por hora}$$

$$\lambda_{\text{bus1}} = 1 \times 10^{-6} \text{ falhas por hora}$$

$$\lambda_{\text{bus2}} = 2 \times 10^{-6} \text{ falhas por hora}$$

$$\lambda_{\text{sistema}} = 1,239 \cdot 10^{-3} \text{ falhas por hora}$$

A confiabilidade é:

$$R_s(t) = e^{-\lambda_s \cdot t} = e^{-1,239 \cdot 10^{-3} \cdot t}$$

Após 5 horas a confiabilidade é

$R_s(5h) = 0,994$

$$O \text{ MTTFs} = \frac{1}{\lambda_s} = \text{MTTF} = 807,10 \text{ horas}$$

λ_s (Baixíssima)

6.8.1.2. Sistemas Paralelos

A característica básica deste sistema é que apenas um dos N elementos idênticos é requerido para o funcionamento do sistema.

O Diagrama em Bloco de Confiabilidade de um sistema paralelo, que contém N elementos idênticos, está mostrado abaixo:

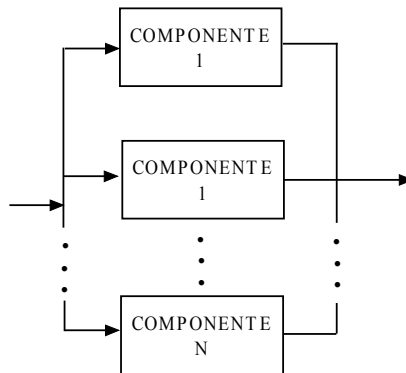


FIGURA 6.4

Figura 6.7

A não confiabilidade do sistema paralelo pode ser computada como a probabilidade de que todos os N elementos falhem.

Seja $C_{if}(t)$ o evento que o elemento i tenha falhado no instante t , e $Q_{\text{paral}}(t)$ a não confiabilidade do sistema paralelo e $Q_i(t)$ a não confiabilidade do i ésimo elemento.

$$Q_{\text{paral}}(t) = P(C_{1f}(t) \cap C_{2f}(t) \cap \dots \cap C_{nf}(t))$$

$$Q_{\text{paral}}(t) = Q_1(t) \cdot Q_2(t) \dots Q_n(t) = \prod_{i=1} Q_i(t)$$

Matematicamente tem-se $R(t) + Q(t) = 1$

$$R(t) = 1,0 - Q(t) = 1,0 - \prod_{i=1}^N Q_i(t) =$$

$$1,0 - \prod_{i=1}^N (1 - R_i(t))$$

Neste modelo considera-se também que as falhas dos elementos são independentes.

Para falhas de hardware a independência das falhas é uma boa suposição, mas para falhas resultantes de distúrbios externos a independência não é uma boa hipótese, pois o distúrbio externo é um evento comum, provador de diversos tipos de falhas.

Para ilustrar a aplicação do modelo seja o sistema a seguir:

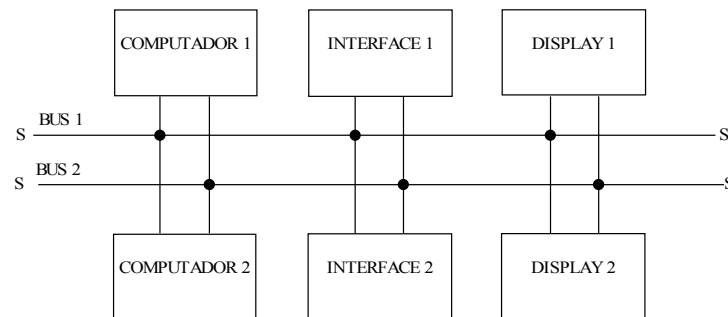


Figura 6.8

O sistema requer que pelo menos uma unidade de cada componente opere corretamente para que o sistema desempenhe suas funções.

Uma vez que uma unidade particular falhe, é assumido que a outra unidade assumirá automaticamente as funções.

Um aspecto importante neste exemplo é que ela apresente estrutura paralela e serial.

O aspecto paralelo é que apenas um dos componentes (mesmo tipo) deve funcionar. O aspecto série é que um componente de cada tipo deve estar funcionando.

O Diagrama em Bloco da Confiabilidade do sistema é mostrado a seguir:

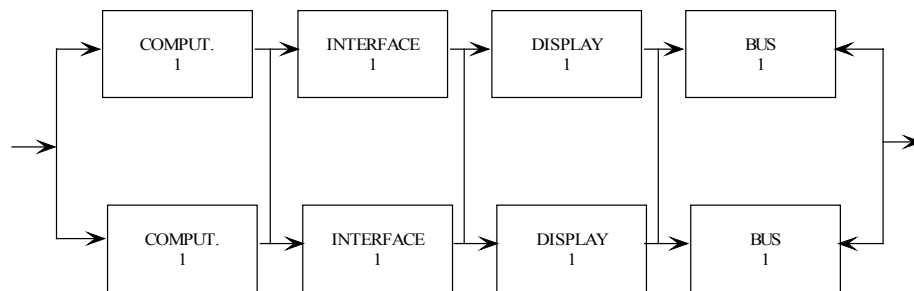


Figura 6.9

Este diagrama Série/ Paralelo pode ser reduzido a um diagrama série trocando cada porção paralela do sistema por uma equivalente, com a mesma confiabilidade.

Veja a figura a seguir:

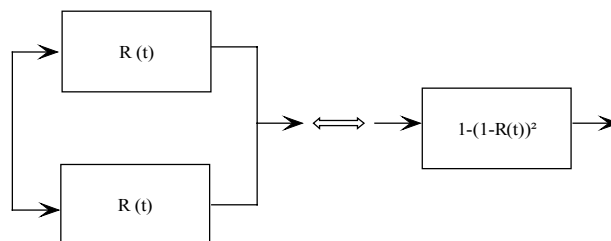


Figura 6.10

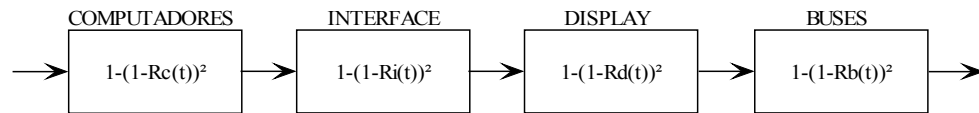
Seja $R_c(t)$ a confiabilidade de um computador

$R_i(t)$ a confiabilidade de uma interface,

$R_d(t)$ a confiabilidade de um display e

$R_b(t)$ a confiabilidade de um bus.

O diagrama de bloco resultante série é:



$$R_{\text{sistema}} = [1 - (1 - R_c(t))^2] \cdot [1 - (1 - R_i(t))^2] \cdot [1 - (1 - R_d(t))^2] \cdot [1 - (1 - R_b(t))^2]$$

Seja a título de exemplo a confiabilidade depois de 1 hora.

$$R_C(1) = R_i(1) = R_d(1) = R_b(1) = 0,9 \quad R_{\text{sistema}}(1 \text{ hora}) = 0,96.$$

Agora já se pode analisar a vantagem da redundância na confiabilidade.

O mesmo sistema anterior sem redundância tem uma confiabilidade depois de 1h de:

$$R_{\text{sistema}}(1 \text{ hora}) = R_c(1h) \cdot R_i(1h) \cdot R_d(1h) \cdot R_b(1h)$$

$$R_{\text{sistema}}(1h) = 0,6561$$

6.8.1.3. Cobertura de Defeito e seu Impacto na Confiabilidade

Como definido anteriormente, cobertura de defeito é a medida da habilidade do sistema recuperar-se de defeitos.

Por exemplo, num sistema redundante antes que a redundância seja usada, requiere-se uma boa cobertura de defeitos.

Durante a análise dos sistemas paralelos supõe-se que a cobertura dos defeitos era perfeita.

A cobertura não perfeita está no fato de o sistema não ser capaz de usar a redundância, pois não pode identificar que a unidade está em falha e, portanto, não a remove e não a troca por uma outra unidade livre de falha.

Para ilustrar veja a figura a seguir:

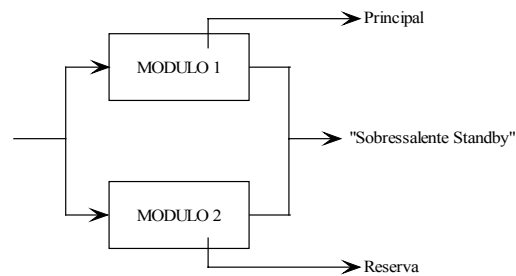


Figura 6.11

Este sistema paralelo de dois módulos funciona corretamente se uma das condições a seguir existirem:

1. Módulo 1 está funcionando corretamente;
2. Módulo 2 está funcionando corretamente, módulo 1 falhou e a falha foi detectada e apropriadamente tratada.

A probabilidade que um destes eventos existe pode ser calculada como:

$$R_{\text{sistema}}(t) = R_1(t) + (1 - R_1(t)) \cdot C_1 R_2(t)$$

onde C_1 é a cobertura de defeitos associada ao módulo 1.

Se a confiabilidade e o fator de cobertura dos dois módulos são idênticos a expressão de confiabilidade é dada por:

$$R_{\text{sistema}}(t) = R(t) + R(t) \cdot C \cdot (1 - R(t))$$

Se o fator de cobertura for $C=1$, então:

$$R_{\text{sistema}}(t) = 2R(t) - R^2(t) = 1 - (1 - R(t))^2$$

Se o fator de cobertura for $C=0$, então:

$R_{\text{sistema}}(t) = R(t)$ que é a confiabilidade de **apenas um módulo**.

A figura a seguir, mostra o impacto do fator de cobertura na confiabilidade do sistema.

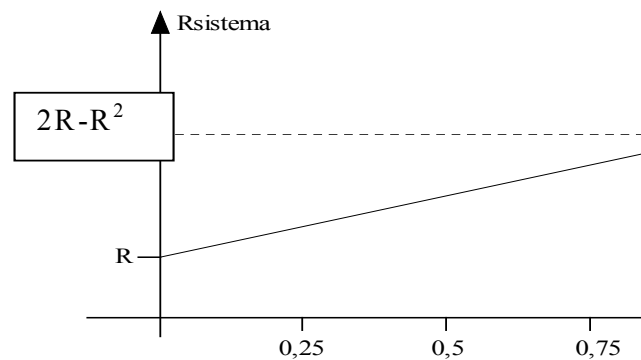


Figura 6.12

Neste modelo, o módulo 1 é o módulo principal enquanto estiver funcionando corretamente, e o sistema funciona corretamente, e o sistema funciona corretamente, até mesmo se o módulo 2 falha. Isto pode não ser verdade em sistemas que realizam comparações entre os dois módulos. Neste caso, se o módulo falho for detectado, o sistema continua operando com o módulo livre de defeito e o mecanismo de comparação é desabilitado. Se o módulo falho não for detectado, o sistema descontinua a operação. Também neste caso, deve ser considerada a cobertura de defeitos.

Vamos ilustrar agora o exemplo anterior com comparação entre os dois módulos como meio de detecção de defeito.

Assume-se que a **comparação é perfeita** e detecta todas as falhas.

Uma vez que o processo de comparação detecte o defeito, o sistema implementa auto diagnóstico para determinar qual módulo está falho.

Se o defeito pode ser localizado, o módulo livre de defeito começa a desempenhar as funções do sistema.

O sistema funciona corretamente enquanto ambos os módulos estiverem funcionando corretamente ou um defeito ocorreu e foi detectado e manipulado (tratado) corretamente.

A confiabilidade do sistema pode ser colocada como:

$$R_{\text{sistema}}(t) = R_1(t) \cdot R_2(t) + \underbrace{R_1(t) \cdot (1 - R_2(t))}_{\text{módulo2 falhou}} \cdot \underbrace{C_2}_{\text{módulo1 falhou}} + (1 - R_1(t)) C_1 \cdot R_2(t)$$

C1 - cobertura do auto - teste do módulo 1.

C2 - cobertura do auto - teste do módulo 2.

Se a confiabilidade e a cobertura de cada módulo são idênticas, então:

$$R_{\text{sistema}}(t) = R^2(t) + 2R(t) \cdot C \cdot (1 - R(t))$$

Para cobertura de defeito perfeita (C=1) têm-se:

$$R_{\text{sistema}}(t) = R^2(t) + 2R(t)(1 - R(t)) = 1 - (1 - R(t))^2$$

Se o fator de cobertura for zero (C=0), então:

$$R_{\text{sistema}}(t) = R^2(t)$$

Caso o sistema deve parar de funcionar quando ambos os módulos discordarem então:

$R_{\text{sistema}}(t) = R_1(t) \cdot R_2(t)$ com funcionamento perfeito da **comparação**.

6.8.1.4. Sistema M de N

Estes sistemas são uma generalização do sistema paralelo ideal.

No sistema paralelo ideal, apenas **um** dos **N** módulos é requerido trabalhar para o sistema funcionar. Num sistema **M de N**, M módulos dos N devem funcionar para o sistema agir corretamente.

Um bom exemplo deste sistema é o TMR (“Redundância Modular Tripla”) onde 2 módulos 3 devem agir no mecanismo de votação.

Vamos analisar um sistema TMR

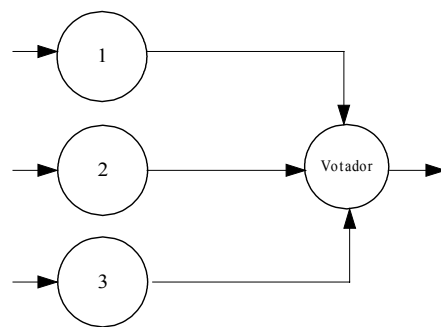


Figura 6.13

Obs.: Considerar que a Confiabilidade do Votador = 1.

$$RTMR(t) = R_1(t).R_2(t).R_3(t) + R_1(t).R_2(t).(1-R_3(t)) +$$

3 módulos
funcionando

Módulo 3
falhou

$$+ R_1(t) . (1-R_2(t)) . R_3(t) + (1-R_1(t)) . R_2(t) . R_3(t)$$

módulo 2
falhou

módulo 1
falhou

Considerando $R_1(t) = R_2(t) = R_3(t) \Rightarrow$

$$RTMR = R^3(t) + 3R^2(t) . (1-R(t)) =$$

$$= R^3(t) + 3R^2(t) - 3R^3(t)$$

$$\Rightarrow RTMR(t) = 3R^2(t) - 2R^3(t)$$

Vamos comparar a confiabilidade de RTMR com um módulo isolado.

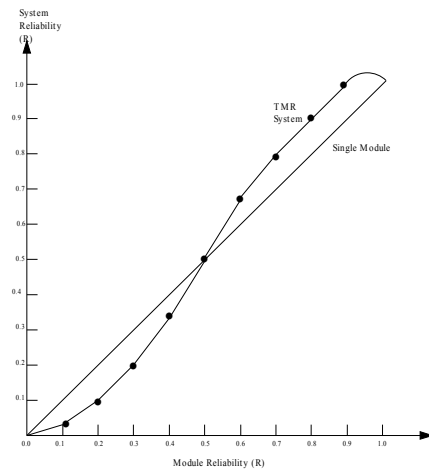


Figura 6.14

Esta figura mostra que existe um cruzamento entre a confiabilidade RTMR e R.

O ponto de cruzamento é:

$$3R^2 - 3R^3(t) = R \Rightarrow 3R - 2R^2 = 1$$

$$R^2 - \frac{3}{2}R + 0,5 = 0 \quad \{ R=0,5 \text{ e } R=1 \}$$

$$\Delta = \frac{9}{4} - 2 = \frac{1}{4}$$

$$R = \frac{3/2 \pm 1/2}{2} \quad \begin{matrix} 1 \\ 1/2 \end{matrix}$$

Agora dá para ilustra melhor a diferença entre o conceito de tolerância a Defeito e Confiabilidade.

Um sistema pode ser Tolerante a Defeito e ainda ter uma baixa confiabilidade.

Um sistema TMR com módulos de confiabilidade R= 0,5 pode tolerar defeitos em um dos módulos, mas a sua confiabilidade é a mesma que a de um módulo único.

Um sistema com um módulo com alta confiabilidade não é tolerante a defeito.

Observação importante: É possível que a confiabilidade de um sistema não redundante se aproxime de um sistema redundante com os mesmos módulos, entretanto o sistema não redundante não será tolerante a defeito.

A expressão geral de um sistema M de N é dada por:

$$R_{M-N}(t) = \sum_{i=0}^{N-M} \sum_i^{N-i} R(t) \cdot (1-R(t))^i \quad \begin{matrix} .M \text{ funcionando} \\ .i \text{ falhos} \end{matrix}$$

6.8.2 Modelos de Markov

Uma limitação do Modelo Combinatório é que não consegue modelar sistemas complexos. Pode ser bastante difícil construir o Diagrama de blocos de Confiabilidade.

O Modelo de Markov permite modelar o processo de reparo que ocorre em muitos sistemas, característica difícil de modelar em sistemas combinatórios.

Os dois principais conceitos no Modelo de Markov são os **Estados** e as **Transições**.

O **Estado** representa todas as situações conhecidas que descrevem o sistema num dado instante.

Para os modelos de confiabilidade, cada **estado** de Markov representa uma combinação diferente de módulos em falha e livres de falha.

Vamos analisar um sistema TMR. Definimos S que representa o estado do sistema, $S = (S_1, S_2, S_3)$ onde $S_i=1$ se o módulo i está livre de falha e $S_i = 0$ se o módulo i está falho.

São 8 estados possíveis.

As **transições** regulamentam as mudanças de estados que ocorrem dentro de um sistema.

O Diagrama de Estados do sistema TMR é apresentando a seguir:

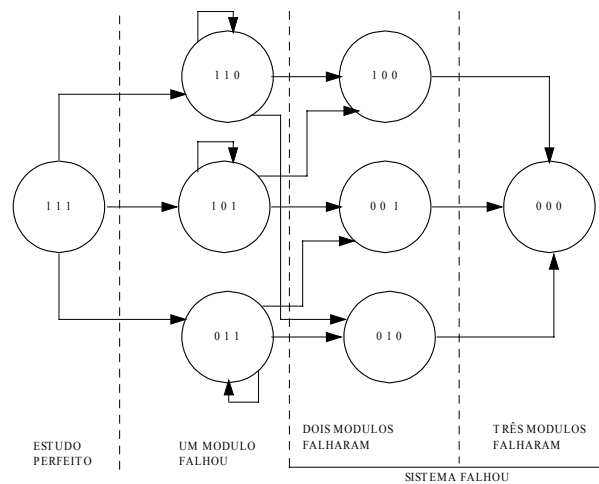


Figura 6.15

Vamos assumir que cada módulo no sistema TMR obedeça a lei de falha exponencial com uma taxa de falha λ , constante.

A probabilidade de um módulo estar falho no instante t é dado por \Rightarrow

$$\Rightarrow (1 - e^{-\lambda \Delta t})$$

Da matemática quando Δt é pequeno, então:

$$1 - e^{-\lambda \Delta t} \cong \lambda \cdot \Delta t$$

Pode-se agora, completar a figura anterior, colocando-se a probabilidade das transições:

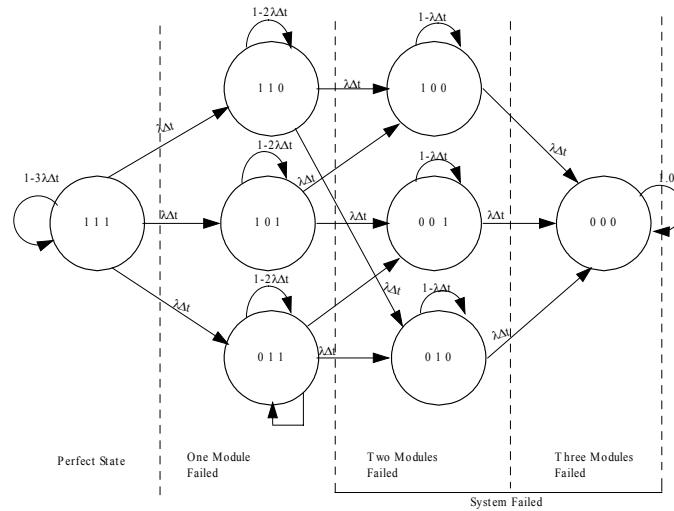
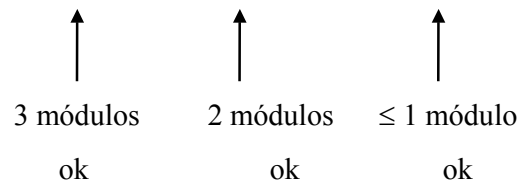


Figura 6.16

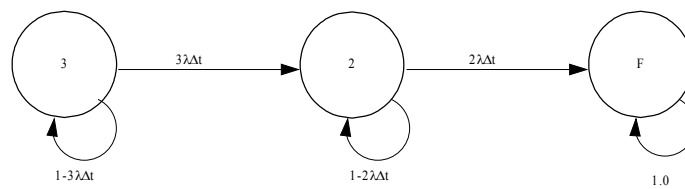
Reduzindo o Diagrama de Estados anterior, considerando os estados globais:

(Estão Perfeitos - Um módulo - Sistema falhou)

falhou



- Diagrama de Estados Resultante é dado por:



$$p_3(t + \Delta t) = (1 - 3\lambda\Delta t) \cdot p_3(t)$$

↑

Probabilidade do sistema estar no
estado 3 no instante t.

$$p_2(t + \Delta t) = (3\lambda\Delta t)p_3(t) + (1 - 2\lambda\Delta t) \cdot p_2(t)$$

$$p_F(t + \Delta t) = (2\lambda\Delta t)p_2(t) + p_F(t)$$

As equações do Modelo de Markov para o sistema TMR podem ser escritas como:

$$\begin{bmatrix} p_3(t + \Delta t) \\ p_2(t + \Delta t) \\ p_F(t + \Delta t) \end{bmatrix} = \begin{bmatrix} (1 - 3\lambda\Delta t) & 0 & 0 \\ 3\lambda\Delta t & (1 - 2\lambda\Delta t) & 0 \\ 0 & 2\lambda\Delta t & 1 \end{bmatrix} \begin{bmatrix} p_3(t) \\ p_2(t) \\ p_F(t) \end{bmatrix}$$

$$P(t + \Delta t) = A \cdot P(t)$$

$$t=0 \quad (P(\Delta t) = A \cdot P(0))$$

$$t= \Delta t \quad P(2\Delta t) = A \cdot P(\Delta t) = A^2 P(0)$$

↓

$$P(n\Delta t) = A^n \cdot P(0)$$

A probabilidade de um sistema falhar é dada pela probabilidade do sistema estar no estado falho.

$$\Rightarrow RTMR(t) = 1 - PF(t) = P_3(t) + P_2(t)$$

Através de manipulação algébrica, tem-se:

$$\frac{p_3(t+\Delta t) - p_3(t)}{\Delta t} = -3\lambda \cdot p_3(t)$$

$$p_2(t+\Delta t) = 3\lambda\Delta t \cdot p_3(t) + (1-2\lambda\Delta t) \cdot p_2(t)$$

$$p_2(t+\Delta t) - p_2(t) = 3\lambda\Delta t p_3(t) - 2\lambda\Delta t (t) \cdot p_2(t)$$

$$\frac{p_F(t+\Delta t) - p_F(t)}{\Delta t} = 2\lambda \cdot p_2(t)$$

Aplicando o limite $\Delta t \rightarrow 0$:

$$\left\{ \begin{array}{l} \frac{dp_3(t)}{dt} = -3\lambda p_3(t) \\ \frac{dp_2(t)}{dt} = 3\lambda p_3(t) - 2\lambda \cdot p_2(t) \\ \frac{dp_F(t)}{dt} = 2\lambda \cdot p_2(t) \end{array} \right.$$

Aplicando a **Transformação de Laplace**:

$$sP_3(s) - p_3(0) = -3\lambda P_3(s)$$

$$sP_2(s) - p_2(0) = 3\lambda P_3(s) - 2\lambda P_2(s)$$

$$sP_F(s) - p_F(0) = 2\lambda P_2(s)$$

Como no instante inicial ($t=0$) estamos supondo que o sistema esteja perfeito

\Rightarrow

$$p_3(0) = 1, p_2(0) = 0, \text{ e } p_F(0) = 0$$

$$P3(s) = \frac{1}{s+3\lambda}$$

$$P2(s) = \frac{3\lambda}{(s+2\lambda)(s+3\lambda)} = \frac{3}{(s+2\lambda)} + \frac{-3}{(s+3\lambda)}$$

$$PF(s) = \frac{6\lambda^2}{(s+2\lambda)(s+3\lambda)} = \frac{1}{s} + \frac{-3}{(s+2\lambda)} + \frac{2}{(s+3\lambda)}$$

$$p3(t) = e^{-3\lambda t}$$

$$p2(t) = 3e^{-2\lambda t} - 3e^{-3\lambda t}$$

$$pF(t) = 1 - 3 \cdot e^{-2\lambda t} + 2e^{-3\lambda t}$$

Desta forma a confiabilidade do sistema:

$$\begin{aligned} RTMR(t) &= p3(t) + p2(t) = e^{-3\lambda t} + 3e^{-2\lambda t} - 3e^{-3\lambda t} \\ &= \underline{3e^{-2\lambda t} - 2e^{-3\lambda t}} \end{aligned}$$

	Reliability	
Time (t) in minutes	Combinatorial results	Markov results
1	0.99999177	0.99999171
2	0.99996674	0.99996686
3	0.99992549	0.99992561
4	0.99986792	0.99986809
5	0.99979424	0.99979442
6	0.99970472	0.99970472
7	0.99959898	0.99959916
8	0.99947786	0.99947786
9	0.99934101	0.99934095
10	0.99918842	0.99948854

Failure rate λ is 0.1 failures per hour, and time step Δt is 0.1 seconds

6.8.2.1. Modelo de Markov com Cobertura de Defeitos

Analizamos o Modelo de Markov, quando usando em sistemas que não dependem de cobertura de defeitos ou processo de reparo.

Varemos agora o Modelo de Markov em sistemas que dependem da cobertura dos defeitos.

O sistema a ser modelado é de redundância tripla que usa técnicas de detecção de defeitos para detectar a ocorrência de um defeito em um dos três módulos independentes.

O modo correto de um módulo falho não interferir é ser removido através da abertura de uma chave.

A probabilidade que uma falha seja corretamente manipulada corresponde a cobertura dos defeitos, denotada por C .

A arquitetura básica do sistema é mostrada na figura que segue:

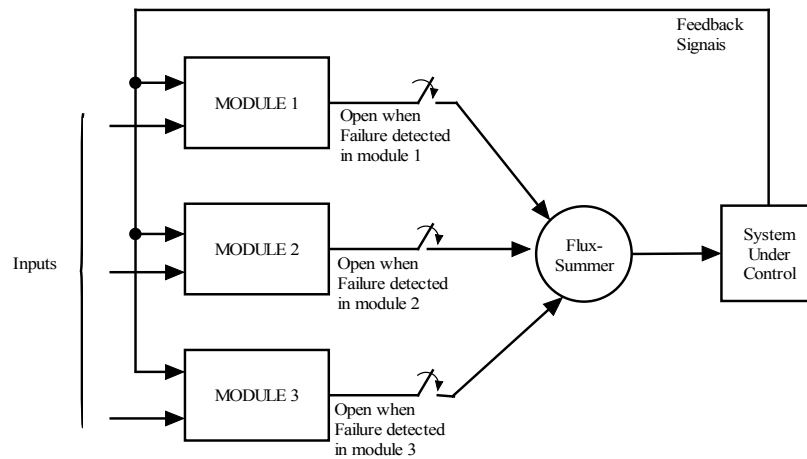


Figura 6.17

O Modelo de Markov, apresentado a seguir, contém erros, descubra!!!:

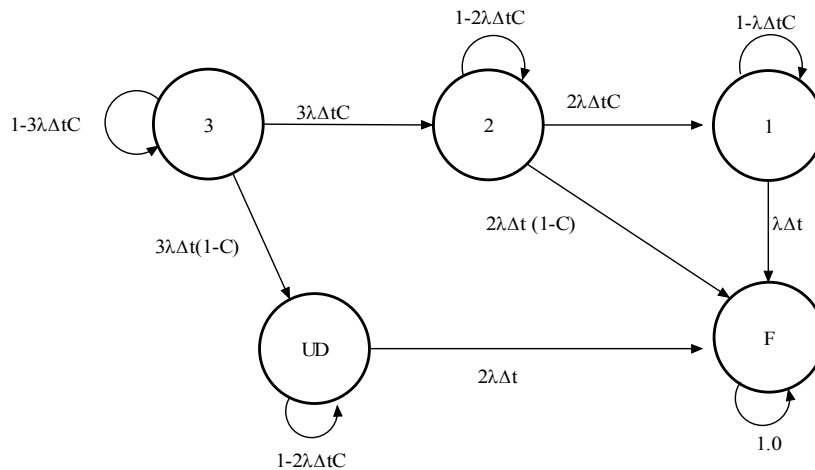


Figura 6.18

O sistema é assumido começar num estado sem defeito (estado 3).

Há dois caminhos a partir deste estado. O primeiro é a transição para o estado 2 e corresponde a uma falha em um dos três módulos detectável (dentro da cobertura).

O segundo é uma transição para o estado UD que corresponde a um dos três módulos falharem sem ser detectável.

Quando o sistema entra no estado UD torna-se um sistema TMR básico com votação majoritária.

Quando o sistema está no estado 2 pode tolerar uma falha detectável para o estado 1, ou caso seja uma falha não detectável o sistema vai para o estado falho.

As equações para o Modelo de Markov são:

$$\begin{cases} p_3(t + \Delta t) = p_3(t) \cdot (1 - 3\lambda\Delta t) \\ p_2(t + \Delta t) = p_3(t) \cdot 3\lambda\Delta t \cdot C + p_1(t) \cdot (1 - 2\lambda\Delta t) \\ p_1(t + \Delta t) = p_2(t) \cdot 2\lambda\Delta t \cdot C + p_1(t) \cdot (1 - \lambda\Delta t) \\ p_{uD}(t + \Delta t) = p_3(t) \cdot 3\lambda\Delta t \cdot (1 - C) + p_{uD}(t) \cdot (1 - 2\lambda\Delta t) \\ p_F(t + \Delta t) = p_2(t) \cdot 2\lambda\Delta t \cdot (1 - C) + p_1(t) \cdot \lambda\Delta t + p_{uD}(t) \cdot 2\lambda\Delta t + p_F(t) \end{cases}$$

A confiabilidade do sistema é a probabilidade de estar nos estados 3, 2, 1, ou UD.

$$R(t) = p_3(t) + p_2(t) + p_1(t) + p_{uD}(t).$$

A tabela a seguir mostra a confiabilidade em função da cobertura.

Fault coverage	Reliability (after 1 hour)	
0.0	0.97460	} → $\frac{\Delta R}{\Delta C} = 0,0024$
0.1	0.97484	
0.2	0.97558	
0.3	0.97680	
0.4	0.97852	} → $\frac{\Delta R}{\Delta C} = 0,0418$
0.5	0.98073	
0.6	0.98343	
0.7	0.98662	
0.8	0.99030	
0.9	0.99448	
1.0	0.99914	

Failure rate λ is 0.1 failures per hour, and time step Δt is 0.1 seconds

$$\begin{pmatrix} p_3(t + \Delta t) \\ p_2(t + \Delta t) \\ p_1(t + \Delta t) \\ p_{UD}(t + \Delta t) \\ p_F(t + \Delta t) \end{pmatrix} = \begin{pmatrix} 1-3\lambda\Delta t & 0 & 0 & 0 \\ 3\lambda\Delta t C & 1-2\lambda\Delta t & 0 & 0 \\ 0 & 2\lambda\Delta t C & 1-\lambda\Delta t & 0 \\ 3\lambda\Delta t(1-C) & 0 & 0 & 1-2\lambda\Delta t \\ 0 & 2\lambda\Delta t(1-C) & \lambda\Delta t & 2\lambda\Delta t \end{pmatrix} \times \begin{pmatrix} p_3(t) \\ p_2(t) \\ p_1(t) \\ p_{LD}(t) \\ p_F(t) \end{pmatrix}$$

Quando o fator de cobertura é zero, o sistema é idêntico a um sistema TMR com votação majoritária.

A tabela mostra que há um impacto maior na confiabilidade em valores de maior cobertura.

6.8.2.2. Modelo de Markov com Reparo

Considera-se agora, o sistema com reparo como forma de recuperação.

Considere o Modelo de Markov de um sistema simples consistindo de um computador sem redundância e assuma que o computador tenha uma taxa de falha constante de λ e taxa de reparo de μ .

O Modelo de Markov deste sistema é apresentado a seguir:

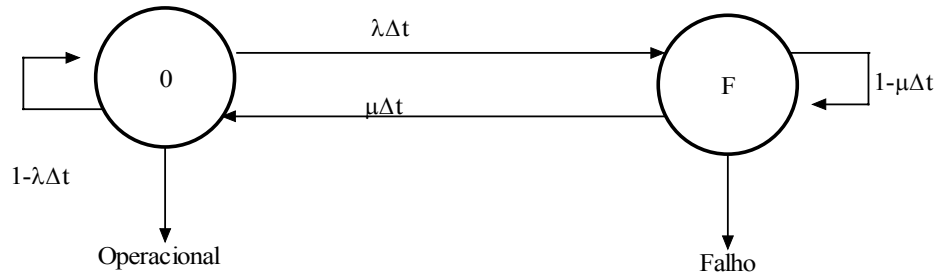


Figura 6.19

As equações deste modelo são:

$$\begin{cases} p_0(t+\Delta t) = p_0(t)(1-\lambda\Delta t) + p_F(t)\mu\Delta t \\ p_F(t+\Delta t) = p_0(t)\lambda\Delta t + p_F(t)(1-\mu\Delta t) \end{cases}$$

$$\begin{cases} \frac{p_0(t+\Delta t) - p_0(t)}{\Delta t} = -\lambda p_0(t) + \mu p_F(t) \\ \frac{p_F(t+\Delta t) - p_F(t)}{\Delta t} = \lambda p_0(t) - \mu p_F(t) \end{cases}$$

$$\Delta t \rightarrow 0$$

$$\begin{cases} \frac{dp_0(t)}{dt} = -\lambda p_0(t) + \mu p_F(t) \\ \frac{dp_F(t)}{dt} = \lambda p_0(t) - \mu p_F(t) \end{cases}$$

Condições iniciais: $p_0(0) = 1$ $p_F(0) = 0$

Passando para o domínio S = (Laplace)

$$\begin{cases} \text{SpO (S)} - \text{po(0)} = -\lambda \text{Po(S)} + \mu \text{PF(S)} \\ \text{SPF (S)} - \text{pF(0)} = -\lambda \text{Po(S)} + \mu \text{Pi(S)} \end{cases}$$

$$\begin{cases} \text{SPo (S)} = 1 - \lambda \text{Po(S)} + \mu \text{PF(S)} \\ \text{SPF (S)} = \lambda \text{Po(S)} - \mu \text{PF(S)} \end{cases}$$

$$\begin{cases} \text{Po (s)} = \frac{1}{\text{S} + (\lambda + \mu)} + \frac{\mu}{\text{S} + (\lambda + \mu)} \\ \text{Po (s)} = \frac{\lambda}{\text{S}(\text{S} + (\lambda + \mu))} \end{cases}$$

$$\begin{cases} \text{Po (s)} = \frac{\mu}{\lambda + \mu} \cdot \frac{1}{\text{S}} + \frac{\lambda}{\lambda + \mu} \cdot \frac{1}{\text{S} + (\lambda + \mu)} \\ \text{Po (s)} = \frac{\lambda}{\text{S}} \cdot \frac{1}{\lambda + \mu} + \frac{\lambda + \mu}{\text{S} + (\lambda + \mu)} \end{cases}$$

Aplicando a transformada de Laplace inversa:

$$\begin{cases} \text{Po (s)} = \frac{\mu}{\lambda + \mu} + \frac{\lambda}{\lambda + \mu} \cdot e^{-(\lambda + \mu)t} \\ \text{Po (s)} = \frac{\lambda}{\lambda + \mu} + \frac{\lambda}{\lambda + \mu} \cdot e^{-(\lambda + \mu)t} \end{cases}$$

Para $t=0 \rightarrow \text{po (0)} = 1$ e $\text{pF (0)} = 0$

Para $t \rightarrow \infty \Rightarrow$

$$p_o(\infty) = \frac{\mu}{\lambda + \mu} = \frac{1}{\frac{\lambda}{\mu} + 1}$$

$$\text{Se } \mu \gg \lambda \rightarrow \frac{\lambda}{\mu} = 0 \rightarrow p_o(\infty) = 1$$

$$\text{Se } \lambda \gg \mu \rightarrow \frac{\lambda}{\mu} = \infty \rightarrow p_o(\infty) = 0$$

$$p_F(\infty) = \frac{\lambda}{\lambda + \mu} = \frac{1}{1 + \frac{\mu}{\lambda}}$$

A seguir é apresentado um gráfico que mostra a probabilidade de um sistema estar operacional ($P_o(t)$) variando a taxa de reparo (μ) e considerando a taxa de falha $\lambda = 0,1$ falhas/h $\Delta t = 0,1$ segundos.

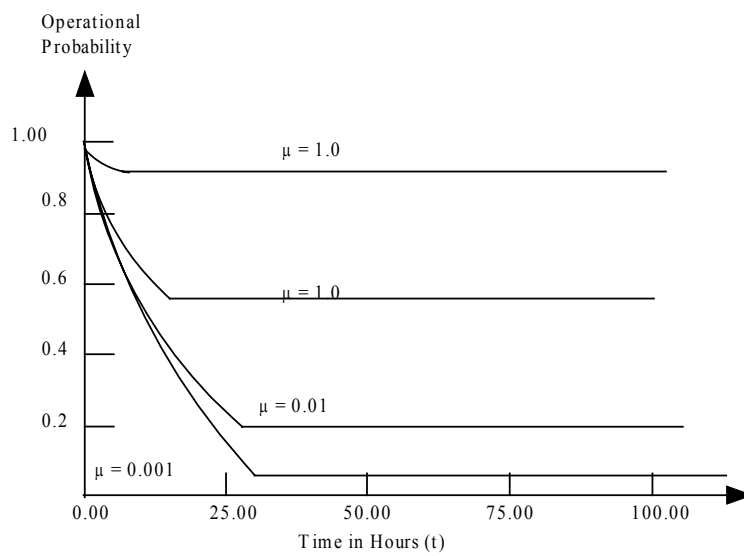


Figura 6.20

6.8.2.3. Modelo de Segurança

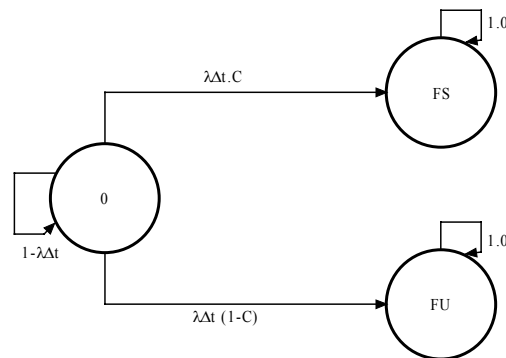
A definição de Segurança diz ser a probabilidade do sistema ou funcionar corretamente ou falhar de maneira segura.

As definições de estados Seguros e Inseguros devem ser criadas para cada aplicação.

No modelo de Markov o estado falho deve ser, explodido em FS, falha segura, e FU, falha insegura.

O modelo de Markov para um sistema com um único módulo com taxa de falha e auto diagnóstico com cobertura C é mostrado a seguir. Neste sistema as falhas seguras são consideradas aquelas detectadas pelo auto - diagnóstico.

Figura 6.21



A segurança do sistema é definida como:

$$S(t) = p_0(t) + p_{FS}(t)$$

As equações do modelo são:

$$\begin{cases} p_0(t+\Delta t) = (1-\lambda\Delta t) \cdot p_0(t) \\ p_{FS}(t+\Delta t) = \lambda\Delta t.C.p_0(t) + p_{FS}(t) \\ p_{FU}(t+\Delta t) = \lambda\Delta t.(1-C)p_0(t) + p_{FU}(t) \end{cases}$$

$$t \rightarrow 0$$

$$\begin{cases} \frac{dp_o(t)}{dt} = \lambda p_o(t) \\ \frac{dp_{FS}(t)}{dt} = \lambda \cdot C \cdot p_o(t) \\ \frac{dp_{FU}(t)}{dt} = \lambda (1-C) \cdot p_o(t) \end{cases}$$

Aplicando a transformada Laplace:

$$\begin{cases} P_o(S) = \frac{p_o(0)}{S+\lambda} \\ P_{FS}(S) = \frac{\lambda \cdot C \cdot P_o(0)}{S(S+\lambda)} + \frac{P_{FS}(0)}{S} \\ P_{FU}(S) = \frac{\lambda \cdot (1-C) \cdot P_o(0)}{S(S+\lambda)} + \frac{P_{FU}(0)}{S} \end{cases}$$

onde:

$$p_o(0) = 1, p_{FS}(0) = p_{FU}(0) = 0$$

$$P_o(S) = \frac{1}{S+\lambda}$$

$$P_{FS}(S) = \frac{\lambda \cdot C}{S(S+\lambda)} = \frac{(1-C)}{S} - \frac{(1-C)}{S+\lambda}$$

$$P_{FU}(S) = \frac{\lambda \cdot (1-C)}{S(S+\lambda)} = \frac{(1-C)}{S} - \frac{(1-C)}{S+\lambda}$$

Anti-transformando tem-se:

$$P_o(t) = e^{-\lambda t}$$

$$p_{FS}(t) = C - C \cdot e^{-\lambda t}$$

$$p_{FU}(t) = (1-C) - (1-C) \cdot e^{-\lambda t}$$

A confiabilidade do sistema é:

$$R(t) = P_o(t) = e^{-\lambda t}$$

A segurança do sistema é dada por:

$$S(t) = p_o(t) + p_{FS}(t) = C + (1-C) e^{-\lambda t}$$

No instante $t=0$ a Segurança é 1.

$$S(\infty) = C$$

A Segurança de um sistema está diretamente dependente da cobertura de detecção de defeito.

6.8.2.4 Comparação entre Sistemas

Pretende-se comparar o MTTF de dois sistemas, ou sua confiabilidade.

Os sistemas a serem comparados são o Simples e o TMR.

Assume-se que a votação é perfeita.

O MTTF = $\int_0^{\infty} R(t) dt$, onde

$R(t)$ é a confiabilidade do sistema.

$$R_{\text{simples}} = \int_0^{\infty} e^{-\lambda t} \cdot dt = \frac{1}{\lambda}$$

$$MTTF_{\text{TMR}} = \int_0^{\infty} (3 \cdot e^{-2\lambda t} - 2 \cdot e^{-3\lambda t}) \cdot dt = \frac{5}{6\lambda}$$

O gráfico que segue mostra a confiabilidade em função de λt .

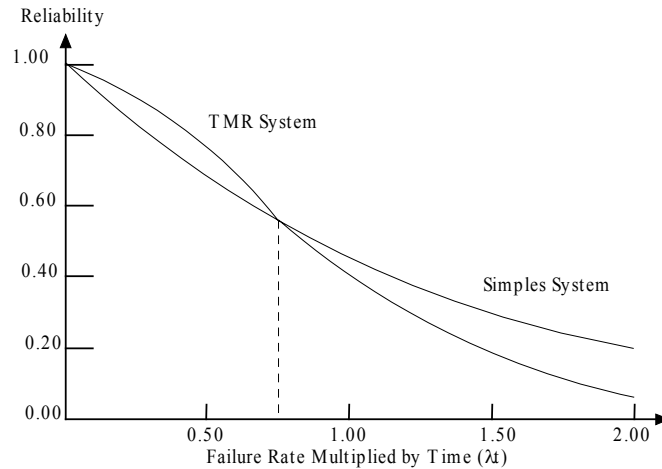


Figura 6.22

Para certos valores λt , a confiabilidade do sistema TMR é inferior em relação ao Sistema simples.

Desta forma, o MTTF pode não representar adequadamente a qualidade do sistema.

Em alguns casos é mais importante determinar o TEMPO DA MISSÃO, que é o tempo requerido no qual a confiabilidade do sistema permaneça superior a um determinado nível.

Seja o sistema simples e o nível de confiabilidade de missão r :

$$r = e^{-\lambda t}$$

$$\ln r = -\lambda \cdot t$$

$$t = \frac{-\ln r}{\lambda}$$

Tempo da Missão

Para um sistema TMR:

$$r = 3 e^{-2\lambda t} - 2 e^{-3\lambda t}$$

Para um $r = 0,86$ e $\lambda = 0,01$ falhas/h

$T_{\text{MISSÃO}} - \text{simples} = 15,08$ horas e

$T_{\text{MISSÃO}} - \text{TMR} = 27$ horas e

Em outras palavras, um sistema TMR pode operar, neste exemplo, 1,8 vezes mais tempo mantendo uma confiabilidade maior do que 0,86.

Seja um exemplo ilustrativo apresentado a seguir.

Apesar da confiabilidade do sistema (1) tender a zero quando λt cresce, o tempo de missão t_1 é maior do que o tempo de missão t_2 para a curva (2).

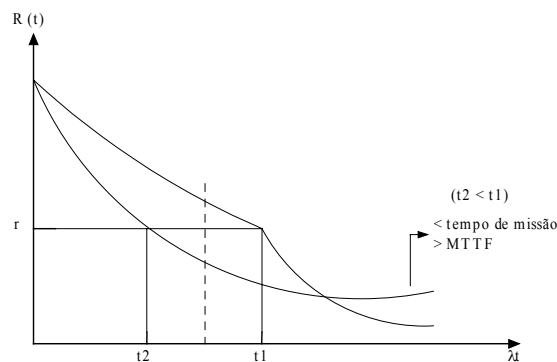


Figura 6.23

6.8.2.5. Modelo de Disponibilidade.

A taxa de reparo pode afetar drasticamente a disponibilidade de um sistema.

A disponibilidade de um sistema $A(t)$, é definida como a probabilidade do sistema estar disponível para desempenhar suas tarefas no instante de tempo t .

Intuitivamente, a disponibilidade pode ser aproximada como o tempo total que um sistema está operacional dividido pelo tempo total que o sistema foi colocado em operação. Em outras palavras, a disponibilidade é a porcentagem do tempo que o sistema está disponível para realizar as operações esperadas.

$$A(t_{\text{corrente}}) = \frac{\text{tempo operacional}}{\text{tempo operacional} + \text{tempo de reparo}}$$

A avaliação experimental é muito difícil devido ao tempo e custo envolvido.

Deve-se então, considerar duas abordagens. A primeira é baseada em parâmetros como MTTF e MTTR e define a chamada Disponibilidade Estável ("Steady - state availability" - ASS).

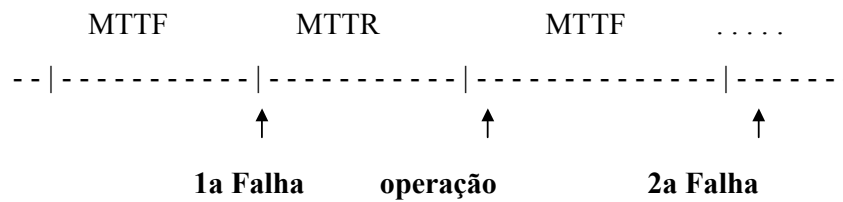


Figura 6.24

$$ASS = \frac{N (MTTF)}{N (MTTF) + N (MTTR)} = \frac{MTTF}{MTTF + MTTR}$$

$$MTTF = \frac{1}{\lambda}, \quad MTTR = \frac{1}{\mu} \quad (P/\text{sistema simples})$$

$$Ass = \frac{1}{\lambda + \frac{1}{\mu}}$$

$$\left. \begin{array}{l} \text{Exemplo : } \lambda = 0,01 \text{ falhas/ hora} \\ \mu = 0,1 \text{ reparos/ hora} \end{array} \right\} Ass = 0.90909$$

A outra abordagem usa a taxa de falhas e a taxa de reparo no Modelo de Markov para calcular a disponibilidade em função do tempo.

Considere agora o diagrama de Markov para um sistema simples, com reparo:

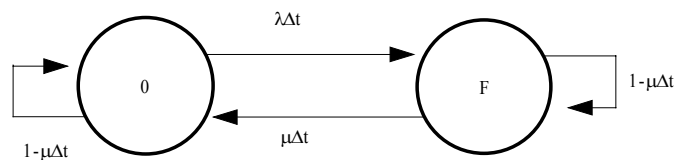


Figura 6.25

$$p_0(t+\Delta t) = p_0(t)(1-\lambda\Delta t) + p_F(t) \cdot \mu \Delta t$$

$$p_F(t+\Delta t) = p_0(t) \lambda \Delta t + p_F(t) \cdot (1-\mu\Delta t)$$

Po (t) probabilidade de o sistema estar operacional no instante t
(Disponibilidade)

$$p_o(t) = \frac{\mu}{\lambda + \mu} + \frac{\lambda}{\lambda + \mu} \cdot e^{-(\lambda + \mu)t}$$

$$\text{Quando } t \rightarrow \infty \rightarrow p_o(\infty) = \frac{\mu}{\lambda + \mu} = \frac{1}{1 + \frac{\lambda}{\mu}} = \text{ASS}$$

Para $\lambda = 0,01$ falhas/h e $\mu = 0,01$ reparos/h o gráfico de disponibilidade está apresentado a seguir.

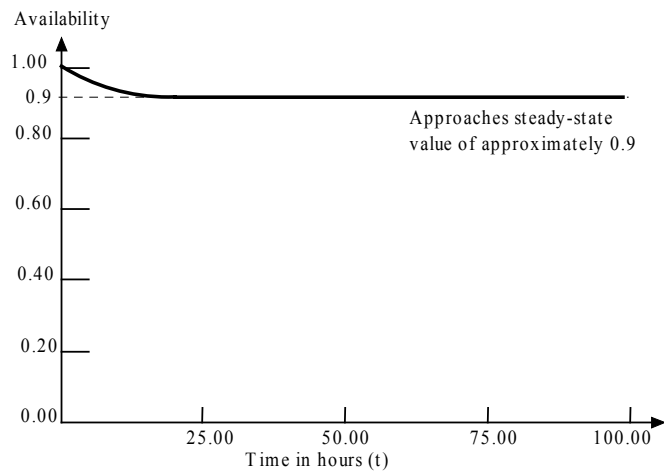


Figura 6.26

6.8.2.6. Modelo de Manutenibilidade

A Manutenibilidade $M(t)$ é a probabilidade de um sistema falho ser reparado dentro de um tempo especificado t .

Um parâmetro importante aqui é a taxa de reparo. A taxa de reparo é o número médio de reparos que pode ser realizado por unidade de tempo.

O inverso da taxa de reparo é o MTTR, que é o tempo médio requerido para realizar um único reparo.

$$MTTR = \frac{1}{\mu}$$

Um sistema pode ser construído e defeitos injetados. O tempo médio requerido para reparar o sistema é medido e anotado como MTTR. Uma boa estimativa do MTTR pode ser obtida apenas se um número suficiente de defeitos diferentes são injetados e pessoal de reparo, de níveis variados, são utilizados.

Suponha que se tenham N sistemas. Injeta-se um único defeito em cada sistema e uma pessoa da manutenção vai reparar cada sistema. Os defeitos são injetados no instante $t = 0$.

No instante t , determinado-se $N_r(t)$ o número de sistema reparados e $N_{nr}(t)$ o número de sistemas não reparados.

$$M(t) = \frac{N_r(t)}{N} = \frac{N_r(t)}{N_r(t) + N_{nr}(t)}$$

$$\frac{dM(t)}{dt} = \frac{1}{N} \frac{dN_r(t)}{dt}$$

$$\frac{dN_r(t)}{dt} = N \cdot \frac{dM(t)}{dt}$$

Define-se $\frac{1}{N_{nr}(t)} \cdot \frac{dN_r(t)}{dt} = \mu$ é a função taxa de reparo

$$\mu = \frac{1}{N_{nr}(t)} \cdot \frac{dN_r(t)}{dt} = \frac{N}{N_{nr}(t)} \cdot \frac{dM(t)}{dt}$$

$$\frac{dM(t)}{dt} = \mu \cdot \frac{N_{nr}(t)}{N}$$

$$\frac{N_r(t)}{N} = M(t) = \frac{N_{nr}(t)}{N}$$

$$M(t) = 1 - \frac{N_{nr}(t)}{N} \rightarrow \frac{N_{nr}(t)}{N} = 1 - M(t)$$

$$\frac{dM(t)}{dt} = \mu (1 - M(t))$$

$$\rightarrow M(t) = 1 - e^{-\mu t}$$

Se $\mu = 0 \rightarrow M(t) = 0$ (O sistema não pode ser reparado)

Se $\mu = \infty \rightarrow M(t) = 1$ (A manutenção pode ser realizada no tempo zero)

Quando $t = MTTR$

$$M(MTTR) = 1 - e^{-(\mu/\mu)} = 1 - e^{-1}$$

$$M(MTTR) = 0,632, \text{ ou seja,}$$

O sistema apresenta a probabilidade de 0,632 de ser reparado num tempo menor ou igual ao MTTR.

Como a manutenção pode diferenciar muito em função do tipo de falha, a taxa de reparo é tipicamente especificada por nível de reparo.

A Divisão mais comum apresenta três níveis de reparo:

- Nível organizacional (menos de 1 hora) : todos reparos que podem ser realizados no local do sistema;

Ex: troca de cartões;

- Nível intermediário (algumas horas) : todos reparos são realizados próximos ao sistema, num laboratório;

- Nível de fábrica (alguns dias): o equipamento deve ser reparado na fábrica.

Exemplo: Para um sistema de computador, tem-se 2 horas de reparo no nível organizacional, 8 horas no nível intermediário e 168 horas (1 semana) no nível de fábrica.

O gráfico a seguir apresenta a diferença entre a Manutenabilidade em cada caso.

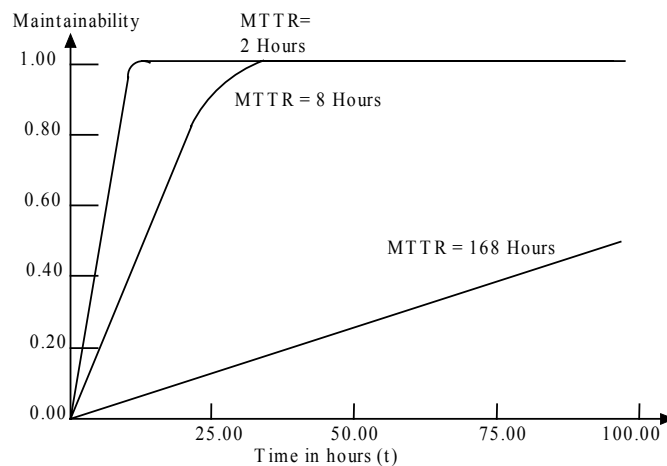


Figura 6.27

7. Segurança

Este capítulo tem como objetivo a apresentação dos principais conceitos na área de segurança, as causas fundamentais dos acidentes, envolvendo aspectos gerenciais, técnicos e humanos, além da importância da cultura de segurança como uma atividade presente em todo ciclo de vida de um sistema crítico. É apresentada também uma sugestão de padronização de terminologia discutida e utilizada dentro do GAS.

7.1. Aspectos Conceituais

O conceito de segurança, dentro deste trabalho, pode ser definido como a probabilidade de um sistema desempenhar, num determinado período de tempo, suas funções ou descontinuí-las sem causar mortes, danos à saúde, destruição de propriedades, perda de missão ou danos ao meio ambiente. Neste sentido, o termo segurança, neste trabalho, corresponde ao termo inglês “safety”.

Para que se possa garantir a segurança de um sistema, devem ser utilizadas técnicas que permitam prevenir os acidentes previsíveis, bem como minimizar as consequências daqueles imprevisíveis. Em um acidente, são consideradas as perdas em geral, tais como destruição de propriedade, cancelamento de missões ou danos causados ao ambiente, além de ferimentos ou morte causados em seres humanos. O desenvolvimento desses sistemas considerados críticos é, normalmente, controlado por regulamentação governamental, que estabelece critérios de certificação de sistemas em cada área de aplicação.

Quando uma perda é considerada como grave ou de vulto, geralmente justifica os esforços e os recursos a serem investidos para sua prevenção. O valor a ser investido é considerado válido ou justificável, considerando-se tanto os aspectos técnicos, quanto outros fatores, tais como sociais, psicológicos, políticos e econômicos.

As atividades relativas à segurança devem se iniciar quando o sistema começa a ser concebido e ter sequência no projeto, produção, teste e operação do sistema. A segurança deve ser considerada como um todo, ou seja, não é suficiente que se assegure apenas a correção de partes ou de sub-sistemas de um sistema maior.

Desta forma, é de fundamental importância ter-se uma atividade de garantia da segurança atuante em todas as fases do ciclo de vida de um sistema, bem como os problemas, apontados durante a análise de risco, devem ser seriamente considerados, nunca se desprezando qualquer indício ou suspeita que possa vir a provocar um acidente. Daí a importância em se adotar e desenvolver metodologias e métodos que cada vez mais assegurem a qualidade dos trabalhos desenvolvidos na garantia da segurança de sistemas, de forma que se obtenham sistemas de funcionamento robusto.

No contexto dos sistemas críticos quanto à segurança, os estados inseguros correspondem àqueles estados também denominados perigosos, onde o sistema está exposto à ocorrência de um acidente. Neste trabalho é adotado o termo “estado perigoso”. Um projeto que tenha de contemplar aspectos referentes à segurança de um sistema deve, em primeiro lugar, buscar a eliminação de estados perigosos. Se isto não for possível, deve ser alcançado o controle desses estados perigosos preferencialmente por meio de dispositivos passivos, baseados em processos físicos, tais como a força da gravidade. Novamente, se não houver possibilidade desse controle, deve ser buscada a redução de eventuais danos causados pela ocorrência do estado perigoso. A segurança de um sistema deve ser eficaz, evitando ou minimizando a ocorrência de acidentes, além de ter um custo compatível com o sistema considerado como um todo. Sua validade só é comprovada se acidentes forem realmente prevenidos ou evitados.

Algumas vezes, a segurança de um sistema atua como uma restrição aos projetos, pois alguns de seus requisitos podem entrar em conflito com aspectos operacionais e de desempenho, bem como podem ocasionar aumentos nos custos envolvidos.

Um acidente pode acontecer se houver alguma falha ou entrada imprópria não prevista ou não coberta pelos dispositivos que deveriam garantir a segurança de um sistema. Outra causa da ocorrência de acidentes deve-se ao fator humano, ou seja, uma falha de operação por parte de um ser humano, que ocasione uma condição que possa levar a um acidente, condição esta não prevista ou não coberta pelo projeto e pela implementação do sistema.

Desta forma, as causas que justificam a ocorrência de um acidente podem ter origem em deficiências na cultura de segurança das instituições, em falhas na estrutura organizacional dessas mesmas instituições ou ainda em atividades técnicas superficiais ou ineficientes, relativas à segurança.

Cada uma dessas formas é a seguir discutida.

a) Aspectos relacionados com as deficiências na cultura de segurança das organizações.

Requisitos desejáveis em um sistema podem ser conflitantes entre si, sendo necessário, portanto, estabelecer compromissos para o desenvolvimento do projeto. Neste aspecto, pode ocorrer que a melhor ou mais avançada tecnologia seja invalidada por decisões incorretas. Pode-se citar, como exemplo, o conflito entre os fatores disponibilidade e segurança. Muitas vezes as pressões existentes podem forçar o comprometimento da segurança em função da obtenção de maiores níveis de disponibilidade. Outro campo de pesquisa e discussão que vem ganhando destaque refere-se ao confronto entre os conceitos de “Safety” e “Security”, ambos denominados “segurança” na língua portuguesa. Com a tendência crescente da utilização de sistemas integrados via redes de computadores em aplicações críticas de segurança, pode-se haver conflitos entre requisitos de “security” e requisitos de “safety”. Se os requisitos de “safety” e “security” forem definidos isoladamente, há o perigo que incongruências não reconhecidas e discutidas, e portanto não resolvidas, possam ocorrer, comprometendo a segurança final do sistema. Talvez a forma de integração mais apropriada seja a harmonização entre os processos de verificação de cada um desses aspectos, permitindo que os conflitos entre ambos sejam reconhecidos e resolvidos antecipadamente. [Eames 99]

Outro aspecto relacionado com a cultura de segurança refere-se à complacência e autoconfiança. As pessoas normalmente desenvolvem uma mentalidade sobre a infalibilidade do equipamento a que estão acostumadas a lidar, fruto de repetidas afirmações sobre a garantia da tecnologia utilizada para aumentar a segurança do sistema. Acredita-se que um acidente não possa ocorrer, pois não seria possível que houvesse a ocorrência de tantas condições adversas simultaneamente, o que nem sempre é verdade. Os piores acidentes

ocorrem quando não se espera que possam vir a acontecer, pois geralmente se gera uma acomodação por parte das pessoas. Ao contrário, quando se acredita na possibilidade de ocorrência de um acidente, são tomadas providências para preveni-lo ou minimizar seus efeitos.

As técnicas de redundância e diversidade de projetos, utilizadas em alguns sistemas para prevenir acidentes e aumentar a segurança, também não devem ser encaradas como a solução ótima para todos os casos. Muitas vezes, há as falhas de modo comum, que podem afetar todos os canais ou parâmetros ao mesmo tempo.

Normalmente as condições perigosas mais evidentes são as que recebem maior atenção e são, por conseguinte, controladas, enquanto que aquelas condições com menor probabilidade de ocorrência são desprezadas. É comum se verificar que, após a ocorrência de um acidente, sua causa era um evento conhecido, que foi desprezado, considerando sua ocorrência como improvável. Outro fator importante a ser considerado no que diz respeito à complacência assumida é o caso em que um sistema opera por longos períodos de tempo sem falhas. Neste caso, acredita-se que o sistema não irá falhar nunca mais, o que não é verdade. De certo modo, o risco da ocorrência de um acidente pode até aumentar, devido a alterações no ambiente em que o sistema estiver inserido.

Também não podem ser ignorados sinais de alarme, pois os acidentes são freqüentemente precedidos por alertas, ou por uma série de ocorrências menores, geralmente ignoradas, pois não se acredita que algo mais grave ainda possa vir a acontecer.

b) Aspectos relacionados com problemas na estrutura organizacional das instituições.

A busca pela segurança deve vir desde os mais altos escalões de uma organização, difundindo-se em todos os setores da instituição. Agências governamentais e grupos de usuários podem tentar fazer com que a segurança seja mais seriamente considerada, o que só se tornará mais eficaz se houver uma conscientização, por parte da sociedade em geral, a respeito da importância fundamental das atividades que garantam a segurança dos sistemas considerados mais críticos.

De particular interesse são os sistemas compostos por diversos sub-sistemas, onde é necessário que haja um órgão centralizador de todos os grupos envolvidos no desenvolvimento do sistema, de forma que se atribua qual, ou quais grupos devem se preocupar com atributos de segurança, principalmente nas interfaces entre os sub-sistemas.

Outro ponto a ser considerado com relação ao grupo responsável pela segurança, é que deve ser independente, no que diz respeito às equipes de projeto. De preferência é recomendável que tal grupo seja vinculado a entidades completamente independentes daquelas que estiverem realizando o projeto.

Devem existir canais de comunicação adequados, entre os diversos grupos envolvidos com o sistema, de forma tal que as metas desejáveis possam ser transmitidas dos níveis hierárquicos mais altos para os mais baixos, e no sentido contrário, permitindo a avaliação sobre a evolução do projeto.

c) Problemas decorridos de atividades técnicas superficiais ou ineficientes quanto à segurança.

As condições perigosas devem ser cuidadosamente analisadas, com registros que justifiquem e suportem cada decisão de projeto, bem como os compromissos assumidos entre demais fatores e segurança.

Muitas vezes, esforços no sentido de garantir a segurança não são eficazes, pois são eliminadas as causas específicas de acidentes, mas não as causas básicas. Outro ponto que ocorre é que o projeto pode ser baseado em falsas suposições, como por exemplo, independência entre os eventos. Pode acontecer ainda de que eventuais modificações para aumentar a segurança acabem por ocasionar efeito contrário, ou seja, incrementar o número de estados perigosos devido ao aumento da complexidade do sistema. Pode ocorrer que a colocação de um dispositivo de segurança instalado para corrigir determinada condição, seja utilizado para justificar redução em margens de segurança em outros aspectos.

Outro fator muito importante é a realimentação com as informações sobre os acidentes ocorridos em outros sistemas, similares ou não, ao que estiver sendo desenvolvido, pois essas informações podem e devem ser efetivamente utilizadas na prevenção de novos acidentes.

7.2. Erro Humano

Muitos fatores podem levar o ser humano a agir de forma incorreta nos sistemas críticos quanto à segurança. Neste aspecto está sendo feita referência especial aos Operadores. Podem ser citados, como exemplo, o fornecimento ao operador de dados incompletos, incorretos, complexos ou excessivos. Outra crença negativa é que os operadores podem superar qualquer emergência, forçando-os a intervir em situações limites.

Além destes aspectos, muitas vezes o operador é responsabilizado por acidentes como forma de negligenciar ou até mesmo esconder erros cometidos por projetistas e gerentes. Na grande maioria das vezes, as ações positivas dos operadores raramente são destacadas, sendo registradas apenas as ações negativas.

Tendo em mente todas essas dificuldades, é apresentado a seguir uma breve discussão sobre a necessidade dos operadores nos sistemas automáticos, os modelos cognitivos da tarefa humana e os possíveis papéis dos operadores nestes sistemas.[Leveson 95]

a) A Necessidade do Operador em Sistemas Críticos quanto à Segurança

Uma das grandes questões que surge se refere à eliminação ou não do operador dos sistemas críticos, substituindo-os por computadores. Diversas razões corroboram para que o operador não seja eliminado desses sistemas. São discutidos a seguir alguns destes aspectos.

Pode-se afirmar que é extremamente difícil antecipar todas as condições e todas as interações não desejadas entre os componentes, que podem ocorrer no ambiente de um sistema. A presença do operador nestes casos pode diminuir o risco de um acidente.

Outro aspecto importante refere-se aos eventuais erros existentes num determinado sistema provocados por erros de seus projetistas. Embora os projetistas trabalhem sob condições de menor pressão que os operadores, eles também cometem erros. Os erros relacionados com os projetistas estão focados em aspectos como, dificuldade em alocar probabilidades em eventos raros, não consideração de efeitos colaterais, não atenção para medidas contingenciais, controle da complexidade do sistema, concentrando-se apenas em alguns aspectos do problema, capacidade limitada de compreender

relações complexas, além de dificuldades em se ter uma visão ampla do sistema, especialmente no que diz respeito aos sistemas críticos computacionais.

Desta forma, é importante a participação de operadores nestes sistemas. A questão que se deve discutir é qual deve ser o seu papel nestes sistemas

b) O papel do ser humano nos Sistemas Críticos quanto à Segurança

A automação de sistemas provoca o reposicionamento do ser humano em novos níveis de complexidade, com um maior nível de controle e supervisão, refletindo, desta forma, maiores níveis de tomada de decisões. Neste sentido, aumenta-se o nível de centralização, tornando a base de decisão extremamente mais complexa. Desta forma, o estudo da confiabilidade humana e da ergonomia passam a ter um papel fundamental dentro do estudo da segurança dos sistemas computacionais. Nessa relação do operador com os sistemas computacionais o ser-humano pode assumir basicamente três papéis: Monitor, Back-Up ou Parceiro.

A experiência demonstra que o ser humano apresenta um desempenho fraco como monitor em sistemas automatizados. Nesta situação o operador está extremamente dependente de informações fornecidas pelo sistema, que podem ser disponibilizadas em grande quantidade e de forma não adequada. Vale ressaltar, nessa situação, que as tarefas, que requerem baixa atividade do operador, podem implicar numa baixa vigilância de sua parte, podendo conduzi-lo a atitudes de super confiança ou complacência em relação ao sistema automatizado.

Quando o operador desempenha o papel de Back-up, o projeto do sistema de automação pode tornar-se de difícil gerenciamento durante uma situação emergência ou em sistemas complexos. Nestas situações o operador pode ter muito pouco tempo para decidir, além do sistema oferecer diversas opções de escolha. Adiciona-se a isto, o fato do operador estar inativo por longos períodos, dificultando ainda mais sua tomada de decisão.

Já quando o operador tem o papel de Parceiro dentro do sistema crítico, ele irá realizar tarefas que não puderam ser automatizadas por algum motivo ou intencionalmente alocadas a ele. Neste modo de trabalho, podem surgir algumas vantagens e desvantagens que devem ser avaliadas em cada caso.

Podem surgir tarefas extremamente complexas e com muita arbitrariedade na tomada de decisão. Por outro lado, o operador pode agir com maior criatividade e utilizando seus conhecimentos a respeito do sistema, se sentindo, desta forma, realmente integrado no processo de automação.

Nesse sentido, o foco, com relação ao Operador, dentro um projeto de um sistema crítico quanto à segurança, deve ser sua Operação e não sua Função. O Operador deve ser envolvido num processo de tomada de decisões do projeto e da sua análise de risco. Quando o operador atua como parceiro, ele se sente realmente integrado e motivado para atuar no âmbito da responsabilidade deste tipo de sistema.

8. Metodologia de Análise de Risco Proposta

Neste capítulo é apresentada uma Metodologia de Análise de Risco para sistemas críticos quanto à segurança. Esta metodologia foi desenvolvida através da sinergia de dois aspectos fundamentais: a pesquisa acadêmica e a identificação de necessidades através de experiências práticas em análise de risco de sistemas críticos na área de transporte metroviário e aeroviário. No entanto, esta metodologia pode ser aplicada a outros sistemas críticos quanto à segurança através da avaliação e a adequação às características das outras áreas de aplicação. Trata-se de uma via de duas mãos. Tanto a pesquisa acadêmica abre novas possibilidades para a resolução de diversos problemas práticos, como a experiência auxilia em prover um maior direcionamento nas pesquisas acadêmicas sendo realizadas.

A metodologia, aqui apresentada, é parte integrante de um processo mais amplo denominado Análise de Segurança, constituído pelo Gerenciamento da Segurança e pela Análise de Risco, propriamente dita. O aspecto de gerenciamento da segurança não é o foco principal desta pesquisa.

Dentro da metodologia de Análise de Risco são destacadas as seguintes etapas:

- Definição e descrição do sistema, suas interfaces e demais informações necessárias para a Análise de Risco;
- Realização do processo de Análise de Perigo;
- Qualificação do Risco Residual;
- Avaliação da severidade dos acidentes relacionados com o estado perigoso; e
- Realimentação e Avaliação da experiência operacional.

No que se refere ao processo de Análise de Perigo são apresentadas as seguintes etapas:

- Análise Preliminar de Perigo;
- Análise de Perigo do Sistema;
- Análise de Perigo do Subsistema;
- Análise de Perigo da Operação e Suporte; e
- Análise Final de Perigo

Finalizando são apresentados alguns métodos que podem ser utilizados durante o processo de Análise de Perigo.

8.1. Gerenciamento da Segurança

O primeiro aspecto em um sistema de segurança corresponde ao gerenciamento da segurança. Nesse sentido, a discussão sobre as questões de segurança deve criar uma atmosfera de cooperação e não acusação entre os diversos grupos técnicos. Este deve ser um cuidado fundamental para o bom andamento dos trabalhos.

Outro ponto importante refere-se à definição de políticas de segurança que definam claramente a relação entre a Segurança (“Safety”) e outros objetivos do sistema sendo avaliado, tais como, Confiabilidade, Disponibilidade, Segurança (“Security”) e Custos, entre outros.

Para que estes aspectos sejam efetivamente implementados torna-se necessária a existência de um Grupo de Segurança responsável por formular, difundir e implantar a política de segurança no âmbito do projeto do sistema crítico, documentar o rastreamento dos perigos e suas relações, adaptar e desenvolver normas específicas, realizar a análise de perigo, planejar e monitorar as tarefas dos testes de segurança, participar de revisões do programa, realizar intercâmbios com outros grupos de segurança e investigar e analisar acidentes. Evidentemente este grupo deve ser tão mais independente quanto maior for o nível de risco envolvido no sistema computacional sendo implantado ou avaliado. A política de segurança estará refletida através de um Plano Global de Segurança, destacando, dentro deste, um Plano Específico para a Segurança do Software, em função de sua grande complexidade. Este Plano Global de Segurança deve destacar a organização da segurança do sistema, o cronograma do programa de segurança, os critérios de segurança além das atividades que devem ser realizadas ao longo do desenvolvimento do sistema visando a comprovação da segurança. [Hall 97]

Em função das grandes responsabilidades envolvidas nas atividades deste Grupo de Segurança, existe uma tendência mundial em se exigir uma maior qualificação dos profissionais participantes dessas atividades. Este ponto não elimina de maneira nenhuma a responsabilidade de todas as demais equipes envolvidas no projeto, implantação, operação e manutenção do sistema de

segurança. [Renn 98] Vale ressaltar que o atributo segurança é consequência de uma cooperação decorrente de um programa global de segurança. Como este Grupo de Segurança deverá manter intercâmbios com as demais equipes envolvidas no sistema de segurança, canais de comunicação devem ser estabelecidos permitindo, desta forma, uma maior rapidez e eficácia.

8.2. Análise de Risco

A Análise de Risco avalia a importância relativa do perigo e permite avaliar sua aceitabilidade ou não. Na realidade, em diversos momentos de nossas vidas, estão sendo avaliados subjetivamente os riscos que podemos estar correndo.[Bohnenblust 98] No sentido de uma melhor compreensão da natureza do risco é extremamente útil considerar a relação entre Perigos e Acidentes, que resultam em um dano à pessoa humana ou ao meio ambiente. Considera-se que o perigo representa uma situação de potencial acidente.

O Risco associado a um perigo é determinado pela combinação de dois fatores: a frequência ou probabilidade da ocorrência de um acidente e a severidade da consequência envolvida no acidente. O Risco pode ser avaliado qualitativamente e quantitativamente. As escalas quantitativas podem variar de aplicação para aplicação, apesar do esforço de algumas normas internacionais em padronizar essa escala. [Storey 96] [Ladkin 01]

Um determinado Risco não é aceitável, se existir um determinado perigo com alta probabilidade de ocorrência e consequências desastrosas. No entanto, poder-se-ia aceitar um risco, em relação a um determinado perigo, com baixíssima probabilidade, apesar de apresentar consequências altamente danosas. O nível de aceitabilidade do risco é determinado pelo benefício associado ao Risco, e pelo esforço requerido em diminuí-lo.

A redução necessária de risco deve ser alcançada para se manter o risco tolerável em uma determinada situação. O conceito de redução necessária de risco é de fundamental importância na avaliação de sua aceitabilidade. [DD ENV 50129:1999]

A norma IEC 61508-1 define níveis de integridade de segurança SIL – “Safety Integrity Level” para sistemas que operam em regime de baixa demanda e sistemas que operam em regime de alta demanda ou de modo contínuo. [IEC 61508]

Um sistema opera em regime de baixa demanda, se a frequência com a qual ele for solicitado não for maior que uma vez por ano e não for maior que duas vezes a frequência com a qual ele é verificado, ou seja, sofre um processo de manutenção preventiva. Caso contrário, considera-se que o sistema tem um regime de operação de alta demanda ou de modo contínuo.

Para sistemas que operam em baixa demanda, são definidos os níveis de integridade de segurança em termos de valores limites da probabilidade média de falha ao executar a função para a qual foi projetado na demanda. Para sistemas que operam em regime de alta demanda os níveis estão em termos de valores limites para a probabilidade de ocorrência de falhas inseguras por hora.

A tabela 8.1 apresenta os 4 (quatro) níveis SIL existentes para os dois tipos de modo de operação citados.

SIL	Baixa Demanda (falha insegura por demanda)	Alta Demanda (falha insegura por hora)
4	$10^{-5} \leq \lambda \leq 10^{-4}$	$10^{-9} \leq \lambda \leq 10^{-8}$
3	$10^{-4} \leq \lambda \leq 10^{-3}$	$10^{-8} \leq \lambda \leq 10^{-7}$
2	$10^{-3} \leq \lambda \leq 10^{-2}$	$10^{-7} \leq \lambda \leq 10^{-6}$
1	$10^{-2} \leq \lambda \leq 10^{-1}$	$10^{-6} \leq \lambda \leq 10^{-5}$

Tabela 8.1 – Níveis de Integridade de Segurança

Um dos métodos de determinação do risco aceitável é denominado ALARP - “As Low As Reasonable Practible”. [Melchers 01] A aplicação do princípio ALARP significa tentar reduzir o nível de risco de um sistema a valores tão baixos quanto a relação entre o ganho e o investimento for aceitável. Esta técnica é usada principalmente no Reino Unido.

Em uma aplicação, os níveis de risco são classificados da seguinte forma:

- a) O Risco é tão grande que não deve ser tolerado; ou
- b) O Risco é ou tornou-se tão pequeno, tornando-se insignificante; ou
- c) O Risco está entre os dois estados especificados nos itens **a** e **b**, tendo sido reduzido ao mais baixo nível praticável, tendo em vista os benefícios resultantes de sua aceitação e os custos de qualquer redução adicional.

Com respeito ao item c, o princípio ALARP requer que qualquer risco deva ser reduzido, o quanto for razoavelmente praticável, para um nível tão baixo quanto razoavelmente aceitável.

As três regiões são mostradas na figura 8.1.

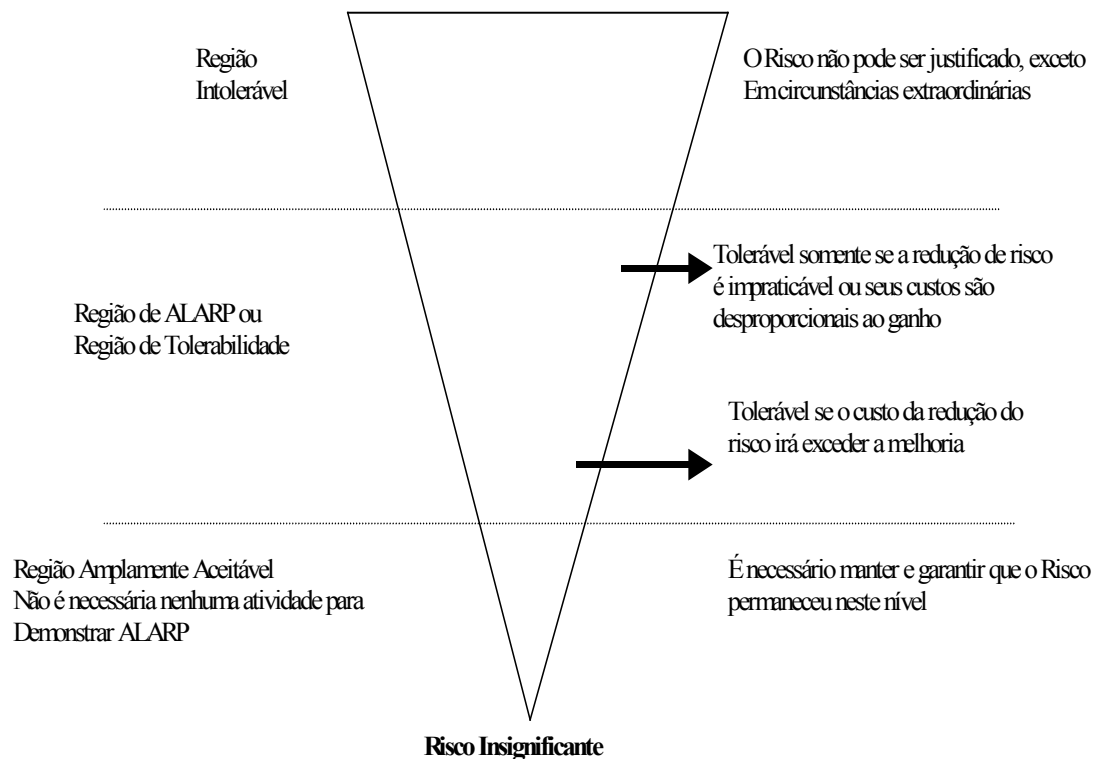


Figura 8.1 – Regiões de Risco

Acima de um certo nível, um Risco é considerado intolerável e não pode ser justificado em qualquer circunstância ordinária. Neste caso, ele está situado na região intolerável.

Abaixo desse nível, há a região de tolerabilidade, onde se permite que uma atividade ocorra com seus Riscos associados, que são tão baixos quanto o razoavelmente praticável. Ser tolerável significa conviver com Riscos, obtendo os benefícios do funcionamento do sistema. Ao mesmo tempo, espera-se que este nível de Risco seja mantido sob constante acompanhamento, sendo reduzido como e quando isto possa ser feito.

Uma avaliação de custo e benefício é exigida explícita ou implicitamente de forma a se avaliar o custo e a necessidade de melhorias, ou ainda para se avaliar a necessidade de medidas de segurança adicionais.

Um trabalho fundamental refere-se à determinação dos níveis SIL. De acordo com a norma IEC 61508-5, os níveis SIL dependem dos riscos aos quais a aplicação está sujeita. Estes riscos devem levar em conta as seguintes consequências:

- Perdas de Vidas Humanas ou de outras vidas (animais);
- Ferimentos ou doenças em pessoas;
- Poluição Ambiental; e
- Perdas ou danos à propriedade.

A norma IEC 61508-5 apresenta alguns métodos para a determinação do nível SIL. Um determinado tipo de avaliação pode ser realizado através de um gráfico de risco, apresentado na figura 8.2, que leva em consideração os seguintes parâmetros:

- Consequência do evento perigoso (**C_i**)
- Frequência e tempo de exposição ao perigo (**F_i**)
- A possibilidade de evitar o evento perigoso (**P_i**)
- A probabilidade de ocorrência indesejável, como um acidente, não considerando a existência do Sistema de Intertravamento (**W_i**)

Nessa figura os parâmetros **a, b, c, d, e, f, g, h** estão relacionados com o Nível de Integridade de Segurança requerido, ou seja, representam a mínima redução necessária do risco.

			W3	W2	W1
C1			a	-	-
C2	F1	P1	b	a	-
		P2	c	b	a
	F2	P1	d	c	b
		P2	e	d	c
C3	F1		f	e	d
	F2		g	f	e
C4			h	g	f

Figura 8.2 – Gráfico de Risco – Norma IEC 61508 -5

A tabela 8.2 apresenta a classificação dos níveis desses parâmetros:

Parâmetro de Risco	Classificação
Consequência (C)	C1 – Perdas Pequenas C2 – Perdas sérias e permanentes para uma ou mais pessoas, ou morte de uma pessoa C3 – Morte de várias pessoas C4 – Muitas pessoas assassinadas
Frequência e tempo de exposição ao perigo (F)	F1 – Exposição de rara a freqüente na região de perigo F2 – Exposição de freqüente a permanente na região de perigo
Possibilidade de evitar o evento perigoso (P)	P1 – Possível sob certas condições P2 – Quase impossível
Probabilidade de ocorrência de eventos indesejáveis (W)	W1 – Probabilidade muito pequena de que as ocorrências indesejáveis acontecerão W2 – Probabilidade pequena de que as ocorrências indesejáveis acontecerão W3 – Probabilidade relativamente alta de que as ocorrências indesejáveis acontecerão

Tabela 8.2 – Classificação dos Níveis do Gráfico de Risco – Norma IEC

61508 -5

Vale ressaltar que, o propósito do fator W é estimar a probabilidade de ocorrência dos eventos indesejáveis não se considerando a existência do Sistema de Intertravamento.

Uma vez classificados, através da tabela anterior, os níveis dos parâmetros **C**, **P**, **F** e **W**, é possível determinar, a partir do Gráfico de Risco, um parâmetro que define a redução mínima do risco necessário para a aplicação.

A tabela 8.3 associa o Gráfico de Risco com o Nível de Integridade de Segurança – SIL requerido para a aplicação.

Redução para um Risco Mínimo	Nível de Integridade de Segurança
-	Sem requisito de segurança
a	Sem requisitos especiais de segurança
b,c	1
d	2
e,f	3
g	4
h	Sistemas de proteção não são suficientes

Tabela 8.3 – Estabelecimento do Nível de Integridade de Segurança – Norma IEC 61508-5

Num sistema de sinalização metroviário, a consequência de um evento perigoso pode apresentar características catastróficas (nível **C3**) com exposição de freqüente a permanente na região de perigo (nível **F2**), podendo apresentar uma probabilidade relativamente alta de ocorrência indevida quando da não existência do Sistema de Intertravamento (nível **W3**). Desta forma, o parâmetro selecionado através da figura 8.2 corresponde à letra **g**. Através da tabela 8.3 pode, portanto, estabelecer o SIL nível 4 para um Sistema de Sinalização Metroviário.

Após a determinação dos Níveis de Integridade de Segurança, pode-se iniciar o processo de Análise de Risco. [Leveson 95] [Storey 96] [NASA 96],[BS EN 50126:1999] Para tal são necessárias as seguintes etapas:

- Definição e Descrição do Sistema, Interfaces e demais informações necessárias para a Análise de Risco
- Realização do processo de **Análise de Perigo**
- Qualificação do Risco Residual
- Caso o Nível de Risco residual não seja aceitável, redução da severidade dos acidentes relacionados com o estado perigoso, ou da probabilidade de sua ocorrência
- Realimentação e Avaliação da experiência operacional

De forma a exemplificar os conceitos apresentados, considere-se um sistema de controle de uma cancela numa passagem de nível. Dentro do processo de Análise de Risco padroniza-se como Análise de Perigo o processo de

determinação da probabilidade de se atingir um estado perigoso. Neste sistema o estado perigoso corresponde ao sistema falhar de forma a não abaixar a cancela quando da aproximação de um trem. Na realidade, se este estado perigoso for alcançado, não obrigatoriamente ocorrerá um acidente. Outros fatores deverão ser considerados. A partir deste estado perigoso, pode-se determinar a probabilidade da ocorrência de um acidente e sua severidade. Se o risco residual não for aceitável deve-se então trabalhar no sentido de diminuí-lo, seja através da diminuição de sua probabilidade ou diminuição da severidade envolvida.

A Análise de Risco pode ser exigida através de um processo denominado de **Certificação**. [Sotrey 96]

8.3. Certificação

Trata-se do processo de emitir um Certificado indicando conformidade com uma norma, um conjunto de recomendações ou algum documento similar. Qualquer Organização ou Indivíduo pode emitir um Certificado, e sua importância irá variar muito com a natureza do objeto certificado. Em alguns casos, pode-se exigir um Certificado devido às exigências legais. Nestes casos o Certificado tem o papel de uma Licença de uma Autoridade Regulamentada. Esta necessidade para sistemas críticos quanto à segurança varia muito conforme o país.

Em áreas não cobertas por exigências legais, o Certificado pode ter, por exemplo, uma importância comercial. Muitas indústrias possuem uma Autoridade Reguladora que governa todos os projetos dentro de um determinado setor.

Por exemplo, no setor de aviação civil do Reino Unido, a entidade certificadora é a “Civil Aviation Authority - CAA” e dos Estados Unidos, a “Federal Aviation Authority – FAA”.

A Certificação pode ser aplicada a Organizações e Indivíduos, a Ferramentas e Métodos e a Sistemas e Produtos.

Com o objetivo de obter certificação, o projetista de um produto crítico deve provar, ao Órgão Certificador, a segurança de seu produto. O projetista deve ser capaz de mostrar que todos os perigos foram identificados e tratados adequadamente e que a integridade do sistema é apropriada para aquela

aplicação, ou seja, atende aos níveis de integridade de segurança exigidos. O trabalho envolvido num processo de Certificação é bastante grande e requer um planejamento cuidadoso.

A Certificação pode ser realizada sobre Organizações e Indivíduos, sobre Ferramentas e Métodos e sobre Sistemas e Produtos, esta última forma a mais conhecida. É apresentada, a seguir, uma breve descrição destes tipos de Certificação.

a) Certificação de Organizações e Indivíduos

Esta Certificação tem como objetivo estabelecer a competência de uma organização em uma área específica de atividade. Existe um paralelismo com a garantia de qualidade através da ISO9000.

A certificação pode ser aplicada também a indivíduos. Estes devem ser certificados com o objetivo de serem autorizados a desempenhar uma determinada profissão.

Poucas indústrias e organizações, possuem alguma forma de certificação de profissionais para trabalhar com o desenvolvimento, teste e avaliação de sistemas críticos quanto à segurança. Como exemplo, pode-se citar que a FAA delega a Certificação de Profissionais para as DER's - "Designated Engineering Representatives". A Boeing e a Mc Donnell Douglas possuem 90 a 95% de suas atividades certificadas pelas DER's. Desta forma, a Certificação pode também ser aplicada a operadores de sistemas críticos quanto à segurança.

b) Certificação de Ferramentas e Métodos

As ferramentas e os métodos de desenvolvimento utilizados na produção de sistemas críticos de segurança desempenham um papel fundamental na obtenção destes sistemas.

No caso do software, um maior nível de qualidade do processo é exigido. Pode-se citar, como exemplo, a exigência de um maior nível CMM, correspondente a um maior de segurança SIL. [Myerson 96]

c) Certificação de Sistemas ou Produtos

Este tipo de certificação é realizado devido a exigências legais ou por motivos de mercado.

Na área médica, os sistemas eletrônicos possuem certificação voluntária no Reino Unido e certificação obrigatória nos EUA e na Alemanha. Na área da aviação civil e na área nuclear a certificação é sempre obrigatória em todos os países.

A certificação pode ser aplicada ao Sistema Completo ou a Componentes Individuais.

Embora a fase de certificação de um sistema se realize no final de seu desenvolvimento, o planejamento deste trabalho deve ser realizado no início do ciclo de vida, da mesma forma que as atividades de verificação e validação. Nestes casos, o projetista precisa preparar um Plano de Certificação para ser aprovado pela Autoridade Reguladora.

Este Plano de Certificação deve conter detalhes do sistema proposto, os métodos de desenvolvimento a serem utilizados e a documentação a ser fornecida. Quando uma norma em especial é adotada, o plano deve indicar os métodos/técnicas adotados para atender à norma. Aqueles aspectos em que a norma não é atendida devem ser amplamente justificados.

A submissão do Plano de Certificação será continuada por um debate entre o Projetista e o Órgão Regulador, com o objetivo de resolver quaisquer desacordos ou mal entendimentos.

Se tudo ocorrer bem, no final deste processo o Projetista recebe um “De Acordo” do Órgão Regulador para o desenvolvimento proposto. Caso contrário, mudanças devem ser realizadas no projeto proposto e gerado um novo Plano de Certificação.

Se, durante o projeto, ocorrerem mudanças que podem influenciar o Plano de Certificação, deve haver uma nova avaliação deste plano. Este processo é válido durante todo o Ciclo de Vida do Sistema.

À medida que o projeto se desenvolve, o Projetista deve fornecer, ao Órgão Regulador, toda a documentação adequada, conforme o Plano de Certificação. Em grandes projetos, a documentação é muito vasta, representando um grande investimento em tempo e esforço.

Uma grande porcentagem desta documentação refere-se ao Plano de Segurança que detalha o tratamento das tarefas de segurança através do processo de desenvolvimento. Através de todo o material fornecido, o Órgão Regulador irá emitir uma série de revisões sobre o mesmo. Se o Órgão Regulador aceitar toda essa documentação, é emitido um Certificado ou uma Licença. Em alguns casos, esta Certificação pode ser condicional, através de certas restrições operacionais.

8.4. Perigo e Risco

Perigo: Probabilidade do Sistema estar exposto a um Acidente – **Estado Inseguro**

Risco: Probabilidade de um Acidente ocorrer, considerando as suas conseqüências.

Risco Individual: número de mortes por passageiros quilômetros (ou jornadas, missões)

Exemplo: 4,7 mortes por 100 milhões de passageiros Quilômetros.

Risco Social: Severidade x Frequência

Severidade: número de pessoas afetadas, prejuízo ao meio ambiente e grandes perdas materiais.

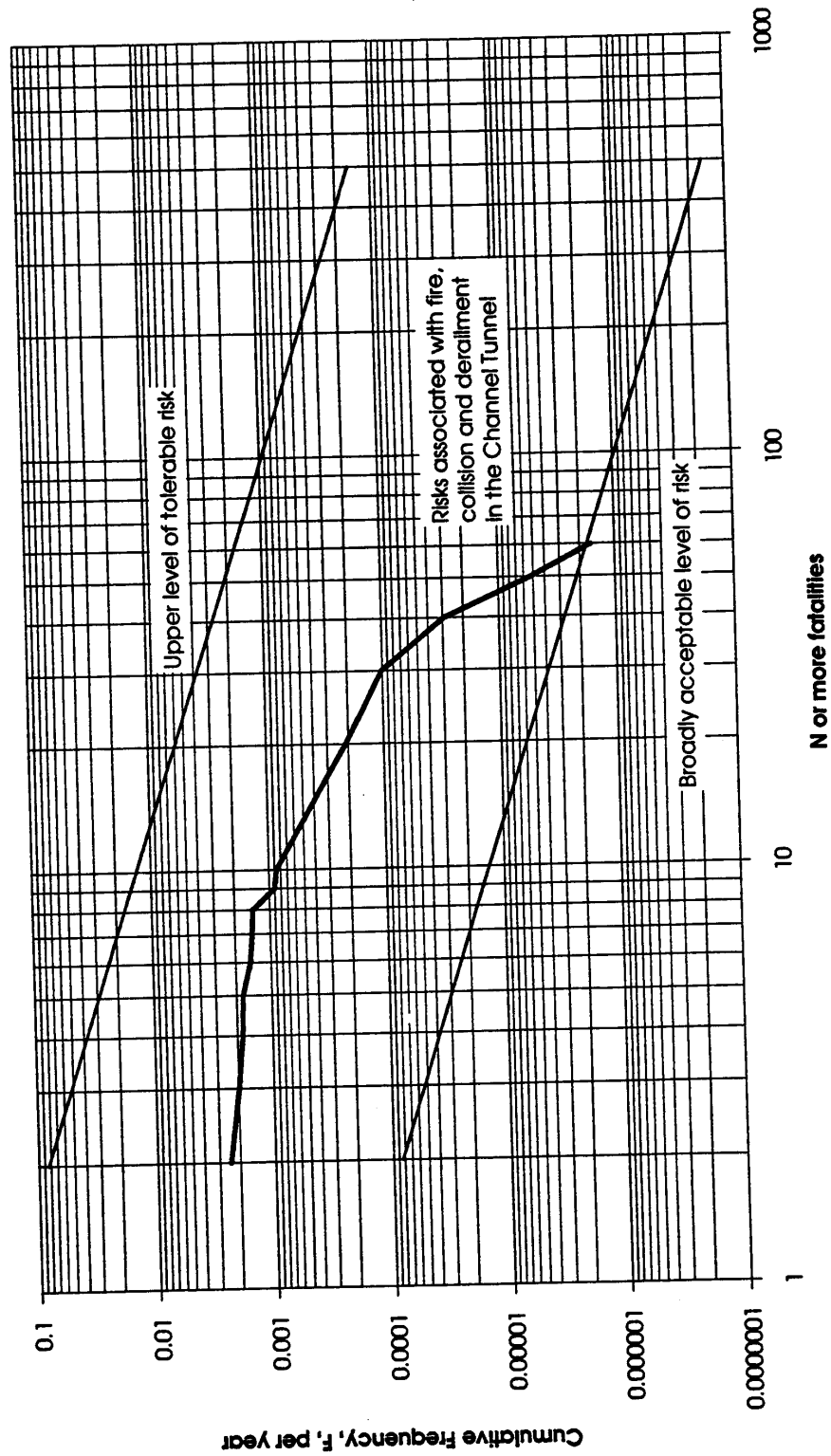
Exemplo: 10 mortes ocorrendo num acidente, no mínimo a cada 90 anos.

Matriz de Nível de Perigo Típica

		Severidade			
Frequência		Catastrófica	Crítica	Marginal	Negligenciável
	Frequente	I-A	II-A	III-A	IV-A
	Moderada	I-B	II-B	III-B	IV-B
	Ocasional	I-C	II-C	III-C	IV-C
	Remota	I-D	II-D	III-D	IV-D
	Improvável	I-E	II-E	III-E	IV-E
	Impossível	I-F	II-F	III-F	IV-F

Análise de Risco: O grande problema é como determinar o **Nível Superior de Risco Tolerável**. Este limite não é estabelecido por cálculos científicos, mas fundamentalmente pela **observação do que a sociedade atual tolera, sendo mais um aspecto político-social do que meramente científico.**

Figure 6.8.1: F-N Curve for the Risks Associated with Fire, Collision and Derailment



8.5. Análise de Perigo

Neste item são apresentados os detalhes que devem nortear um trabalho de Análise de Perigo. O escopo da aplicação de um processo de Análise de Perigo é basicamente constituído por dois objetivos.

O primeiro refere-se ao desenvolvimento de novos sistemas. Neste aspecto a Análise de Perigo procura identificar e avaliar potenciais perigos, além de eliminá-los ou controlá-los.

O segundo escopo refere-se à Análise de Perigo de sistemas existentes. Neste caso o trabalho tem como meta identificar e avaliar perigos, visando quantificar os seus níveis de segurança, formular políticas de segurança, treinar profissionais e aumentar a motivação em se atingir uma operação segura e eficiente.

A Análise de Perigo deve ser um processo contínuo e interativo, se estendendo por todo o ciclo de vida de um sistema.

Ao longo de cada uma das etapas de Análise de Perigo, podem ser utilizados diversos métodos, sendo apresentados parte significativa e representativa deles. Alguns métodos são aplicáveis ao sistema, outros aplicáveis aos módulos de hardware, enquanto outros se adaptam mais aos módulos de software. A melhor aplicação é discutida na apresentação específica de cada método.

O processo de Análise de Perigo pode ser dividido em cinco etapas:

- Análise Preliminar de Perigo
- Análise de Perigo do Sistema
- Análise de Perigo do Subsistema
- Análise de Perigo da Operação e Suporte
- Análise Final de Perigo

A seguir são descritas cada uma destas etapas.

Análise Preliminar de Perigo – Preliminary Hazard Analysis (PHA)

Os objetivos desta etapa são:

- Determinar quais perigos podem existir durante a operação do sistema e sua magnitude relativa;

- Desenvolver recomendações, especificações e critérios para serem seguidos no projeto do sistema. Neste sentido pode ser estabelecido um conjunto de Requisitos Gerais de Segurança – RGS do Sistema;
- Ações iniciais para o controle de um perigo em particular;
- Identificação de Responsabilidades Técnicas e Gerenciais para a Ação e Aceitação de Riscos, como também para Avaliação do Controle sobre os Perigos Indentificados;
- Determinação da magnitude e complexidade dos problemas de segurança.

Análise de Perigo do Sistema – System Hazard Analysis (SHA)

A Análise de Perigo do Sistema pode ter seu início na Revisão Preliminar de Projeto, devendo se estender ao longo de todo o ciclo de vida de um projeto. O principal objetivo desta atividade é recomendar mudanças além de controlar e avaliar o atendimento aos Requisitos Gerais de Segurança – RGS, tanto em operação normal como em operação degradada do sistema, levando evidentemente em consideração a presença de falhas. Nesta etapa, os componentes envolvidos são os diversos subsistemas, incluindo a interface homem-máquina. Como no início deste tipo de análise já se tem uma visão preliminar da arquitetura do sistema, deve-se iniciar o estudo da interação entre os diversos subsistemas constituintes e como suas interações podem afetar a segurança do sistema. Em função desse trabalho podem ser determinados os Requisitos Gerais de Segurança relativos aos subsistemas envolvidos.

Análise de Perigo dos Subsistemas – Subsystem Hazard Analysis (SSHA)

Esta etapa de análise deve ter início a partir da existência de um projeto mais detalhado relativo aos subsistemas. Ela apresenta os mesmos objetivos da análise anterior, só que focada em cada subsistema, de forma mais detalhada. Evidentemente que os outros subsistemas envolvidos, implementados através de outras tecnologias, deverão sofrer um processo de análise com técnicas específicas e desenvolvidas para este fim.

Tendo em mente os sistemas computacionais, objeto deste trabalho de pesquisa, a etapa de análise de perigo dos subsistemas pode ser dividida em: Análise de Perigo do Hardware e Análise de Perigo do Software.

Análise de Perigo do Hardware

A Análise de Perigo do Hardware pode ser subdividida em Análise de Perigo do Hardware Fail-Safe, Análise de Perigo do Hardware Redundante, Determinação dos Meios de Detecção e Recuperação de Falhas Implementados por Hardware e Análise dos Aspectos Construtivos do Hardware.

Um hardware é considerado redundante quando diversas réplicas deste módulo de hardware são utilizadas no sistema visando o atendimento a requisitos não funcionais como, por exemplo, segurança e confiabilidade. Já um hardware é considerado “fail safe” quando, na presença de qualquer falha, simples ou múltipla, sempre é atingido um estado seguro. A exigência de falhas simples ou múltiplas está intimamente ligado ao grau de tolerância de falhas considerado no conceito “fail safe” de um determinado projeto. Pode-se dizer que em sistemas metroviários, o conceito “fail safe” geralmente trabalha com falhas simples. Já na aplicação aeroviária, o conceito “fail safe” lida normalmente com falhas duplas. A diferença básica entre a Análise de Perigo de um Hardware Redundante e de um Hardware Fail-Safe é que, no primeiro, deve haver também uma análise de independência entre os canais redundantes, procurando identificar as falhas de causa comum.

De acordo com a norma [IEC 61508], algumas considerações são feitas com relação ao aspecto de independência dos canais redundantes:

“Os sistemas redundantes podem ser tratados como independentes, ou seja, não apresentam falhas de modo comum, desde que:

- *Sejam funcionalmente diversos para atingir o mesmo resultado;*
- *Sejam baseados em tecnologias diversas;*
- *Não compartilhem partes ou serviços cuja falha possa resultar numa situação perigosa;*
- *Sejam projetados de forma que o modo de falha predominante da parte comum do sistema de suporte (energia) seja na direção segura;*
- *Não devam compartilhar procedimentos operacionais ou de manutenção e testes comuns; e*
- *Estejam fisicamente separados de maneira que falhas externas não afetem os sistemas redundantes e as facilidades externas de diminuição de risco.*

Se todas as exigências anteriores não puderem ser atendidas, então os sistemas relacionados com a segurança não devem ser considerados como independentes do ponto de vista da Alocação da Integridade de Segurança, a menos que uma análise tenha sido realizada, mostrando que a probabilidade da falha dependente (de modo comum) seja suficientemente baixa em comparação com o Requisito de Integridade de Segurança desejado.”

De acordo com a norma [EN 50128] são feitas algumas considerações com relação às Falhas de Causa Comum:

“Algumas falhas podem ser comuns em mais de um componente redundante. Por exemplo, se um sistema computacional é instalado em uma única sala, problemas no ar condicionado podem reduzir os benefícios da redundância. O mesmo pode-se dizer para outros eventos externos como: fogo, inundação, interferência eletromagnética, acidentes aéreos e terremotos. O sistema computacional pode também ser afetado por incidentes relacionados com sua operação e manutenção. É essencial, portanto, que sejam estabelecidos procedimentos adequados de operação e manutenção, além de serem bem documentados. O treinamento da equipe de operação e manutenção é também essencial”

a) Análise de Perigo do Hardware “Fail-Safe”

A Análise de Perigo do Hardware “Fail-Safe” é constituída pelas seguintes atividades:

- **Descrição Funcional e Análise do Módulo em Operação Normal:** esta atividade tem como função primordial descrever e entender todos os aspectos funcionais envolvidos na implementação do módulo em questão. No processo de análise são utilizadas técnicas de simulação com o apoio de ferramentas apropriadas, além de discussões sobre a funcionalidade do módulo entre os profissionais da equipe.
- **Detalhamento dos Requisitos Gerais de Segurança:** esta atividade tem a meta de determinar os requisitos de segurança específicos para determinados blocos de hardware. O objetivo nesta atividade é fornecer subsídios para as próximas atividades, visando a identificação da presença de estados perigosos.

- **Análise Crítica dos Efeitos dos Modos de Falhas – FMECA:** esta atividade tem como finalidade analisar todos os efeitos locais e no sistema, de cada um dos modos de falhas dos diversos componentes existentes neste módulo. No caso de falhas não detectáveis, devem ser avaliadas as combinações com outras falhas possíveis ou entradas impróprias, no sentido de se avaliar qualquer possibilidade de se atingir um estado perigoso. [Beerthuizen 01] São também realizadas simulações na presença de falhas, tanto operacionais como do próprio hardware.
- **Análise de Entradas Impróprias:** nesta atividade são avaliadas as consequências envolvidas quando da ocorrência de entradas impróprias ao módulo, seja através de sinais de entrada errados, não de acordo com a especificação, seja através de variações na alimentação do módulo, considerando os aspectos de diminuição, aumento e oscilação do nível de alimentação.

b) Análise de Perigo do Hardware Redundante

A Análise de Perigo do Hardware Redundante é constituída pelas mesmas atividades da Análise de Perigo do Hardware “Fail-Safe”, acrescentando-se a Análise de Independência dos Canais Redundantes. O grande objetivo desta análise de independência é a determinação de focos de Falhas de Causa Comum. Na realidade este tipo de análise pode ser extremamente rígido conforme o tipo de aplicação envolvido e as recomendações apresentadas anteriormente.

c) Determinação dos Meios de Detecção e Recuperação de Falhas

Implementados por Hardware.

Esses meios de detecção irão influenciar na determinação do Fator de Cobertura de falhas do sistema, e por consequência interferir na avaliação do grau de segurança avaliado. Esse meios de detecção são determinados a partir da análise do hardware redundante e “fail-safe”.

d) Análise dos Aspectos Construtivos do Hardware

Esta atividade procura identificar possíveis focos de falhas que possam levar o sistema a uma condição perigosa.

Análise de Perigo do Software

Um dos grandes desafios refere-se à Análise de Perigo do Software. Esta etapa pode ser subdividida em Descrição Funcional do Software, Detalhamento dos Requisitos Gerais de Segurança, Elaboração da Descrição Funcional das Rotinas a partir do Código Fonte, Elaboração da Descrição das Variáveis Globais, Identificação dos Meios de Detecção e Recuperação de Falhas Implementados por Software, Inspeção Formal do Código Fonte, Reuniões Formais de Análise das Rotinas do Software, e Elaboração de Casos de Testes e Simulações.

a) Descrição Funcional do Software.

Nesta atividade é elaborada uma descrição funcional do software visando identificar a arquitetura utilizada e os atributos funcionais dos módulos envolvidos. Esta atividade tem importância fundamental no sentido de se fornecer uma visão funcional ampla do software.

b) Detalhamento dos Requisitos Gerais de Segurança

Nesta atividade são gerados os requisitos de segurança específicos para o software a partir dos requisitos gerais de segurança, procurando fornecer subsídios para as próximas atividades, visando a identificação da presença de estados perigosos.

c) Elaboração da Descrição Funcional das Rotinas a partir do Código Fonte.

Nesta atividade devem ser incluídas descrições textuais, diagramas de fluxo de dados (contexto e detalhados), diagramas estruturados NS, redes de Petri quando aplicáveis, em especial na representação de eventos concorrentes e sincronizados.

d) Elaboração da Descrição das Variáveis Globais

Nesta atividade é elaborada a descrição funcional das variáveis globais, seu tipo e tamanho, no caso de vetores e matrizes, e sua relação de dependência em relação às rotinas, como ações de Leitura ou Escrita na variável em questão.

e) Identificação dos Meios de Detecção e Recuperação de Falhas Implementados por Software.

Esses meios de detecção irão influenciar na determinação do Fator de Cobertura de falhas do sistema, e por consequência interferir na avaliação do grau de segurança avaliado. Esses meios de detecção são selecionados através de uma classificação das rotinas de software do sistema computacional.

f) Inspeção Formal do Código Fonte

Esta atividade é realizada através da aplicação de uma Lista de Verificações, “Checklist”, desenvolvido especialmente para uma linguagem sendo utilizada. Vale ressaltar que este tipo de análise não exige um conhecimento prévio da funcionalidade do sistema em análise, podendo ser realizado por especialistas em software sem conhecimento da aplicação prática.

A não observância de qualquer dos itens contidos nesta Lista de Verificações pode provocar a realização de processamento não correto, ou mesmo não previsto nas especificações do sistema sob análise. A verificação dos pontos apresentados na Lista de Verificações constitui-se já em um forte indício de que o código verificado tem possibilidade de atender aos requisitos mínimos para utilização em aplicações críticas.

Embora algumas linguagens como, por exemplo, C não tenham sua utilização recomendada, inclusive por normas internacionais, sua aplicação tem sido grande em sistemas críticos de segurança. Um dos motivos que levam a essa utilização é o fato de que há grande difusão e conhecimento no meio técnico dessas linguagens não totalmente recomendadas para aplicações críticas. Especialmente nestes caso é extremamente útil a aplicação da inspeção formal, tendo como referência uma Lista de Verificação. [Lawrence 00]

Outro motivo é que a linguagem “assembly”, que seria a mais recomendada pelo fato de se lidar quase que diretamente com o hardware do processador, tem a grande desvantagem de tornar os programas muito complexos à medida em que as funções implementadas têm a sua complexidade aumentada, exigindo grande habilidade do programador nesta linguagem.

Além disso, o corpo técnico das organizações não tem, em geral, cultura apropriada ao projeto de sistemas críticos de segurança. Dentro desse quadro,

a preocupação com a Inspeção Formal do código fonte é fornecer subsídios para uma utilização, das diversas linguagens, voltada para sistemas críticos.

g) Reuniões Formais de Análise das Rotinas do Software

Nesta atividade, cada uma das rotinas do software é avaliada em uma reunião formal de análise. Esta reunião tem a participação de um Moderador, um Relator, um Apresentador da Rotina e demais profissionais relacionados com a análise do software e do hardware do sistema. Cada uma dessas reuniões deve ter duração máxima de 2 (duas) horas, procurando manter ativo o senso crítico da equipe em relação aos requisitos de segurança, aspecto fundamental durante este tipo de atividade.

Durante essas reuniões, as rotinas são avaliadas e são realizadas simulações, conforme as necessidades envolvidas. As avaliações são realizadas de acordo com critérios a seguir apresentados. Em alguns casos são necessários esclarecimentos junto ao operador do sistema sendo avaliado.

h) Elaboração de Critérios de Avaliação e Simulações

Os critérios básicos no planejamento destas avaliações e simulações são: cobertura lógica, verificação de valores inválidos e de fronteira, verificação de caminhos independentes e “error guessing”. A técnica “error-guessing” é extremamente útil quando a equipe de análise apresenta vasta experiência com o sistema sendo analisado. Evidentemente nesta análise são incluídas as verificações dos “time-outs” críticos envolvidos na operação do sistema, sempre considerando o pior caso. Diversos métodos podem ser utilizados nesta atividade, podendo destacar, como exemplo, os métodos de árvore de falhas, árvore de eventos, redes de Petri, statecharts, análise de completeza e autômatos híbridos.

Análise de Perigo da Operação e Suporte: Operating Support Hazard Analysis - OSHA

Nesta etapa são identificados os perigos e os procedimentos de redução de risco durante a operação e manutenção. Em especial são analisados os perigos criados através da interface homem máquina.

Nesta etapa são avaliados os procedimentos operacionais visando identificar seqüências operacionais que possam levar o sistema a um estado perigoso e,

portanto, devem ser evitadas ou pelo menos minimizada a sua possibilidade de ocorrência.

Análise Final de Perigo

A Análise Final do Perigo do sistema constitui-se na determinação do grau de segurança do sistema sendo analisado. Trata-se de uma integração entre a Avaliação Qualitativa e a Avaliação Quantitativa.

Na Avaliação Qualitativa, são apresentados os problemas encontrados no Software, no Hardware, nos Procedimentos Operacionais e no nível de sistema e subsistema.

Com relação ao software são apresentados os resultados de acordo com a seguinte classificação: problemas potencialmente perigosos, problemas que afetam a disponibilidade do sistema, problemas que afetam a manutenção do sistema, problemas relacionados com aspectos metodológicos além de aspectos estruturais como comentários errados, código não executado.

Com relação ao hardware, são apresentadas as conclusões da análise destacando as falhas ou entradas impróprias que podem levar o sistema a situações perigosas. São incluídas, nesta análise, as falhas no software decorrentes de falhas oriundas do hardware.

No que diz respeito aos Aspectos Operacionais são apresentadas as seqüências operacionais, considerando também as falhas no software e hardware, que podem levar o sistema a alguma condição perigosa.

São também avaliados os aspectos de manutenção corretiva no requisito de qualidade do alarme fornecido. Com relação aos aspectos de manutenção preventiva é avaliada a cobertura de falha dos testes realizados.

Neste momento, no nível de sistema e subsistema, em função de um maior conhecimento detalhado do projeto sendo avaliado, podem ser realizadas as simulações com o intuito de se verificar o atendimento aos Requisitos Gerais de Segurança. Evidentemente que se torna impraticável a modelagem de todo um sistema em função de sua complexidade. Neste sentido, este tipo de avaliação pode ser conduzido sobre partes do sistema, escolhidas em função da criticidade envolvida com a segurança. Neste tipo de avaliação também podem ser utilizados os métodos como Verificações Formais, através de Model Checking, Statecharts, entre outros. Outra possibilidade de expansão

dessa avaliação qualitativa é incluir, em seu método, algum aspecto de avaliação quantitativa, através do levantamento de probabilidades dos eventos envolvidos.

Na Avaliação Quantitativa calcula-se o grau de segurança do sistema representado através do valor do seu MTTUF – “Mean Time to Unsafe Failure”, ou seja, Tempo Médio entre Disfunções Inseguras. Vale ressaltar que o Fator de Cobertura de Falhas, utilizado nesta avaliação, é decorrente da composição dos meios de detecção e recuperação de falhas implementados por software e por hardware, e já determinados nas suas respectivas análises.

8.5.1. Métodos para Realizar Análise de Perigo

É apresentado, neste item, um conjunto de métodos que podem ser utilizados durante o processo de Análise de Perigo. É importante destacar que estes métodos devem ser utilizados em conjunto e serem avaliados por especialistas. Vale dizer também que a melhor cobertura obtida na Análise de Perigo é consequência da combinação dos diversos métodos. Evidentemente existem diversas interseções lógicas entre os métodos. No entanto, a complementariedade entre eles é que garante uma maior cobertura no processo de Análise de Perigo. [Profit 95]

Estes métodos foram escolhidos em função de pesquisas realizadas pelo GAS, além de experiências já adquiridas pelo autor. A relação dos métodos não pretende ser completa, tendo a finalidade de dar uma visão ampla de diferentes métodos em diferentes fases do ciclo de vida do sistema. Foram também escolhidos alguns métodos, que embora ainda não tão detalhados, apresentam um futuro bastante promissor dentro do escopo das pesquisas aqui apresentadas.

8.5.1.1. Lista de Verificação

Esta técnica é derivada, normalmente, de normas, práticas de boa engenharia e da experiência adquirida através de diversos projetos. Esta lista de verificação pode ser uma referência para reflexão sobre o sistema que estiver sendo desenvolvido ou avaliado, devendo ser utilizada ao longo de todo o ciclo de vida do sistema.

Dentro do escopo da Análise de Perigo esta técnica pode ser utilizada na obtenção de maiores detalhes sobre os possíveis perigos, como também, na orientação de decisões de projeto, procurando, desta forma, diminuir os riscos envolvidos.

8.5.1.2. Árvore de Falhas

A técnica de Árvore de Falhas tem como objetivo fundamental estudar, em detalhes, a forma com que os perigos podem ser alcançados, através da combinação lógica de eventos primários. Neste sentido, a Árvore de Falhas não avalia a possibilidade de um novo perigo, mas estuda em detalhes os perigos já identificados.

O topo de uma Árvore de Falha constitui-se num perigo, ou até mesmo, em um acidente. Desta forma, a partir de um determinado Requisito Geral ou Específico de Segurança, adota-se como *Topo* desta árvore o evento da negação do respectivo requisito. Assim sendo, essa técnica utiliza-se de uma filosofia top-down de detalhamento.

Na construção de Árvores de Falhas são utilizados os conectivos lógicos, na maioria das vezes OR e AND para manter a simplicidade do seu entendimento. A Árvore de Falhas é detalhada até se atingir um evento primário, ou básico, ou se chegar a um determinado evento que deverá ser detalhado posteriormente. Após a construção de uma Árvore de Falhas, ela pode sofrer dois processos de avaliação: qualitativo e quantitativo. [Kececioglu 91]

A avaliação qualitativa tem como finalidade representar a ocorrência do perigo através de uma forma lógica equivalente, mostrando a combinação dos eventos básicos que podem causar o evento de topo. Pode-se, através desta avaliação, determinar a criticidade dos eventos envolvidos, podendo inclusive influenciar em decisões de projeto ao longo do desenvolvimento.

A avaliação quantitativa tem como meta avaliar a probabilidade de ocorrência do evento de topo em função das probabilidades de ocorrência dos eventos básicos. Neste sentido é de fundamental importância verificar a independência dos eventos básicos envolvidos.

A título de ilustração é apresentado na figura 8.6 um exemplo simples de Árvore de Falhas com suas respectivas avaliações qualitativa e quantitativa.

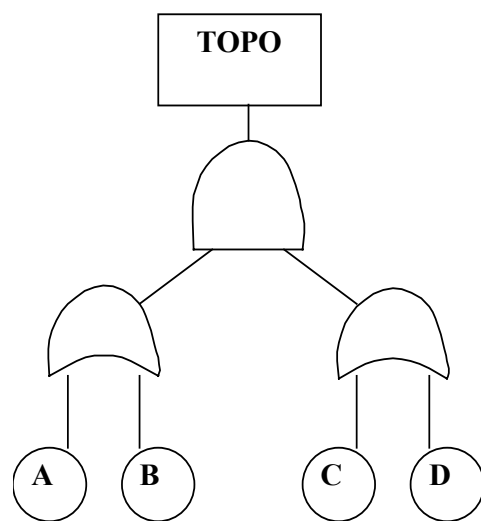


Figura 8.6 – Árvore de Falhas

Avaliação Qualitativa:

$$\text{TOPO} = (A + B) \cdot (C + D) \\ AC + AD + BC + BD$$

Avaliação Quantitativa:

$$P(\text{TOPO}) = P(A) \cdot P(C) + P(A) \cdot P(D) + P(B) \cdot P(C) + P(B) \cdot P(D) \\ - P(A) \cdot P(C) \cdot P(D) - P(B) \cdot P(C) \cdot P(D) - P(A) \cdot P(B) \cdot P(C) \\ - P(A) \cdot P(B) \cdot P(D) + P(A) \cdot P(B) \cdot P(C) \cdot P(D)$$

onde $P(X)$ indica a probabilidade da ocorrência do evento X .

8.5.1.3. Árvore de Eventos

O método de Árvore de Eventos tem como principal meta avaliar quais eventos finais podem acontecer como consequência da combinação da ocorrência de eventos iniciais. Estes eventos iniciais podem se constituir em uma determinada falha de componente do sistema ou ocorrência de eventos externos. Para cada evento inicial devem existir, no mínimo, duas ramificações, uma com seu sucesso e outra com seu fracasso. Nesse sentido, este método tem como filosofia de detalhamento a técnica bottom-up, contrário ao método de Árvore de Falhas e, portanto, tem uma função complementar. Como para sua aplicação são necessárias maiores informações sobre eventos iniciais, este tipo de método é mais adequado de ser aplicado após o projeto detalhado. Na figura 8.7 é apresentada uma ilustração simplificada de aplicação deste método. Foram omitidas, nas probabilidades dos eventos, os termos $(1 - P_x)$, para efeito de simplificação.

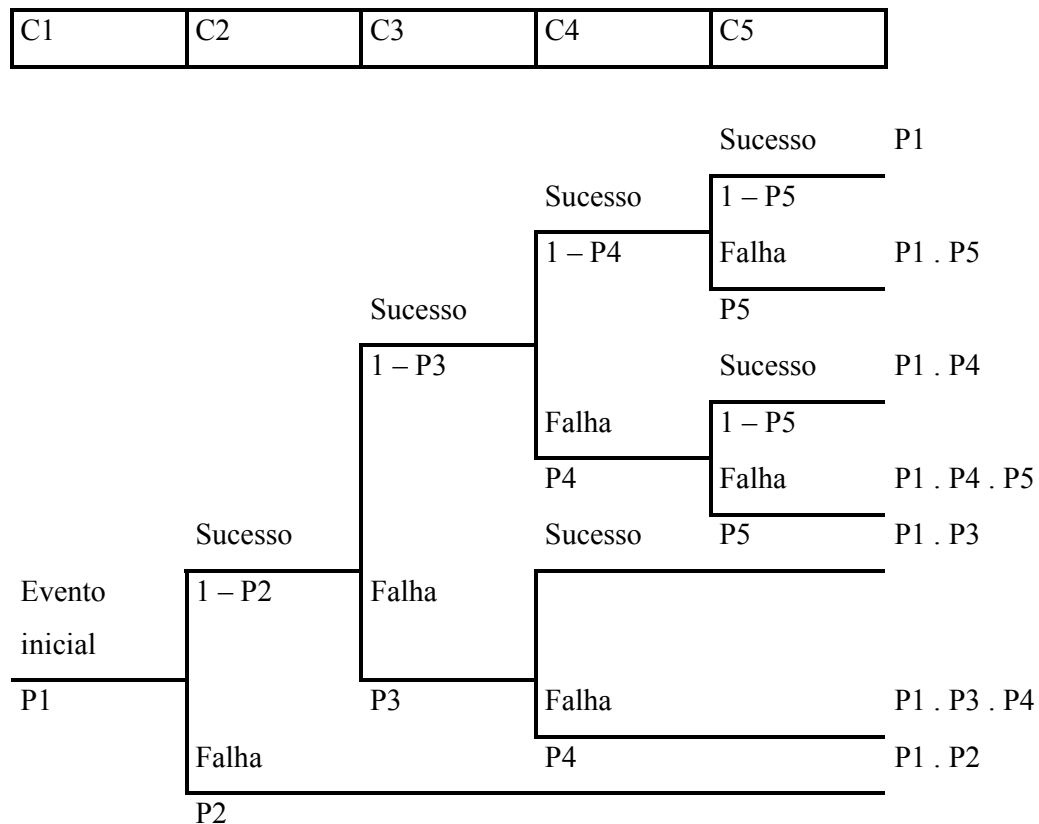


Figura 8.7 – Árvore de Eventos

8.5.1.4. Análise dos Efeitos dos Modos de Falhas - FMEA - Failure Modes and Effects Analysis

O método FMEA tem como objetivo identificar os efeitos dos modos de falha de um determinado componente do sistema. Este método pode ser aplicado sobre os níveis de sistema, subsistema ou módulos, como uma placa de hardware ou uma rotina de software. Quando é aplicado sobre o nível de sistema, os componentes a serem avaliados são os seus subsistemas constituintes. Quando o método é aplicado ao nível de subsistema os componentes são seus grandes módulos, sejam eles hardware ou software. Por outro lado, quando o método é aplicado sobre módulos específicos como, por exemplo, uma placa de hardware, os componentes referem-se aos seus elementos básicos, como um circuito integrado, um capacitor, um resistor, uma memória, entre outros. Esta última forma de aplicação tem sido a mais comumente utilizada. A aplicação do FMEA sobre módulos de software, apesar de possível, tem sido pouco utilizada, devido à grande complexidade de

sua aplicação em relação aos benefícios obtidos na qualidade do projeto. Na realidade, outros métodos são apresentados neste trabalho que melhor se aplicam à avaliação de módulos de software.

A tabela 8.5 apresenta um exemplo estrutural de uma tabela de FMEA que pode ser utilizada. Para cada componente considerado existe uma coluna que representa a sua probabilidade de falha. Em seguida existe outra coluna onde são detalhados todos os modos de falha possíveis de cada um dos componentes. Para cada um destes modos de falha são calculadas as probabilidades de sua ocorrência, seus efeitos locais e no sistema.

Componentes	Probabilidade de falha	Modos de falha	% por modo de falha	Efeitos Locais	Efeitos No Sistema	
					Crítico	Não Crítico
A	1×10^{-3}	Aberto	90	Diminui Tensão		X
		Curto	5	Aumenta Tensão	5×10^{-5}	
		Outros	5	Aumenta Tensão	5×10^{-5}	

Tabela 8.5- Tabela de FMEA

8.5.1.5. Análise Crítica dos Efeitos dos Modos de Falhas - FMECA - Failure Modes, Effects, and Criticality Analysis

O método FMECA constitui-se na adição, ao método FMEA, de uma análise mais detalhada da criticidade da falha. Este estudo mais detalhado pode envolver a determinação dos meios de controle da falha ou até mesmo a necessidade de projeto de um controle adicional visando a diminuição do risco envolvido.

8.5.1.6. Análise de Operação e Perigo - Hazard and Operability Analysis - HAZOP

Para se aplicar o método HAZOP não há a necessidade da identificação prévia dos perigos existentes. Por esta razão, é extremamente importante realizar um HAZOP preliminar para se determinar os Requisitos Gerais de Segurança. No

que diz respeito à Análise de Perigo, o método HAZOP é confundido algumas vezes com o método FMEA. Quando o HAZOP é realizado, há sempre algum elemento de FMEA fazendo parte do método. O HAZOP constitui-se num exercício, realizado por uma equipe, para identificar as causas e as conseqüências dos perigos, enquanto que o FMEA examina apenas as conseqüências das falhas nos componentes e pode ser realizado por apenas um profissional. O ponto inicial da aplicação do HAZOP é a avaliação de possíveis desvios da intenção original do projeto.

O HAZOP identifica o perigo enquanto o FMEA verifica se um determinado componente pode falhar e alcançar aquele perigo. O uso do HAZOP antes do FMEA pode resultar na realização de um FMEA mais focalizado e eficiente.

[Redmill 99]

O aumento do conhecimento das condições de projeto, operacionais e ambientais provoca alguns questionamentos, como por exemplo:

- O projeto resolve adequadamente os perigos identificados previamente?
- Algum novo perigo foi introduzido?
- O conhecimento crescente do sistema permitiu que novos perigos fossem identificados?
- Quais informações adicionais podem ser obtidas sobre as causas potenciais e conseqüências dos perigos identificados anteriormente?

O objetivo do HAZOP é, portanto, rastrear os perigos em potencial sobre o ambiente do sistema e determinar os Requisitos Gerais de Segurança, além de construir estratégias que possam ser utilizadas para implementar os requisitos e evitar a possível existência de um determinado perigo.

É o ambiente em que o sistema está mergulhado que irá determinar os perigos em potencial. Neste sentido, o software, por si só, não se constitui num perigo, mas pode contribuir para a sua ocorrência.

A aplicação de HAZOP em Sistemas Eletrônicos Programáveis é uma inovação recente. Há pouco consenso entre os especialistas em engenharia de software e segurança em como analisar um sistema computacional de aplicação crítica quanto à segurança. O software é freqüentemente pouco considerado durante a análise dos sistemas. No entanto, tal aspecto é inaceitável quando o software desempenha funções de segurança. Neste

sentido, a aplicação do HAZOP sobre especificações de software é bastante eficaz do ponto de vista de segurança, procurando identificar caminhos perigosos não facilmente localizáveis. [Lawrence 97] Desta forma, apontam-se duas questões:

- Se o software operar de acordo com sua especificação, qual o efeito sobre os perigos do sistema?
- Se o software operar em desacordo com sua especificação, qual o efeito sobre os perigos do sistema?

Neste sentido algumas questões devem ser consideradas quando o software apresenta um papel primordial na segurança do sistema:

- O software responde inadvertidamente a um estímulo;
- O software falha em responder quando requerido;
- O software responde fora da sequência especificada; e
- O software responde de forma não planejada quando combinado com outras ações.

Dentro desta filosofia, há basicamente pelo menos quatro impactos potenciais do software com relação aos perigos já identificados:

- O software pode comprometer a segurança do sistema: falha na operação do software tem o potencial de criar condições de perigo que devem ser removidas ou amenizadas através de outros sistemas;
- O software pode ser responsável em prevenir que perigos evoluam para acidentes;
- O software pode ser utilizado para transpor o sistema de um estado perigoso para um estado não perigoso; e
- O software pode ser utilizado para amenizar as consequências de um acidente.

Outros aspectos importantes relacionados com o software, como por exemplo, ferramentas, linguagens, técnicas de codificação devem ser examinadas de forma a avaliar o seu potencial de introduzir falhas de causa comum a todos os módulos redundantes.

Desta forma, existem no HAZOP as *Guide Words* cujo objetivo é focalizar o estudo e maximizar as chances de identificar algum perigo.

Na realidade cada *Guide Word* pode ter mais de uma interpretação no contexto da representação do projeto. O objetivo destas palavras é focalizar melhor a análise permitindo, desta forma, maximizar as chances de identificação dos perigos. São aplicadas sobre os sinais de dados e controle.

As *Guide Words* consideradas são [Redmill 97]:

NO : nenhum dado ou controle é fornecido;

MORE: Os dados são fornecidos a uma taxa superior do que a esperada, ou mais dados são fornecidos;

LESS: Os dados são fornecidos a uma taxa menor do que a esperada, ou menos dados são fornecidos;

AS WELL AS: Os objetivos foram alcançados, mas com resultados adicionais;

PART OF: Os dados ou controle estão incompletos;

REVERSE: Ocorre fluxo reverso;

OTHER THAN: Os dados ou controle estão completos, mas incorretos;

EARLY: O sinal chega mais cedo que o esperado;

LATE: O sinal chega mais tarde que o esperado;

BEFORE: O sinal chega antes, dentro de uma seqüência de eventos esperados; e

AFTER: O sinal chega depois, dentro de uma seqüência de eventos esperados.

8.5.1.7. Avaliação de Completeza de Especificações

Este método foi desenvolvido na tese de doutorado do autor, sendo apresentado, nesta seção, um breve resumo do trabalho de pesquisa, com o intuito de destacar este tipo de análise como um método importante dentro de um processo de Análise de Perigo.

Para uma especificação de requisitos, seja de software, hardware ou sistema, ser considerada robusta, são necessários certos atributos fundamentais: não ser ambígua, ser completa, verificável, consistente, modificável, rastreável e utilizável durante as fases de operação e manutenção.[IEEE - Std 830-1993]

A segurança é definida como uma probabilidade de o sistema não atingir um estado perigoso, que pode afetar vidas humanas e bens materiais. De acordo com alguns pesquisadores dessa área, a análise probabilística da segurança, ou

seja, a análise quantitativa, não é muito adequada para assegurar ou avaliar a segurança de um sistema. [LEVESON 83], [HAMLET 92], [BUTLER 93] Esse grupo adota a postura de garantir ou melhorar a segurança através de métodos específicos de projeto.

Em função também da experiência adquirida por diversos engenheiros de segurança de sistemas, concluiu-se que a grande causa de problemas de segurança em sistemas críticos é a existência de uma especificação inadequada de requisitos, ou seja, falta de robustez, além dos erros na aplicação de técnicas de engenharia.[LEVESON 86], [RUSHBY 94], [SHELDON 92]

Pela experiência profissional do autor, também adquirida através de diversos trabalhos de análise de risco de sistemas críticos, especificamente metro-ferroviários, é possível concluir que grande parte das possíveis falhas inseguras ocorre devido à falta de especificação com relação ao universo da falha em questão, ou a possíveis mudanças no ambiente operacional.

Em razão dessas observações, pode-se dizer que uma melhoria na análise da completeza de uma especificação deverá aumentar muito a qualidade das mesmas em diversos aspectos, especialmente no que diz respeito à segurança. Dentro desse enfoque, desenvolveu-se um método de avaliação da completeza de uma especificação de um sistema crítico em relação ao ambiente de aplicação.

Os critérios de completeza são classificados em função de suas características sobre o modelo de transição de estados e devem ser aplicados conforme o nível de detalhe em que se analisa a especificação, ou seja, nos níveis de sistema, subsistema, módulo ou componente.

Tendo como referência básica o trabalho de [JAFFE 91], e adaptando-se aos aspectos propostos neste trabalho, criaram-se seis categorias básicas de Critérios para verificar a completeza:

- Variáveis de Entrada/Saída;
- Estados;
- Predicados de Entrada;
- Predicados de Saída;

- Relação Entrada/Saída; e
- Transições.

a) Critérios para as Variáveis de Entrada/Saída

- **Critério 1.1:** As variáveis de entrada/saída devem ser utilizadas, caso contrário não haveria a necessidade de existirem. Uma variável de entrada qualquer deve ser utilizada em algum predicado de entrada de uma transição. Da mesma forma, uma variável de saída deve ser utilizada em algum predicado de saída de uma transição. Se isso não acontecer, ou a variável de entrada/saída não deve fazer parte do sistema, ou então existe uma omissão na especificação.
- **Critério 1.2:** Seja **v** uma variável de entrada ou saída. A **validade(v)** reflete a sua validade ou consistência. Esse atributo pode ser usado, por exemplo, para especificar valores aceitáveis, paridade, precisão das variáveis. Quando do não atendimento a essas verificações, uma resposta adequada deve ser declarada, ou, pelo menos, ser armazenada num arquivo histórico, para posterior análise “off-line”, conforme for mais conveniente.

b) Critérios para os Estados

- **Critério 2.1:** O Sistema deve iniciar e finalizar num estado seguro. O sistema de segurança deve estar apto a funcionar na iniciação e na finalização do sistema, incluindo também, nesse aspecto, a reiniciação após um período fora de operação.

Com relação às condições de iniciação, existem duas situações básicas: iniciação após um processo completo de desligamento e após um desligamento temporário, as quais caracterizam o que se denomina de iniciação a “frio” e a “quente”, respectivamente. Em ambos os casos, muitos acidentes acontecem em função de um processamento não adequado.

- **Critério 2.2:** O comportamento do sistema com relação às entradas antes da iniciação, após o desligamento e durante o desligamento, deve ser especificado, seja detalhando a forma de utilização dessas entradas,

ou podendo ignorá-las de maneira a não comprometer a segurança do sistema.

A utilização dos modos de operação auxilia na simplificação da descrição da operação do sistema, permitindo uma visão de alto nível do fluxo de controle do mesmo. Em sistemas de segurança crítica, os modos de operação estão associados a estados operacionais que são subdivididos em estados seguros e estados perigosos. Os estados perigosos podem ainda ser classificados em estados de alto risco e estados de baixo risco, conforme a probabilidade de serem levados a estados perigosos. Dessa forma, as atitudes a serem tomadas quando se atinge um estado perigoso podem diferir dependendo do modo de operação em que se encontra o sistema.

- **Critério 2.3:** Os caminhos a partir de um estado seguro devem ser especificados. Além disso, deve ser minimizado o tempo de permanência num estado seguro de funções reduzidas, ou seja, um estado em que a degradação é caracterizada pela não habilitação de algumas funções. Deve também ser minimizado o tempo de permanência num estado perigoso.
- **Critério 2.4:** Falhas no sistema de segurança devem provocar a transição do sistema para um estado seguro degradado, com o desligamento das funções que envolvem risco.
- **Critério 2.5:** A falha no sistema deve provocar a transição do mesmo para um estado seguro.

c) Critérios para os Predicados de Entrada

Para que o sistema seja robusto, é necessário também que haja um caminho previsto para todas as entradas em cada estado. Outro aspecto importante, já comentado anteriormente, é a temporização na especificação de sistema críticos, em função do instante da ação. Dessa forma, os critérios para os predicados de entrada estarão organizados na seguinte classificação: aspectos Lógicos, de Temporização e de Capacidade, que são explicados nos seus respectivos itens.

c.1) Aspectos Lógicos

Os aspectos lógicos relacionam-se com a consistência lógica desses predicados dentro do modelo de transição de estados adotado. Assim, enquadram-se dentro dessa categoria os seguintes critérios:

- **Critério 3.1:** Cada estado deve ter uma transição definida para cada entrada possível.
- **Critério 3.2:** O “OR” lógico dos predicados de entrada nas transições de saída de qualquer estado deve constituir uma tautologia; caso contrário, constata-se a presença de alguma situação não prevista no sistema.
- **Critério 3.3:** Para haver um comportamento determinístico do sistema, é necessário que, num determinado instante, seja verdadeiro apenas um predicado de entrada, correspondente a uma determinada transição a partir do estado corrente do sistema.

c.2) Aspectos de Temporização

Os aspectos de temporização englobam as condições relacionadas com os intervalos válidos para as entradas, bem como para a ausência desses sinais. Assim, dentro dessa categoria, enquadram-se os seguintes critérios:

- **Critério 3.4:** Em cada estado o sistema deve ter um comportamento bem definido, ou seja, deve executar alguma transição no caso de não haver entradas por um período de tempo denominado “Time Out”. Esse aspecto pode ser também utilizado na reiniciação do sistema quando do não recebimento de determinadas entradas.
- **Critério 3.5:** Todos os eventos de entrada devem ser totalmente limitados no tempo, através da especificação de um tempo mínimo e de um tempo máximo, para sua coerência.

Deve ser especificada uma saída para a ausência do sinal por um período de tempo em relação a algum evento observável. Assim sendo, deve ser estabelecido um tempo específico a partir do qual se inicia a temporização correspondente à ausência de entradas.

- **Critério 3.6:** Para um estado qualquer, deve existir uma transição de saída, cujo predicado de entrada envolva a não existência de uma

determinada entrada durante um intervalo específico em relação a algum evento observável.

c.3) Aspectos de Capacidade

Embora as entradas fornecidas ao sistema pelo operador ou por um outro sistema dificilmente causem alguma sobrecarga no sistema avaliado, outros problemas, devidos a mal funcionamento, podem causar esse excesso, ultrapassando o limite de capacidade. A robustez, nesse caso, requer que se especifique a forma de se manipularem entradas excessivas e determina um limite de capacidade para tais entradas, como meio de se detectarem possíveis problemas externos e internos ao sistema em questão. Isso implica em se determinar o número máximo de entradas dentro de um certo intervalo de tempo. Para os casos em que a capacidade é excedida, deve haver alguma especificação da maneira como o sistema poderá falhar.

Software e hardware requerem que uma declaração seja feita sobre o número máximo de entradas N por um período de tempo de duração T .

A capacidade de entrada C_e depende do processo controlado e dos sensores que enviam informações sobre o processo. Num sistema de controle, essa capacidade de entrada subdivide seus estados em: estados normais e estados de sobrecarga.

Dessa forma, enquadram-se dentro dessa categoria os seguintes critérios:

- **Critério 3.7:** Uma variável de entrada no sistema requer uma declaração de capacidade.
- **Critério 3.8:** Declarações de capacidade mínima e máxima devem ser especificadas para cada variável de entrada, cuja taxa de chegada não seja limitada por outro tipo de evento.
- **Critério 3.9:** Deve ser requerida uma verificação das taxas mínima e máxima de chegada de eventos para cada caminho de comunicação fisicamente distinto. O sistema deve ser capaz de supervisionar o ambiente de comunicação.

- **Critério 3.10:** As respostas às entradas que excedem a capacidade do sistema devem ser especificadas. Essas respostas devem se enquadrar em alguma das classes detalhadas a seguir:
 - Requisitos para gerar mensagens de advertência, ou seja, avisos aos operadores do sistema para adotarem atitudes operacionais que amenizem as consequências da sobrecarga.
 - Requisitos para gerar sinais controle para sistemas externos visando a diminuição de capacidade, “slowdown”. Nesse caso, a sobrecarga pode ser limitada através de um controle indireto dos sistemas externos. É fundamental a interface de comunicação entre os sistemas envolvidos.
 - Requisitos para bloquear os canais em sobrecarga, “lockout”, havendo, dessa forma, uma degradação do sistema quanto ao tratamento das informações bloqueadas.
 - Requisitos para reduzir a precisão, o tempo de resposta, ou outra característica que permita ao sistema processar uma capacidade maior com relação às entradas. Nesse aspecto, o sistema passa para um estado de degradação, diminuindo a qualidade do processamento das informações de entrada.
 - Requisitos para reduzir a funcionalidade do sistema ou, em casos extremos, solicitar o desligamento do controle ou, até mesmo, do processo, quando necessário.
- **Critério 3.11:** Se a degradação ou um processo de reconfiguração é utilizado como meio de atender a uma capacidade excessiva, deve ser especificado um atraso referente à histerese para que o sistema possa retornar ao processamento com carga normal. Esse atraso devido à histerese tem, por objetivo, evitar o chaveamento indevido entre carga normal e sobrecarga, também denominado “efeito ping-pong”.

d) Critérios para os Predicados de Saída

Para que uma especificação seja robusta em relação aos predicados de saída, deve-se verificar os limites de tempo inferior e superior das saídas. Esses requisitos estão classificados de acordo com a seguinte estrutura: capacidade do ambiente, envelhecimento da informação e latência.

d.1) Capacidade do Ambiente

Define-se a capacidade do ambiente como sendo a taxa máxima para a qual os atuadores podem aceitar e reagir aos dados produzidos pelo controlador. Esse valor-limite é afetado pelos seguintes elementos:

- limitação de capacidade dos próprios atuadores;
- limitações no comportamento do processo controlado; e
- considerações de segurança.

Dentro dessa filosofia, enquadram-se os seguintes critérios:

- **Critério 4.1:** A capacidade do ambiente deve corresponder à capacidade máxima de saída do sistema. Se a capacidade máxima de saída do sistema for excedida, é necessária a realização de alguma ação especial, visando normalizar a taxa de saída.

d.2) Envelhecimento da Informação

Outro aspecto importante envolve a obsolescência das informações. As decisões de controle devem ser baseadas em dados atualizados do estado do sistema.

- **Critério 4.2:** Todas as entradas utilizadas na especificação de predicados de saída devem ser adequadamente limitadas no tempo, para garantir, dessa forma, o seu não envelhecimento.
- **Critério 4.3:** A seqüência de ações perigosas deve ser limitada no tempo, após o qual o sistema deve requerer o cancelamento automático dessas ações e informar ao operador.

d.3) Latência da Informação

Dado que um sistema de controle não é arbitrariamente rápido, há um intervalo de tempo durante o qual o recebimento de uma nova informação não pode mudar a variável de controle de saída, mesmo que chegue antes dessa geração. Esse intervalo de tempo é influenciado pelo hardware e pelo software do sistema de controle, podendo ser bastante pequeno, mas nunca reduzido a zero. A escolha do sistema operacional a ser utilizado, a lógica de interrupção, a prioridade no escalonamento de tarefas e os diversos parâmetros de projeto

serão influenciados pelo valor máximo permitido para esse intervalo de latência.

- **Critério 4.4:** Um fator de latência deve ser incluído na especificação quando uma saída não restritiva é disparada pela ausência ou pela chegada de uma entrada específica, para a qual o limite superior desse intervalo de tempo não é um evento observável simples, em função de prováveis interferências ou da própria dinamicidade do sistema de controle.
- **Critério 4.5:** Ações contingentes podem ser necessárias na especificação, para tratar os eventos de entrada que ocorrem dentro do período de latência.
- **Critério 4.6:** Um fator de latência deve ser especificado para mudanças ocorridas na interface homem máquina, quando usadas para decisões críticas. Da mesma forma, as ações contingentes devem ser especificadas quando da mudança das informações na interface homem máquina durante o período de latência. Esse critério está relacionado com o tempo de arrependimento do operador.
- **Critério 4.7:** Um período de histerese deve ser especificado, correspondente ao atraso da ação do operador na interpretação das informações a ele apresentadas. Devem ser especificadas as ações caso as informações apresentadas, ao operador, mudem durante esse período de histerese.

e) Critérios para a Relação Entrada/Saída

Existem requisitos que estão relacionados com Entrada/Saída, podendo ser classificados de acordo com os seguintes aspectos:

- Capacidade de resposta; e
- Espontaneidade.

A Capacidade de resposta e a Espontaneidade lidam com o comportamento do processo e, como ele, reage às saídas produzidas pelo sistema de controle. Esses parâmetros são supervisionados pela realimentação.

- **Critério 5.1:** Deve haver variáveis supervisionadas que detectem o efeito das variáveis de controle. As informações sobre a característica do processo devem ser utilizadas como características preditivas do comportamento esperado do sistema.
- **Critério 5.2:** Para cada variável de controle que possui uma variável de supervisão de comportamento esperado, devem ser avaliadas as condições para resposta muito demorada ou muito rápida.
- **Critério 5.3:** O recebimento espontâneo de uma entrada apenas esperada em resposta a alguma saída anterior do sistema, deve ser detectada e respondida como uma situação anormal, realizando alguma ação contingente.

f) Critérios para as Transições

Em particular, os requisitos relacionados com as transições servem para garantir que certos estados sejam alcançáveis.

As características básicas consideradas são Alcance Básico, Comportamento Recorrente, Reversibilidade e Alcance de Estados Seguros.

f.1) Alcance Básico

- **Critério 6.1:** Todos os estados devem ser alcançáveis a partir de um estado inicial. Caso um estado não seja alcançável, há duas possibilidades:
 - o estado não tem função e pode ser eliminado da especificação; e
 - o estado deve ser alcançável e a especificação precisa ser modificada.

f.2) Comportamento Recorrente

- **Critério 6.2:** O comportamento recorrente desejável deve ser parte de um ciclo, ou seja, deve ser possível atingir um estado, de forma coerente, a partir dele mesmo e através de um caminho não vazio, passando por outros estados. Esse critério passa a ter um papel fundamental se o estado em questão for de importância para a segurança do sistema.

f.3) Reversibilidade

A reversibilidade está relacionada com a capacidade de os comandos sobre os atuadores serem cancelados ou revertidos por algum outro comando ou combinação.

- **Critério 6.3:** A reversibilidade de um procedimento de operação no sistema, por outro procedimento, requer a existência de caminhos possíveis entre os estados atingidos e revertidos.

f.4) Alcance dos Estados Seguros

Nesse item, são avaliados os critérios específicos relacionados com os estados seguros e as ações esperadas caso os mesmos, por algum motivo, não possam ser alcançados.

- **Critério 6.4:** Não deve haver caminho coerente para um estado perigoso.

Dado que o sistema atingiu um estado perigoso, seja devido a falhas de componentes ou do sistema computacional, erro humano ou distúrbio externo,

entre outros, o sistema de controle deverá conduzir o processo para um estado seguro. A decisão da ação adequada a ser tomada deverá depender das condições ambientais em que o processo se localiza no momento.

- **Critério 6.5:** Todo caminho a partir de um estado perigoso deve conduzir a um estado seguro.

Pode não ser possível construir um sistema intrinsecamente seguro, ou seja, “fail-safe”. Nesse caso, o sistema deverá transformar o estado perigoso em um estado de mínimo risco aceitável, aspecto relacionado com o MTTUF do sistema.

- **Critério 6.6:** Se um estado seguro não puder ser alcançado a partir de um estado perigoso, todos os caminhos a partir desse estado perigoso devem levar o sistema a estados de mínimo risco aceitável.

8.5.1.8. Métodos Semi Formais

De acordo com a norma IEC 61508, os Métodos Semi Formais são aqueles que fornecem meios de se desenvolver uma descrição do sistema em algum nível de seu desenvolvimento. A descrição pode, em alguns casos, ser analisada automaticamente ou com animação, mostrando vários aspectos do comportamento do sistema. O aspecto de animação pode fornecer informações adicionais, mostrando que o sistema atende os requisitos especificados. Os métodos apresentados aqui são Diagrama de Transição de Estados, Redes de Petri e Statecharts.

a) Diagrama de Transição de Estados

O Diagrama de Transição de Estados é especialmente apropriado para sistemas sequenciais. A figura 8.8 apresenta um exemplo de formalização deste método, onde $Ci\uparrow$ representa o evento condição da ocorrência da transição e $Si\uparrow$ representa o evento saída, gerado na ocorrência da transição.

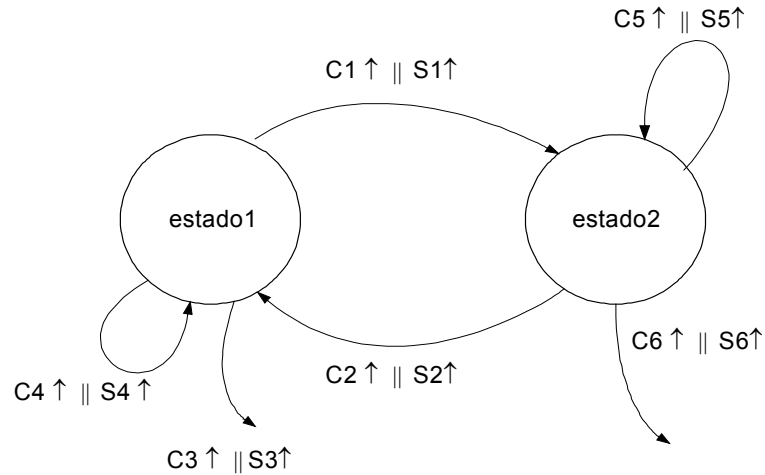


Figura 8.8. – Diagrama de Transição de Estados

b) Redes de Petri

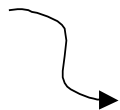
As Redes de Petri são especialmente apropriadas para representar sistemas com alto grau de concorrência. Os elementos de uma Rede de Petri são:



NÓ



TRANSIÇÃO



CAMINHOS.ARCOS



MARCAS - TOKEN

As Regras básicas de sua formação são:

- Uma transição é permitida quando todos os estados que possuem arcos orientados para a transição, contêm marcas;
- Quando uma transição é permitida, ela é disparada, retirando-se uma marca de cada um dos estados de entrada da transição e colocando-se uma marca em cada um dos estados para os quais existe um arco orientado que sai da transição;

- Um estado qualquer nunca pode estar ligado por outro arco orientado a outro estado; e
- Uma transição qualquer nunca pode estar ligada por um arco orientado a outra transição.

Na figura 8.9 é apresentado um modelo de um cruzamento de uma ferrovia com uma rodovia, através de Redes de Petri. Através de sua execução, pode ser gerado o Grafo de Alcançabilidade, conforme apresentado na figura 8.10. Através deste Grafo de Alcançabilidade podem ser identificados os estados perigosos (X X P3 X X P11 X X) e avaliados métodos de controle destes perigos.

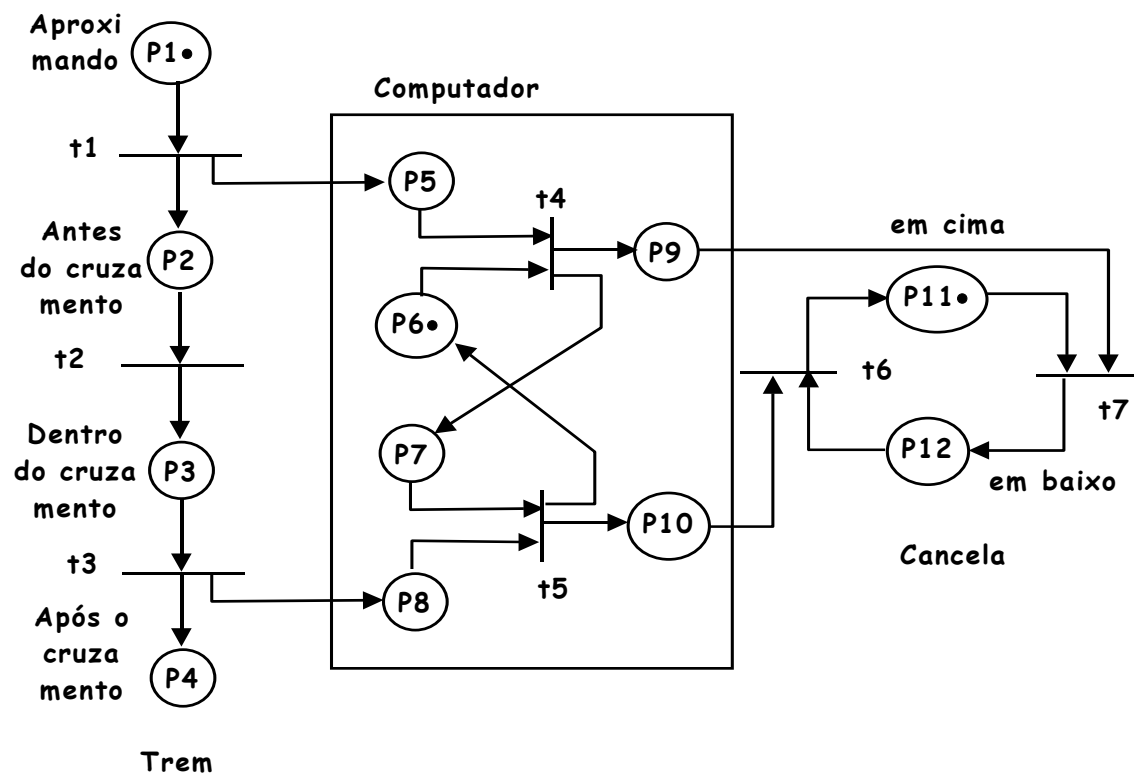


Figura 8.9 – Modelo de Cruzamento através de Rede de Petri

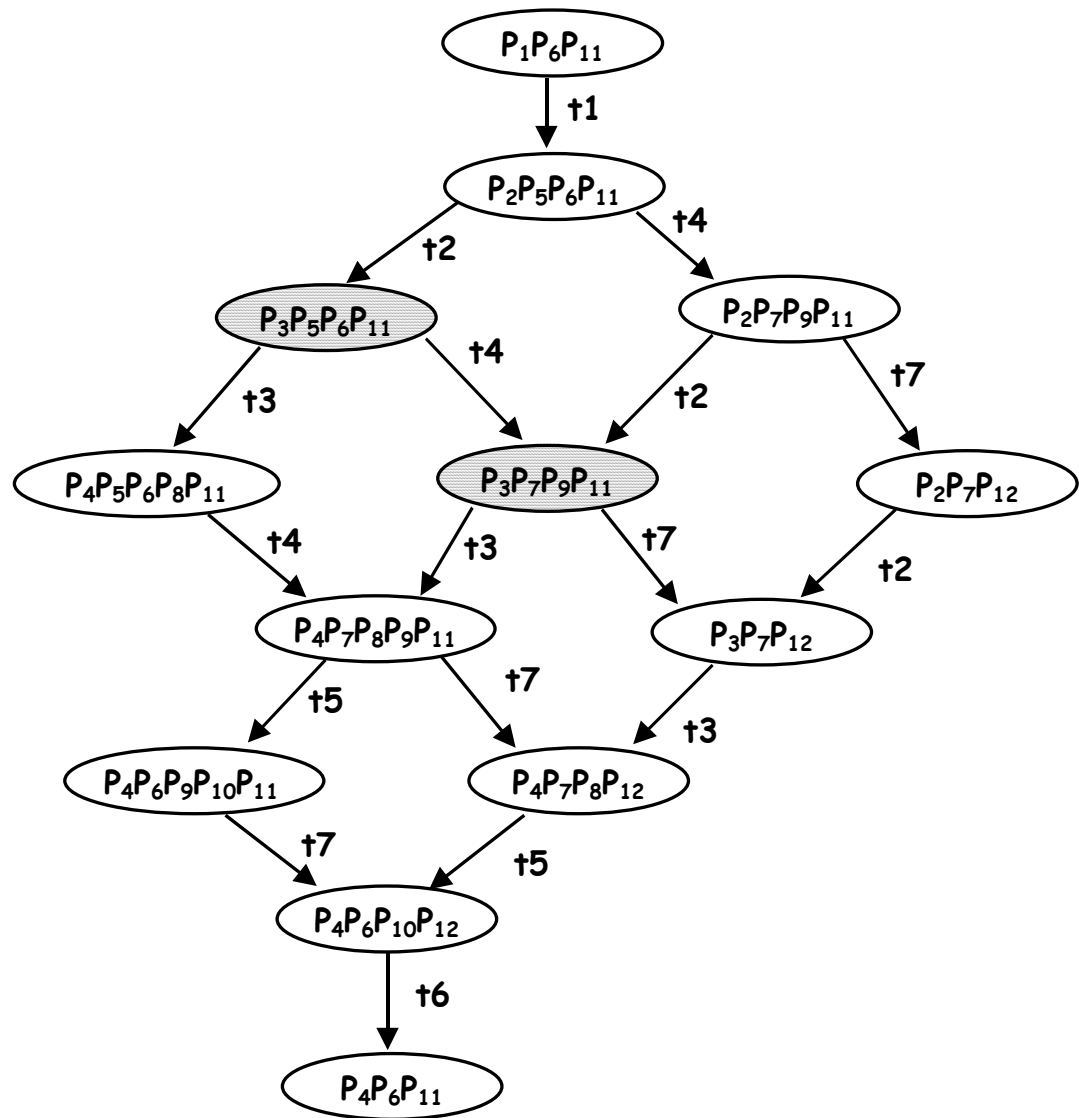


Figura 8.10 – Grafo de Alcançabilidade do Cruzamento através de Rede de Petri

c) Statecharts

Este método engloba as vantagens do Diagrama de Transição de Estados com a Rede de Petri, permitindo, desta forma, representar o comportamento seqüencial e concorrente. Neste formalismo as bolhas AND correspondem às linhas tracejadas enquanto que as bolhas XOR correspondem às linhas cheias. Se uma determinada bolha estiver ativa (for o estado corrente) e tiver sido decomposta em outras bolhas XOR, então apenas uma dessas bolhas XOR(bolhas descendentes) será ativada.

Se por outro lado, ela tiver sido decomposta em bolhas AND, todas essas bolhas descendentes (bolha AND) serão ativadas quando seu antecessor for ativado. Desta maneira, as bolhas AND são um meio de descrever o comportamento paralelo de forma explícita.

Uma transição acontece na dependência de condições restritivas e na ocorrência de eventos específicos, sendo representada por meio de arcos direcionados. Os eventos e condições restritivas que disparam uma transição, bem como as ações específicas a serem realizadas quando de uma ocorrência, são representados como atributos do arco correspondente. Ações associadas às transições de estado podem ser utilizadas para gerar novos eventos, os quais podem ter efeito sobre outras transições dos componentes concorrentes. A comunicação entre os componentes concorrentes pode ser estabelecida pela geração de eventos internos, ou seja, gerados diretamente pelos processos representados pelas Bolhas e, portanto, internamente ao Statechart.

Além das ações, um atributo de história pode ser associado às bolhas. Atributos de história levam ao processo de ativação de bolhas. Sempre que uma bolha com o atributo de história for ativada, verifica-se, em seu Statechart de decomposição, quais foram as bolhas que estavam ativas na última vez em que se realizou a simulação de tal detalhamento, com a finalidade de que se ativem tais bolhas. Se não houver uma história nos descendentes ou se for a primeira vez em que a bolha com o atributo de história foi ativada, então seus descendentes “default”, se existirem, é que serão ativados.

Como exemplo de aplicação é apresentado, na figura 8.11, a modelagem do sistema de cancela, anteriormente formalizado através de Rede de Petri.

SISTEMA DE CONTROLE DE CANCELA

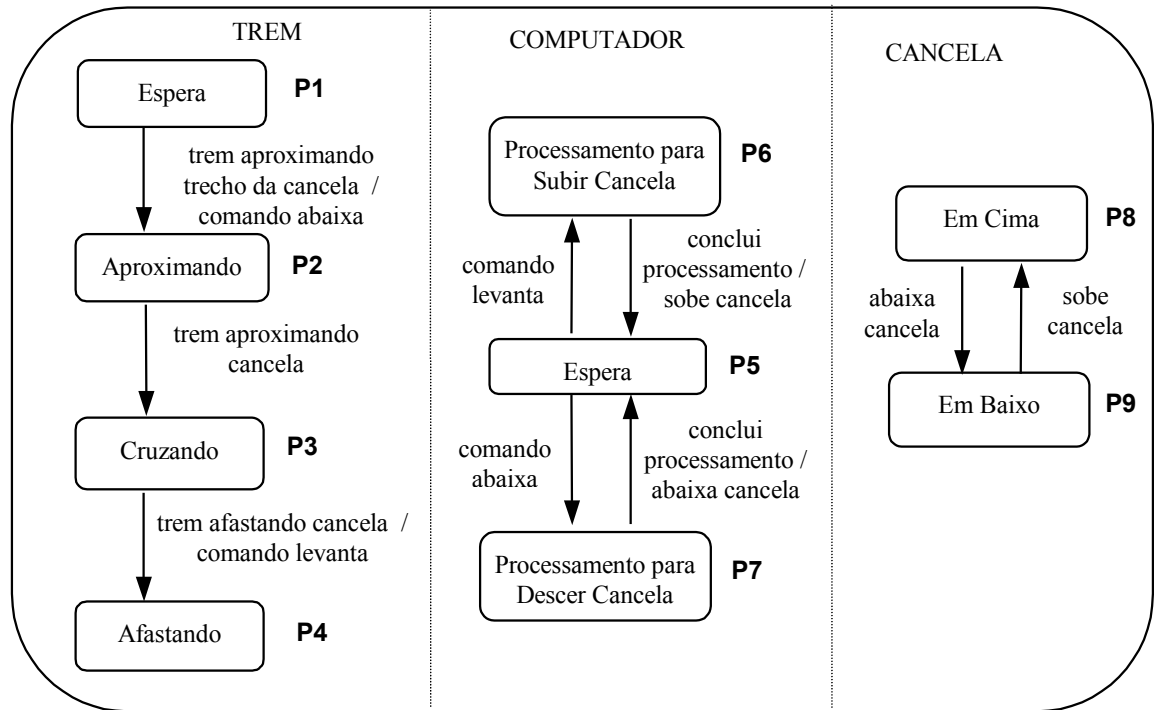


Figura 8.11 – Modelo de Cruzamento com Statechart

Na figura 8.12 é apresentado o Grafo de Alcançabilidade correspondente a esse novo modelo, onde o estado (P3 P7 P8) corresponde a um estado perigoso.

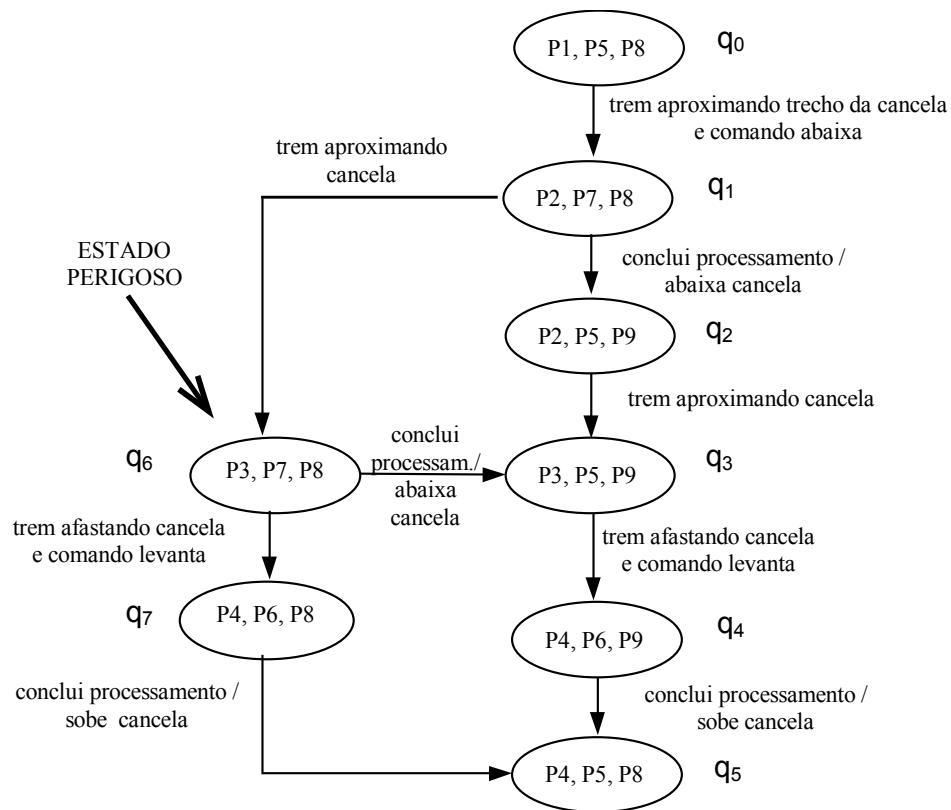


Figura 8.12 – Grafo de Alcançabilidade do Cruzamento Modelado com Statechart

8.5.1.9. Métodos Formais

De acordo coma norma [IEC 61508] um Método Formal é aquele que fornece meios de desenvolver uma descrição do sistema em algum nível de seu desenvolvimento. Esta descrição resultante apresenta uma forma matemática e pode ser submetida a análise matemática com o intuito de detectar várias classes de inconsistências ou incorreções.

Com o aumento da complexidade do software e do hardware, aumenta-se a probabilidade da presença de erros ainda não detectados. Especial atenção deve ser dada quando esses erros fazem parte de um sistema crítico quanto a segurança.

Um dos grandes objetivos da engenharia de software é permitir aos projetistas a construção de sistemas com maior segurança e confiabilidade, apesar da grande complexidade. Um dos meios de se atingir esse objetivo são os Métodos Formais.[Broomfield 97]

O Uso dos Métodos Formais não garante a correção *a priori*. Entretanto, eles podem aumentar bastante o entendimento do sistema, revelando inconsistências, ambigüidades ou incompletezas, que poderiam continuar não detectáveis.

As duas grandes abordagens dentro deste campo são: ESPECIFICAÇÃO E VERIFICAÇÃO. [Clarke]

A Especificação corresponde ao processo de descrever o sistema e suas propriedades. A especificação formal utiliza-se de linguagem com sintática e semântica matematicamente definidas. As características formalizadas podem incluir Comportamento Funcional, Temporal, Características de Desempenho ou Estrutura Interna.

Uma linha atual de trabalho corresponde à integração de diferentes linguagens de especificação, cada uma permitindo a manipulação de aspectos diferentes do sistema, enquanto outras manipulam aspectos de desempenho, limitações de tempo real, políticas de segurança e projeto de arquitetura.

Como exemplos de formalismos de especificação podem ser citados:

- Z: usado na especificação do comportamento de sistemas seqüenciais;
- CSP, CCS, Temporal Logic, I/O Automata: utilizados na especificação do comportamento de sistemas concorrentes.

Já o processo formal de Verificação apresenta duas abordagens: Verificação de Modelos , “Model Checking” [Henzinger], [Atlee], [Henzinger 1] e Prova de Teorema. O processo de Verificação caminha além da Especificação, pois ela é utilizada para analisar um sistema sob o enfoque de propriedades desejadas.

A Verificação de Modelos corresponde ao método que constrói um modelo finito de um sistema e verifica que certas propriedades desejadas são atendidas pelo modelo. Na realidade, a verificação é realizada como uma pesquisa exaustiva no espaço de estados com término garantido em função de se tratar de um modelo finito. Este método teve seu início na verificação de hardware e protocolos e a tendência atual é sua aplicação na análise da especificação de software. Em contrapartida à Prova de Teorema, a Verificação de Modelos pode ser automática e mais rápida. Pode ser usada para verificar a especificação por etapas, auxiliando em sistemas que ainda não foram totalmente especificados.

A principal desvantagem da Verificação de Modelos corresponde ao problema do número muito grande de estados. As ferramentas para Verificação de Modelos atuais têm uma expectativa de manipular sistemas de 100 a 200 variáveis de estado, gerando uma quantidade enorme de estados alcançáveis.

Há duas abordagens utilizadas para a Verificação de Modelos: A Verificação de Modelos Temporais, “Temporal Model Checking” e o Autômato, “Automaton”.

Na abordagem do temporal, a especificação é expressa numa lógica temporal e os sistemas são modelados como máquinas de transição de estados finitos. Um procedimento eficiente é a utilização para a verificação se o modelo de transição de estados finitos corresponde a um modelo para a especificação. Neste tipo de representação se enquadram os Autômatos Híbridos.

Por outro lado, na abordagem por Automato, o autômato do sistema/subsistema é comparado com a especificação para verificar se seu comportamento está em harmonia com a especificação.

A Prova de Teorema, por outro lado, é uma técnica onde tanto o sistema como suas propriedades desejadas são expressas através de fórmulas em alguma lógica matemática. São definidos um conjunto de axiomas e um conjunto de regras de inferência. A Prova de Teorema é um processo de provar uma propriedade a partir dos axiomas do sistema. Essas provas podem ser auxiliadas por ferramentas computacionais denominadas Provadores de Teorema Semi-Automáticos, “Machine-Assisted Theorem Proving”. Como estas ferramentas requerem interação com o ser-humano, seu processo é lento e sujeito a erros. Por outro lado, o projetista ganha uma visão privilegiada do sistema e de suas propriedades. Esse provadores têm sido utilizados na verificação matemática de propriedades de sistemas críticos, tanto em projeto de hardware como de software. Em contraste com o método de Verificação de Modelos, a Prova de Teorema pode manipular com espaços infinitos de estados. A técnica corresponde à indução estrutural para provar aspectos sobre um domínio infinito.

Todos os métodos formais, apresentados a seguir, podem gerar todos os caminhos possíveis a partir de um determinado estado do sistema, além de determinar se alguns dos estados alcançáveis são ou não perigosos. Em função

do nível de abstração técnico do método, ele pode ser mais aplicável ao sistema, onde o nível de abstração exigido é maior ou em partes do sistema mais críticas. Em cada método será apresentada uma breve discussão sobre sua aplicabilidade mais adequada. [Bowen 96].

8.5.1.10. Avaliação Quantitativa da Segurança de Aplicações Microprocessadas

A avaliação quantitativa é o último passo de um processo de Análise de Perigo, dentro da etapa Análise Final de Perigo. No método aqui apresentado são feitas algumas considerações considerando o sistema microprocessado triplicado TMR – “Triple Modular Redundancy”. [Camargo 01] Geralmente nestes sistemas de segurança crítica, são utilizadas técnicas de redundância em combinação com técnicas de blocos “fail-safe”, que normalmente se constituem no gargalo destes sistemas como, por exemplo, comparadores e votadores, no caso do sistema TMR. Nos sistemas microprocessados não é recomendável realizar uma análise de risco sem considerar a presença do software, em especial quando a segurança é dependente do software. Neste trabalho a função do bloco microprocessado é dividida em módulos funcionais, de diagnóstico e de reconfiguração. A avaliação das falhas inseguras está relacionada com módulos funcionais. Os módulos de diagnóstico irão influenciar o fator de cobertura de falhas e os módulos de reconfiguração influenciam na modelagem final do sistema.

As taxas de falhas do hardware e do software podem agrupadas visando obter a taxa de falhas global do sistema. Para esta decisão foram adotadas algumas hipóteses [Iyer 85]: Os módulos funcionais estão logicamente em série num determinado módulo do sistema;

- Quando um módulo funcional falha, ele não influencia na falha de um outro bloco funcional. Um subconjunto de erros de software tem grande proximidade com erros no hardware. Um erro pode causar uma falha num bloco único. Tais erros são denominados erros de software relacionados com o hardware. [Houtermans 98] Neste trabalho erros deste tipo são classificados como erros de hardware. São considerados erros de software aqueles que não apresentam conexão com falhas no hardware;

- Erros de projeto do hardware não são considerados pois o projeto do hardware microprocessado já apresenta grande maturidade;
- As taxas de falhas são expressas em relação a um tempo absoluto, e função de uma referência comum de tempo; e
- Deve-se ter o cuidado de não contar duas vezes a mesma falha, que poderia acontecer no caso de módulos funcionais em série.

A taxa de falhas de um canal simples pode ser assumida como a soma da taxa de falhas seguras e da taxa de falhas inseguras. [Houtermans 98]

$$\lambda_{\text{singleboard}} = \lambda_{\text{Ssingleboard}} + \lambda_{\text{Usingleboard}}$$

A falha segura representa uma falha que respeita os Requisitos Gerais de Segurança, caso contrário, ela será considerada falha insegura.

A avaliação da taxa de falhas de um canal microprocessado deve se basear em probabilidades de eventos bem definidos. Considerando a existência de erros de projeto de software, a ocorrência destes erros no canal simples, corresponde a um evento bem definido. [Garrett 99] Neste caso, a taxa de falha do software deve ser igual à taxa de ocorrência das condições associadas àquele perigo. Por outro lado, a taxa de falhas do hardware corresponderá às falhas decorrentes do desgaste do hardware envolvido. Neste método não foram consideradas falhas transientes. As falhas transientes devem ser avaliadas através de métodos qualitativos apresentados nesta tese. Feitas estas considerações pode-se dizer que:

$$\lambda_{\text{singleboard}} = \lambda_{\text{HW}} + \lambda_{\text{SW}}$$

$$\lambda_{\text{singleboard}} = (\lambda_{\text{SHW}} + \lambda_{\text{UHW}}) + (\lambda_{\text{SSW}} + \lambda_{\text{USW}})$$

onde λ_{HW} corresponde à taxa de falhas do hardware, λ_{SW} à taxa de falha do software, λ_{SHW} à taxa de falha segura do hardware, λ_{UHW} à taxa de falha insegura do hardware, λ_{SSW} à taxa de falha segura do software e λ_{USW} à taxa de falha insegura do software.

Um grande problema reside na determinação da porcentagem da taxa de falha do canal simples $\lambda_{\text{singleboard}}$ que corresponde às falhas seguras e inseguras. Esta

decisão é altamente dependente da aplicação e, desta forma, é altamente dependente do software utilizado. [Bastt 98] Além deste aspecto, esta taxa de falhas inseguras pode ser dividida em duas categorias: detectável λ_{UD} e não detectável λ_{UND} . [Laprie 90]

A taxa de falhas inseguras do canal simples pode ser então avaliada como:

$$\lambda_{\text{Singleboard}} = \lambda_{UHW} + \lambda_{USW}$$

$$\lambda_{\text{Singleboard}} = (\lambda_{UDHW} + \lambda_{UNDHW}) + (\lambda_{UDSW} + \lambda_{UNDSW})$$

onde λ_{UDHW} corresponde à taxa de falhas inseguras detectáveis do hardware, λ_{UNDHW} corresponde à taxa de falhas inseguras não detectáveis do hardware, λ_{UDSW} corresponde à taxa de falhas inseguras detectáveis do software e λ_{UNDSW} à taxa de falhas inseguras não detectável do software. Pesquisas futuras devem ser realizadas na estimativa de λ_{UNDSW} e λ_{USW} já que existem estudos em como estimar λ_{SW} . [Wright 94] [Schneidewind 97]

Considerando cada módulo funcional de hardware, a taxa de falhas pode ser avaliada através do modelo combinatório série, de acordo com a seguinte equação:

$$\lambda_{\text{MODULE}} = \sum \lambda_{\text{COMPONENT}}$$

Para os módulos funcionais de hardware com circuitos digitais de baixa complexidade, a taxa de falhas dos componentes pode obtida, por exemplo, de normas internacionais como a do Departamento de Defesa Norte Americano – MIL-HDBK-217. [Military Handbook]. Os modos de falhas destes componentes são bem conhecidos e através da técnica FMECA podem ser determinados os modos de falhas perigosos detectáveis e não detectáveis. Assim a avaliação destas taxas de falhas pode ser realizada da seguinte forma:

$$\lambda_{UDCOMP} = \frac{\lambda_{\text{COMPONENT}}}{\sum cfm} * \sum udfm$$

$$\lambda_{UNDCOMP} = \frac{\lambda_{\text{COMPONENT}}}{\sum cfm} * \sum uufm$$

onde Σ_{cfm} corresponde ao número total de modos de falhas do componente, Σ_{udfm} ao número de modos de falhas perigosos detectáveis, Σ_{uufm} ao número de modos de falhas perigosos não detectáveis, todos determinados através do FMECA. O valor λ_{UDCOMP} corresponde à taxa de falhas insegura detectável do componente e $\lambda_{UNDCOMP}$ à taxa de falha insegura não detectável do componente.

Entretanto, é extremamente complexo analisar o modo de falhas de componentes de alta escala de integração (LSI/VLSI) e seu efeito num sistema crítico. Não é possível, nestes casos, realizar um FMECA dos blocos funcionais nem determinar as taxas de falhas inseguras. Nestes casos, são propostas três faixas de falhas inseguras:

- $\lambda_U = 0.1 \% \cdot \lambda_{MODULE}$ (menos pessimista)
- $\lambda_U = 1.0 \% \cdot \lambda_{MODULE}$
- $\lambda_U = 10.0 \% \cdot \lambda_{MODULE}$ (mais pessimista)

Outro aspecto importante com relação aos módulos funcionais de hardware com componentes LSI/VLSI é a determinação de falhas inseguras não detectáveis. Este tipo de falha num canal simples permanece no sistema e, no caso de utilização de técnicas redundantes, uma falha compensatória em outro canal pode conduzir o sistema a uma condição insegura. Os valores da taxa de falhas inseguras detectáveis e não detectáveis do sistema irão influenciar o valor final do MTTUF – “Mean Time to Unsafe Failure” do sistema de acordo com a seguinte expressão:

$$MTTUF = \int_0^{\infty} S(t).dt$$

onde $S(t)$ corresponde à probabilidade do sistema permanecer num estado seguro.

É apresentado a seguir um método de avaliação da taxa de falha insegura do canal simples.

$$\lambda_{UDHW} = (A_{SW} * C_{SW} + A_{HW} * C_{HW}) * \lambda_{UHW}$$

onde:

- A_{SW} - Disponibilidade dos recursos de hardware necessários para que o software, que detecta falhas no hardware, seja executado;
- C_{SW} - Diagnóstico implementado por software, que determina o fator de cobertura do software; [Leveson 90]
- A_{HW} - Disponibilidade de outros recursos de hardware, necessários para detectar falhas, quando o software não está sendo executado;
- C_{HW} - Diagnóstico implementado através destes outros recursos de hardware, que determina o fator de cobertura do hardware.

8.5.1.11. Injeção de Falhas

O conceito essencial fundamentando a aplicação do método de Injeção de Falhas num sistema computacional está no processo computacional, que realiza transições sobre diversas entradas, passando através de estados intermediários, e gerando saídas. A Injeção de Falhas corresponde à ação de adicionar uma nova transição ou retirar uma existente ao processo e observar seus efeitos na saída do sistema. Neste aspecto é fundamental realizar esta atividade num ambiente real de operação. Através deste método é possível verificar o efeito da falha em todo sistema e não apenas localmente. [Voas 98]

A Injeção de Falhas pode ser aplicável às entradas, aos diversos estados do sistema, às suas informações, com o propósito de verificar o que ocorre com suas respectivas saídas. Desta forma, podem ser inseridas falhas de hardware ou software, cuja simulação do comportamento pode ser inserido através de técnicas de software, evitando assim ações destrutivas no teste do sistema. Como o método de Injeção de Injeção de Falhas é aqui utilizado dentro do processo de Análise de Perigo, seu enfoque deve ser o de avaliar o efeito na saída, verificando o atendimento ou não aos Requisitos Gerais de Segurança. Este método pode inclusive ser utilizado para auxiliar na pesquisa dos valores das taxas de falhas inseguras relacionados com os sistemas computacionais e que já foi discutido um método apresentado no item anterior. Desta forma, a Injeção de Falhas pode auxiliar em fornecer dados mais confiáveis a serem utilizados num processo de Avaliação Quantitativa da segurança de sistemas microprocessados.

9. Referências Bibliográficas

ARTIGOS

[Accurso 99] Accurso, A.; Timóteo, C. A. F.; Freitas, J. H. Z.; Borloni, R. N. Operar com Segurança. Pgs 94-98. Revista Engenharia. Operação Metrô de São Paulo. 1999.

[Alur] Alur, R. Henzinger, T.A., Ho, P.H. Automatic Symbolic Verification of Embedded Systems. 37 pgs. Bell Labs, Lucent Technologies, Department of Computer Science, Cornell University, University of California at Berkley.

[Atlee], Atlee, J. ChechiK, Gannon, J. Using Model Checking to Analyze Requirements and Designs. 23 pgs. University of Maryland, USA, and University of Waterloo, Canada.

[Bartolomeu 2000] Bartolomeu, C. CNS/ATM no Mundo. Pgs28-36. Revista Aeroespço. Agosto 2000. Edição Especial. CNS/ATM

[Bastt 98] Bastt,W.;Bock,H.W; German Qualification and Assessment of Digital IEC Systems important to safety; Reliability Engineering and System Safety, 59, 1998, 163-170. Elsevier Science Limited

[Beerthuizen 01] Beerthuizen, P. G.; Kruidhof, W. System and software safety analysis for the ERA control computer. Reliability Engineering and System Safety, 71, 2001, 285-297. Elsevier Science Limited

[Boeing 97] Air Traffic Managment Concept Baseline Definition. Prepared by Boeing Commercial Airplane Group. Nextor Report #RR-97-3. October 31, 1997.

[Bohnenblust 98] Bohnenblust, H. Slovic, P. Integrating technical analysis and public values in risk-baesd decision making. Reliability Engineering and System Safety 59. 1998. Pgs 151-159. Elsevier Science Limited.

[Bonifácio 99] Bonifácio, A.L.; Moura, A. M.; Camargo Jr., J.B.; Almeida Jr.,J.R. Análise, Verificação e Síntese de Segmentos de Via de uma Malha Metroviária. Relatório Técnico IC-99-45. 48 páginas. Agosto 1999. Instituto de Computação. UNICAMP.

[Bowen 96] Bowen, J.P.; Butler, R.W.; Dill,D.; Glass, R. L.; Gries, D.; Hall, A.; Hinchey, M.G.; Holloway, C.M.; Jackson, D.; Jones, C.; Lutz, M.J.; Parnas, D.L.; Rushby, J.; Wing, J.; Zave, P. An Invitation to Formal Methods. IEEE Computer. Pgs 16-30. April 1996.

[Broomfield 97] Broomfield, E.J.; Chung, P.W.H. Safety assessment and the software requirements specification. Reliability Engineering and System Safety 55. 1997. Pgs 295-309. Elsevier Science Limited.

[Bonifácio 99]Bonifácio, A.L; Moura, A.V; Camargo Jr.,J.B; Almeida Jr.,J.R. Análise e Verificação de Segmentos de Via de uma Malha Metroviária. II Workshop on Formal Methods, pags 13-22, realizado de 12 a 13 de outubro de 1999 em Florianópolis, SC, Brasil.

[Butler 93] BUTLER, R. W.;FINELLI, G. B. The infeasibility of quantifying the reliability of life-critical real-time software. IEEE Transactions on Software Engineering, v.19, n.1, p.3-12, Jan. 1993.

[CANSO 99] Demystifying CNS/ATM. CANSO CNS/ATM WORKING GROUP. Final Version June 1999.

[Camargo 01] Camargo Jr., J.B., Canzian, E., Almeida Jr., J.R., Paz, S.M., Basseto, B.A., Quantitative Analysis Methodology in Safety-Critical Microprocessor Applications, Reliability Engineering & System Safety 74 – 2001. Pgs 53-62. Elsevier Science Limited .

[Camargo 99]Camargo JR.,J.B; Almeida JR., J. R. Applying HAZOP to a Subway Signaling System. Artigo aceito para ser publicado na Conferência “ISSC’99 – 17th International System Safety Conference – System Safety at

the Dawn of a New Millenium”, a ser realizada em Orlando, Florida, USA, de 16 a 21 de agosto de 1999.

[Clarke] Clarke, E.M.; Wing, J.M. Formal Methods: State of the Art and Future Directions. 22 pgs. Carnegie Mellon University.

[Eames 99] Eames, P. D.; Moffett, J. The Integration of Safety and Security Requirements. In: Safecom 1999, Toulouse, France, September 1999.

[Garrett 99] Garrett, C.; Apostolakis, G. Context in the Risk Assessment of Digital Systems; Risk Analysis, vol 19, No.1, 1999.

[Hamlet 92] Hamlet,D. Are we testing for true reliability?. IEEE Software, v.9, n.4, p.21-7, July 1992.

[Haxthausen 00] Hazthausen, A.E.; Peleska,J. Formal Development and Verification of a Distributed Railway Control System. IEEE Transaction on Software Engineering, Vol 26, NO 8, pgs 687-701, August 2000.

[Hebbron 97] Hebbron, B.D.; Fenelon, P. The applicationof hazard and operability studies to real time structured requirements models. Reliability Engineering and System Safety 55. 1997. Pgs 311-325. Elsevier Science Limited.

[Henzinger] Henzinger, T.A; Ho, P.H.; Wong-Toi, H. HyTech: A Model Checker for Hybrid Systems. 23 pgs. Research suppoerted by ONR YIP, NSF CAREER, AFOSR, ARO MURI, ARPA and SRC.

[Henzinger 1] Henzinger, T.A. The Theory of Hybrid Automata. 24 pgs. Electrical Engineering and Computer Sciences. University of California at Berkeley.

[Houtermans 98]Houtermans, M.J.M.;Rouvroye,J.L.; The Influence of Design Parameters on the Performance of safety —Related Systems; Eindhoven University of Technology – TUV Product Services Inc., 1998.

[Iyer 85] Iyer, R. K.; Velardi, P. Hardware-Related Software Errors: Measurement and Analysis. IEEE Transactions on Software Engineering, Vol SE-11, No.2, February 1985.

[Jaffe 91] Jaffe, M. S.; Leveson, N. G.; Heimdahl, M. P. E. M.; Bonnie E. Software requirements analysis for real time process control systems. IEEE Transactions on Software Engineering, v.17, n.3, p.241-58, Mar. 1991.

[Ladkin 01] Ladkin, P.B. Na Example of Everyday Risk Assessment. Faculty of Technology. University of Bielefeld. 12 pgs. Februaray 2001.

[Laprie 90] Laprie, Jean C.; Arlat, J.; Beouner, C.; Anoun, K.; Definition and Analysis of Hardware and Software Fault Tolerant Architectures; LAAS – CNRS, Computer – July 1990 – 39-51.

[Lawrence 97] Lawrence, J.D.; Gallagher, J.M. A proposal for performing software safety hazard analysis. Reliability Engineering and System Safety 55. 1997. Pg 267-282. 1997 Elsiver Science Limited.

[Lawrence 00] Lawrence, J.D. Software qualification in safety applications. Reliability Engineering and System Safety 70. 2000. Pg 167-184. 2000 Elsiver Science Limited.

[Leveson 90] Leveson, N.G.; Cha, S.S.; Knight, J.C.; Shimeall, T.J.; The Use of Self Checks and Voting in Software Error Detection: An Empirical Study; IEEE Transactions on Software Engineering, Vol 16., No 4, April 1990.

[Leveson 86] Leveson, N. G. Software safety: why, what, and how. Computing Surveys, v.18, n.2, p.25-163, June 1986.

[Leveson 83] Leveson, N. G.; Harvey, P. Analyzing software safety. IEEE Transactions on Software Engineering, v.9, n.5, p.569-79, Sept. 1983.

[Machado 2000] Machado, W.C.C. O GNSS Transitório Brasileiro. Pgs14-22. Revista Aeroespaço. Agosto 2000. Edição Especial. CNS/ATM

[Melchers 01] Melchers, R.E. On the ALARP approach to risk managment. Reliability Engineering and System Safety 71, 2001, pgs 201-208. Elsevier Science Limited.

[Moura 00] MOURA,A.; BONIFÁCIO,A.; CAMARGO Jr.,J.B.; RADY Jr..J.R. Formal Parameter Synthesis for Track Segments of a Subway Mesh. 7th Annual IEEE International Conference and Workshop on the Engineering of Computer Based Systems, a ser realizada de 3 a 7 de abril de 2000 em Napier University, Edinburh, Scotland, UK.

[Naufal 01] NAUFAL JR, J.K.; CAMARGO JR, J.B.; ALMEIDA JR., J.R.; CUGNASCA, P.S. Avaliação da Influência da Precisão do Posicionamento das Aeronaves nos Níveis de Segurança do Controle de Tráfego Aéreo. In: Congresso SAE BRASIL, São Paulo – SP, Brasil, 19/11 a 22/11/2001. Artigo n. 230.

[Rady 00] RADY A.,J.R.; CAMARGO Jr.,J.B; BASSETO, B.A.; CUNHA, R.S.; PAZ,S.M. An Inspection List for Critical Software Analysis. In: PSAM 5 -International Conference on Probabilistic Safety Assessment and Management, Osaka, Japan, de 27/11 a 01/12/2000.

[Redmill 97] Redmill, F.; Chudleigh, M.F.; Catmur, J.R. Principles underlying a guideline for applying HAZOP to programmable electronic systems. Reliability Engineering and System Safety 55. 1997. Pgs 283-293. Elsevier Science Limited.

[Renn 98] Renn, O. The role of risk perception for risk managment. Reliability Engineering and System Safety 59. 1998. Pgs 49-62. Elsevier Science Limited

[Rushby 94] RUSHBY,J. Critical system properties: survey and taxonomy. Reliability Engineering & System Safety, v.43, n.2, p.189-219, aug. 1994.

[Sheldon92] Sheldon, F. T.; Kavi, K. M.; Tausworthe, R. C.; Yu, J.; Brettschneider, R.; Everett, W. W. Reliability measurement: from theory to practice. IEEE Software, v.9, n.4, p.13-20, July 1992.

[Schneidewind 97] Schneidewind, N. F. Reliability Modeling for Safety-Critical Software, IEEE Transaction on Reliability, Vol.46, NO.1, 1997 March.

[Yu 98] Yu, W. D. A Software Fault Prevention Approach in Coding and Root Cause Analysis. Bell Labs Technical Journal. Pg 3-21. April-June 1998.

[Wright 94] Wright, D.; Cai, K. Representing Uncertainty for Safety Critical Systems. Draft PDCS2 Project Deliverable, City University, Northampton Square, London EC1V 0HB, May 5, 1994.

LIVROS

[Friedman 95] Friedman, M. A; Voas, J. M.; Software Assessment: Reliability, Safety, Testability. John Wiley & Sons, Inc. 1995.

[Galotti 99] Galotti Jr., V. P. The Future Air Navigation System (FANS). Ashgate. ISBN 0 291 39833 2. 1999.

[Hall 97] Hall, E. M. Managing Risk – Methods for Software Systems Development. SEI Series in Software Engineering. Addison-Wesley. ISBN 0-201-25592-8. 1997.

[Hurn 89] Hurn, J. GPS – A Guide to the Next Utility. Trimble. 1989.

[Hurn 93] Hurn, J. GPS – Differential GPS. Trimble. 1993.

[Johnson 89] Johnson, B. W. Design and Analysis of Fault Tolerant Digital Systems. University of Virginia, Addison-Wesley Publishing Company. 1989.

[Kececioglu 91] Kececioglu, D. Reliability Engineering Handbook, Volume 2. Prentice Hall, Englewood Cliffs, New Jersey, ISBN 0 13 772302 4, 1991.

[Leveson 95] Leveson, N. G. Safeware – System Safety and Computers. University of Washington. Addison-Wesley Publishing Company. 1995.

[Myerson 96] Myerson, M. Risk Management Processes for Software Engineering Models. Artech House. ISBN 0-89006-653-3.

[Profit 95] Profit, R. Systematic Safety Management in the Air Traffic Services. Euromoney Publications. ISBN 1 85564 470 3. 1995.

[Redmill 99] Redmill, F.; Chudleigh, M.; Catmur, J. System Safety: HAZOP and Software HAZOP, John Wiley & Sons, ISBN 0 471 98280 6, 1999.

[Storey 96] Storey, N. Safety-Critical Computer Systems. Addison-Wesley Publishing Company. 1996.

[Voas 98] Voas, J.M. Software Fault Injection. John Wiley & Sons, ISBN 0-471-18381-4. 1998.

[Walters 00] Walters, J.M.; Sumwalt, R.L. Aircraft Accident Analysis: Final Reports. 2000. McGraw Hill. ISBN 0-07-135149-3.

NORMAS

[DD ENV 50126:1999] Railway applications – The specification and demonstration of Reliability, Availability, maintainability and Safety (RAMS). EN 50126. CENELEC - European Committee for Electrotechnical Standardization. September 1999.

[DD ENV 50129:1999] Railway applications Safety related electronic systems for signalling. – European Presatandard ENV 50129. CENELEC – European Committee for Electrotechnical Standardization. May 1998

[EN 50128] Railway applications – Software for railway control and protection systems. CENELEC – European Committee for Electrotechnical Standardization. July 1998.

[IEC 61508] Functional Safety Electrical/Electronic/Programmable Electronic Safety-related Systems. International Electrotechnical Commission. IEC 61508. 1997.

[Military Handbook] Military Handbook – Reliability Prediction of Electronic Equipment -Department of Defense, Washington DC, 1990.

[NASA 96] NASA Software Safety Standard, NSS. 1740.13. 1996.

[NBR 9126] Tecnologia de Informação – Avaliação de Produto de Software – Características de Qualidade e Diretrizes para o seu Uso. 1994.