Sistemas Computacionais Críticos:

Conceituação e Técnicas de Programação Defensiva

João Batista Camargo Junior
Professor Associado (Poli-USP)

Lúcio F. Vismari
Engº de Segurança de Sistemas (FDTE)
Pesquisador, B.Eng., M.Eng. (GAS/Poli-USP)

Grupo de Análise de Segurança
Departamento de Engenharia da Computação e Sistemas Digitais
Escola Politécnica da Universidade de São Paulo

AGENDA

1. O Grupo de Análise de Segurança da Poli-USP
2. Sistemas Computacionais em Aplicações Críticas
3. Programação Defensiva (PD)
4. Exemplos de PD em sistemas críticos
5. Comentários Finais e Discussão

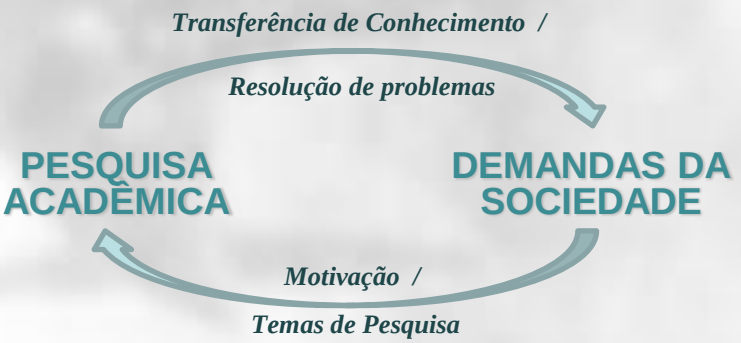
AGENDA

1. O Grupo de Análise de Segurança da Poli-USP
2. Sistemas Computacionais em Aplicações Críticas
3. Programação Defensiva (PD)
4. Exemplos de PD em sistemas críticos
5. Comentários Finais e Discussão

1. GAS/POLI-USP

O GAS (Grupo de Análise de Segurança) é um **Grupo de Pesquisa** do Departamento de Engenharia da Computação e Sistemas Digitais (PCS) da **Escola Politécnica da USP** (Poli-USP).

ATUAÇÃO em **PESQUISAS ACADÊMICAS** e **PROJETOS DE EXTENSÃO (CONSULTORIA)** relacionados com **DEPENDABILIDADE** (em especial **Segurança (*Safety*)**, **Confiabilidade** e **Disponibilidade**) de **Sistemas Computacionais** em **Aplicações Críticas**, tais como Sistemas Metroferroviários, Transporte Aéreo, Sistemas Nucleares, Sistemas de Defesa, Indústria Aeroespacial, entre outros.



FDTE : fundação de apoio à **Escola Politécnica da USP** (Poli-USP) que realiza, entre outros, a **interface administrativa** (gerência de contratos e de RH) dos projetos de extensão realizados entre a **Universidade** (Poli-USP) e **Sociedade – Iniciativa Público e/ou Privada**.

Áreas de atuação – Pesquisa Acadêmica

Os principais temas abordados pelo GAS Grupo de Análise de Segurança são:

- Garantia de Dependabilidade em Aplicações Críticas
  - Metodologias de Análise de Segurança, Confiabilidade e Disponibilidade.
  - Gestão de Perigo e de Risco. Resiliência.
  - Sistemas Computacionais Tolerantes a Falhas.
- Certificação visando Dependabilidade.
- Confiabilidade e Usabilidade Humana.
- Safety *versus* Security

Áreas de Atuação em parceria com o INCT SEC :

- Metodologia de Avaliação de Segurança ("safety") de Sistemas Críticos
  - **Certificação de Sistemas Embarcados Críticos**
  - **Verificação e Validação (V&V) para VANTs e inserção no ATM**
- Desenvolvimento de sistemas computacionais críticos em segurança
  - **Desenvolvimento orientado à Tolerância a Falhas.**

Áreas de atuação – Projetos de Extensão

Exemplos de PROJETOS DE EXTENSÃO (CONSULTORIA) - finalizados ou em realização:

PROJETOS PARA A COMUNIDADE CIVIL (Público / Privada)

- **Sistemas Metro-Ferroviários (desde 1981)**
  - **Análise de Segurança Independente de Sistemas de Sinalização e Controle** (**METRO-SP**, VALE S.A., METRÔ-RJ, TRENSURB-RS, DATAPROM, ...)
- Sistemas de Distribuição de Energia Elétrica
  - Análise de Disponibilidade de Sistemas de Distribuição (CPFL)

PROJETOS MILITARES E DE SEGURANÇA NACIONAL

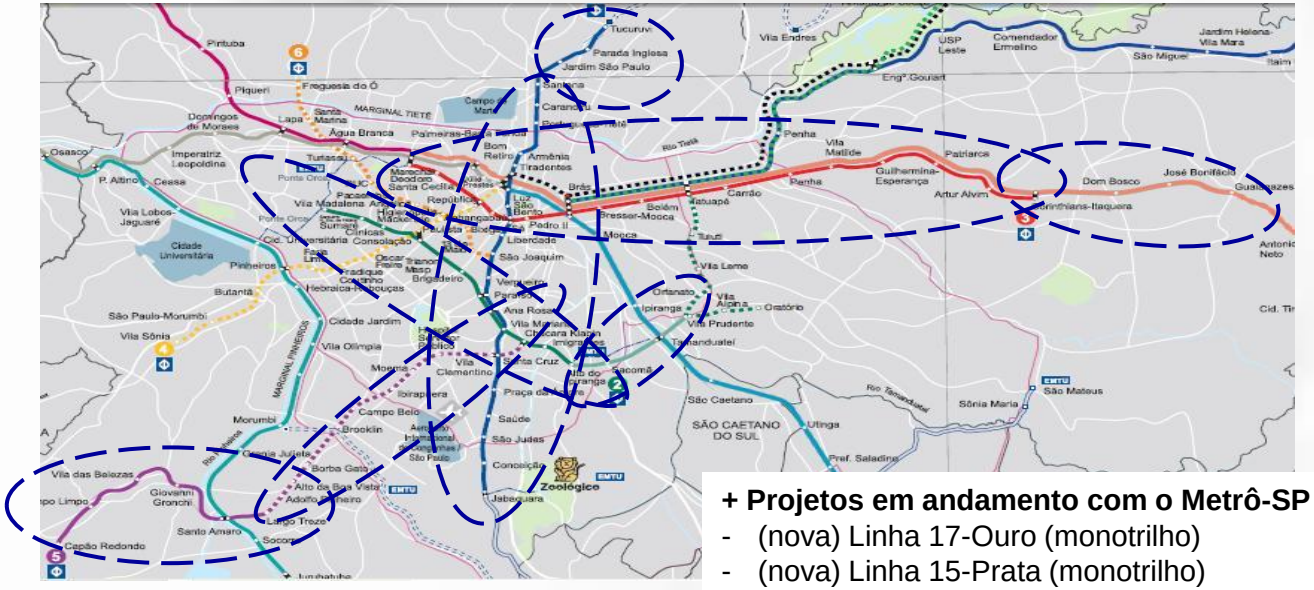
- Centro Tecnológico do **Exército** Brasileiro (CTEx)
  - Análise de Segurança e Disponibilidade do RADAR de Defesa Antiaérea (SABER M60)
- Centro Tecnológico da **Marinha** em São Paulo – (CTMSP)
  - Análise de Segurança e Disponibilidade do Sistema de Supervisão e Controle do Submarino Nuclear Brasileiro
- Comando da **Aeronáutica**
  - Análise Independente de Segurança e Disponibilidade.

Áreas de atuação – Projetos de Extensão

Análise de Segurança Independente do Sistema de Sinalização e Controle do METRÔ DE SÃO PAULO

- Sistemas E/E/PE (**Hardware** e **Software**) Procedimentos Operacionais e de Manutenção, Arquitetura de Sistema, etc.
- ATC (*Automatic Train Control*) de Bordo e de Via.

ATUAÇÃO DO GAS/Poli-USP COMO 'ISA' NO METRÔ DE SÃO PAULO (DESDE 1981)



Mapa das Linhas do Metrô de São Paulo

+ Projetos em andamento com o Metrô-SP (ISA do SSC):
- (nova) Linha 17-Ouro (monotrilho)
- (nova) Linha 15-Prata (monotrilho)
- (modernização da) Linha 5-Lilás
- (modernização das) Linhas 1-Azul, 2-Verde e 3-Vermelha

AGENDA

1. O Grupo de Análise de Segurança da Poli-USP

2. **Sistemas Computacionais em Aplicações Críticas**

   "Serão apresentados os **fundamentos da engenharia de Dependabilidade** (sobretudo de segurança crítica - *safety*) de **sistemas computacionais em aplicações críticas** ..."

3. Programação Defensiva (PD)

4. Exemplos de PD em sistemas críticos

5. Comentários Finais e Discussão

SISTEMAS CRÍTICOS:

Sistemas nos quais a ocorrência de

'FALHAS'

pode levar a

CONSEQUÊNCIAS INACEITÁVEIS

Mortes e ferimentos,
Prejuízos ambientais e materiais,
Não cumprimento de missão,
Perda$ e Dano$,...

Operadores
Usuários
Ambiente
Terceiros

Segundo Sommerville, existem 3 tipos principais de sistemas críticos:

Sistemas Críticos em Negócios (*Business-critical system)*, onde **falhas** podem causar **perdas** econômicas significativas aos negócios.

(*ex.: sistemas de reservas, sistemas de e-commerce, ...*)

Sistemas Críticos em Missão (*Mission-critical system***)**, onde **falhas** podem causar **perda** (não cumprimento) da missão.

(*ex.: sistema autônomo de guiamento de espaçonave, sistemas de bagagens*)

Sistemas Críticos em Segurança (*Safety-critical system)*, onde **falhas** podem causar **perdas e/ou danos** à vida humana e ao ambiente.

(*ex.: marca-passo cardíaco, sistema de sinalização ferroviário, ...*)

Sommerville (2011). Software Engineering. A.Wesley
Sommerville (1998). Requirements Engineering. J.Wiley

2. SISTEMAS x APLICAÇÕES CRÍTICAS

Onde as Consequências Inaceitáveis (**Perdas & Danos**) podem ocorrer

Domínio de Aplicação Crítica

Supervisão / Monitoramento

Controle / Gerência

Sistema Crítico

Onde **falhas** podem levar a **consequências** inaceitáveis (no domínio de aplicação)

Garantia dos níveis de dependabilidade.

Modos de falha novos e (ainda) desconhecidos

DOMÍNIO DA APLICAÇÃO

AFETA

EVOLUÇÃO

DEMANDA

DOMÍNIO DO SISTEMA

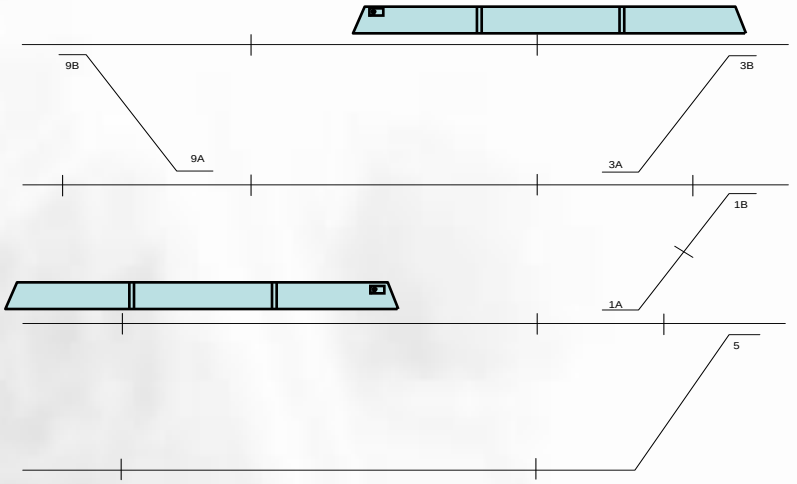
Novos desafios e demandas (capacidade, eficiência, custos, …)

Novas tecnologias / conceitos (habilitadores das novas demandas)

EXEMPLO (SAFETY):

SISTEMA METROFERROVIÁRIO

(Sistema de Sinalização e Controle)



Aplicação Crítica em Segurança (*safety*)

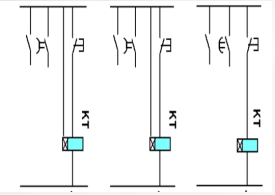
Supervisão | Controle

Sistema Crítico em Segurança (Supervisão e Controle)

ANTES

Supervisão | Controle

S&C baseado em Lógica Booleana com Relés Eletromecânicos

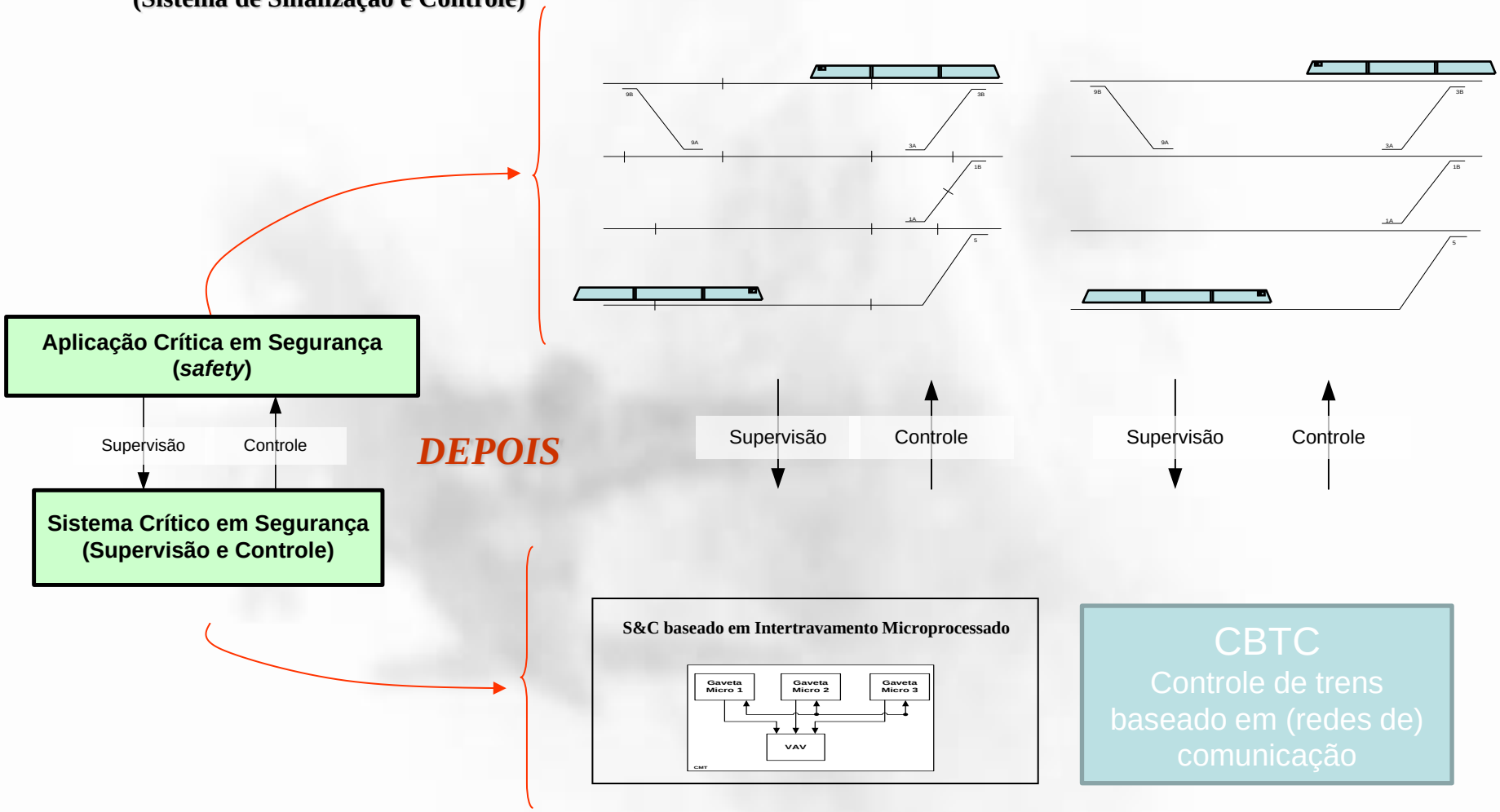


2. SISTEMAS COMPUTACIONAIS EM APLICAÇÕES CRÍTICAS

EXEMPLO (SAFETY):

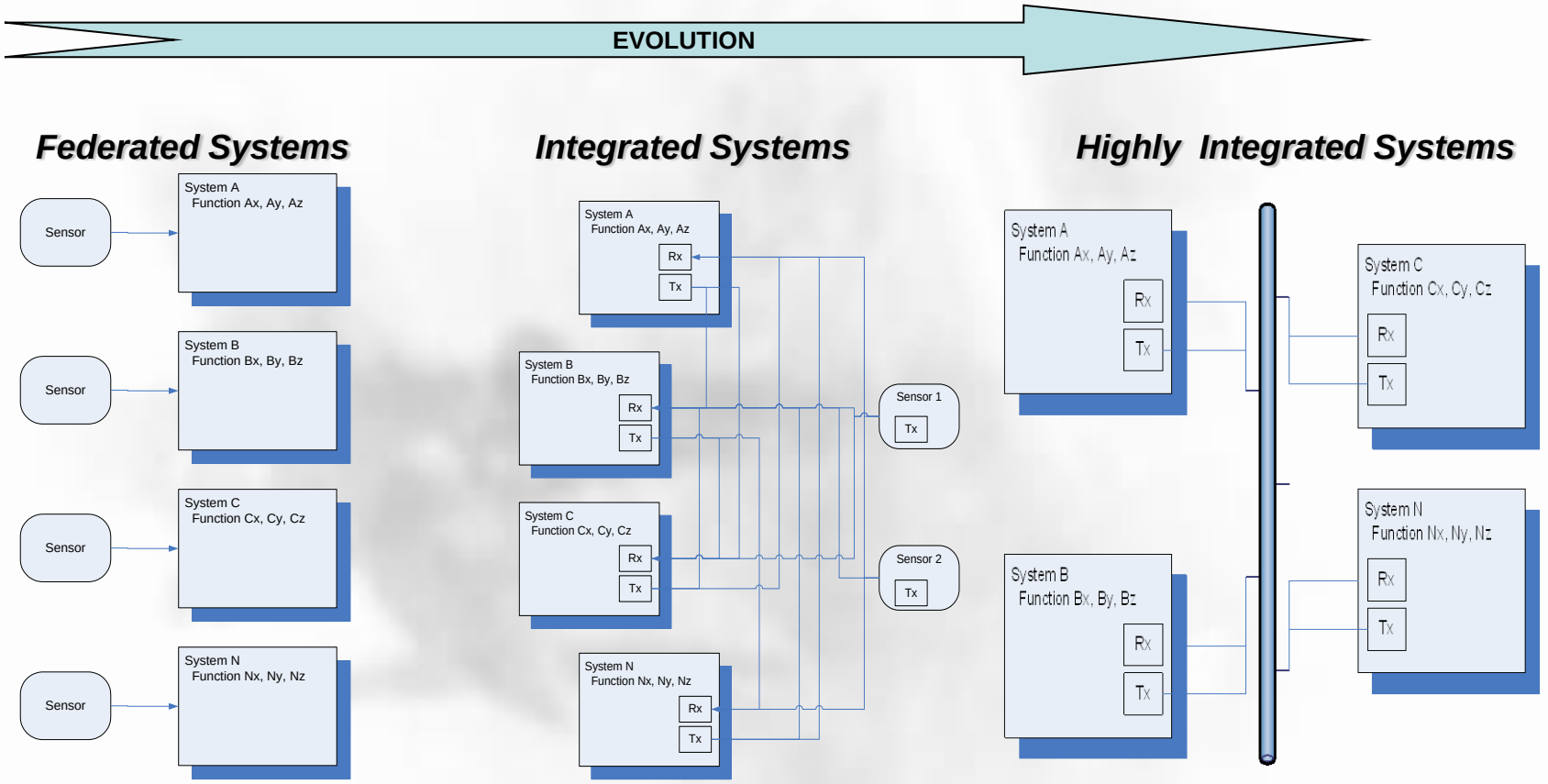
SISTEMA METROFERROVIÁRIO

(Sistema de Sinalização e Controle)



EVOLUÇÃO DO TRANSPORTE AÉREO

AERONAVES (AVIÔNICOS)



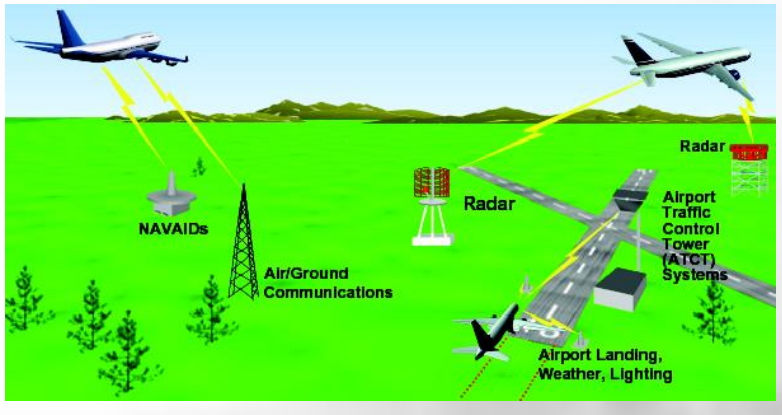
- Intensive use of mechanical, electromechanical and analog electronic systems.

- Use of analog and digital electronic systems and some electromechanical controls.

- Control systems implemented with E/E/PE modules

EVOLUÇÃO DO TRANSPORTE AÉREO

AIR TRAFFIC MANAGEMENT (ATM) SYSTEMS



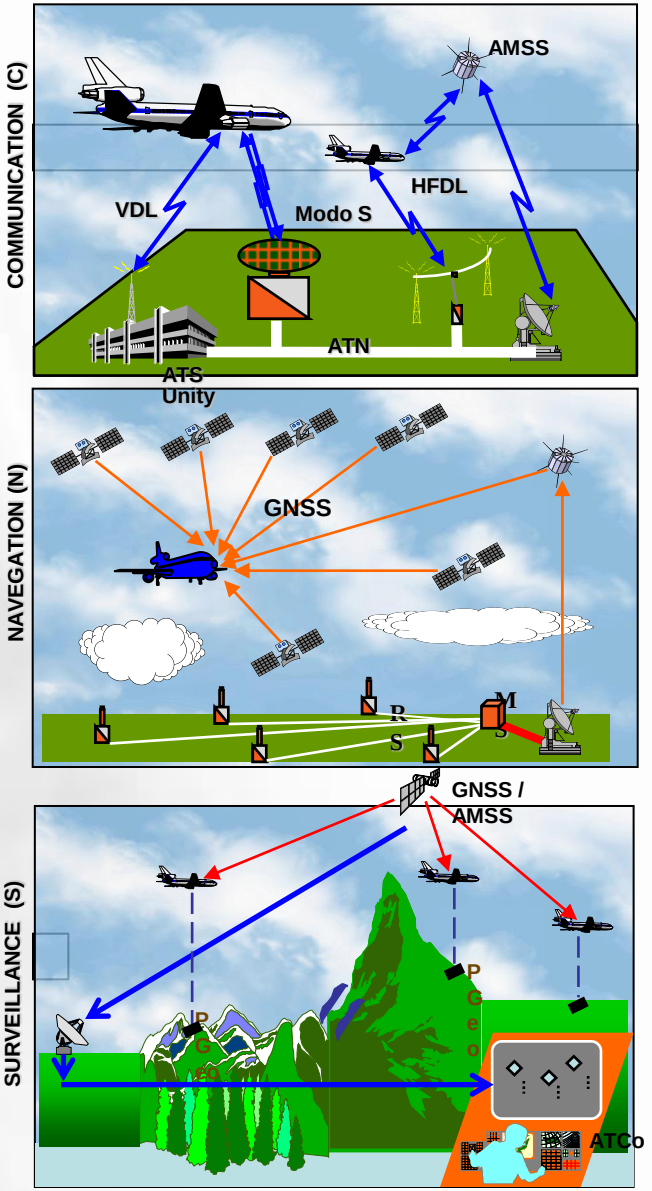
Today's Air Traffic System

- Ground-based (NavAids, Radar)
- Single Channel voice control
- Human-centered with non automated support.
- Proprietary Hardware and Software.

EVOLUÇÃO (Tecnologias e Procedimentos)

CNS/ATM System

- Satellite-based Navigation and Communication
- Data-based Control (ATN)
- Human-centered, but "automated" Decision-Making.
- Hardware COTS, Proprietary Software

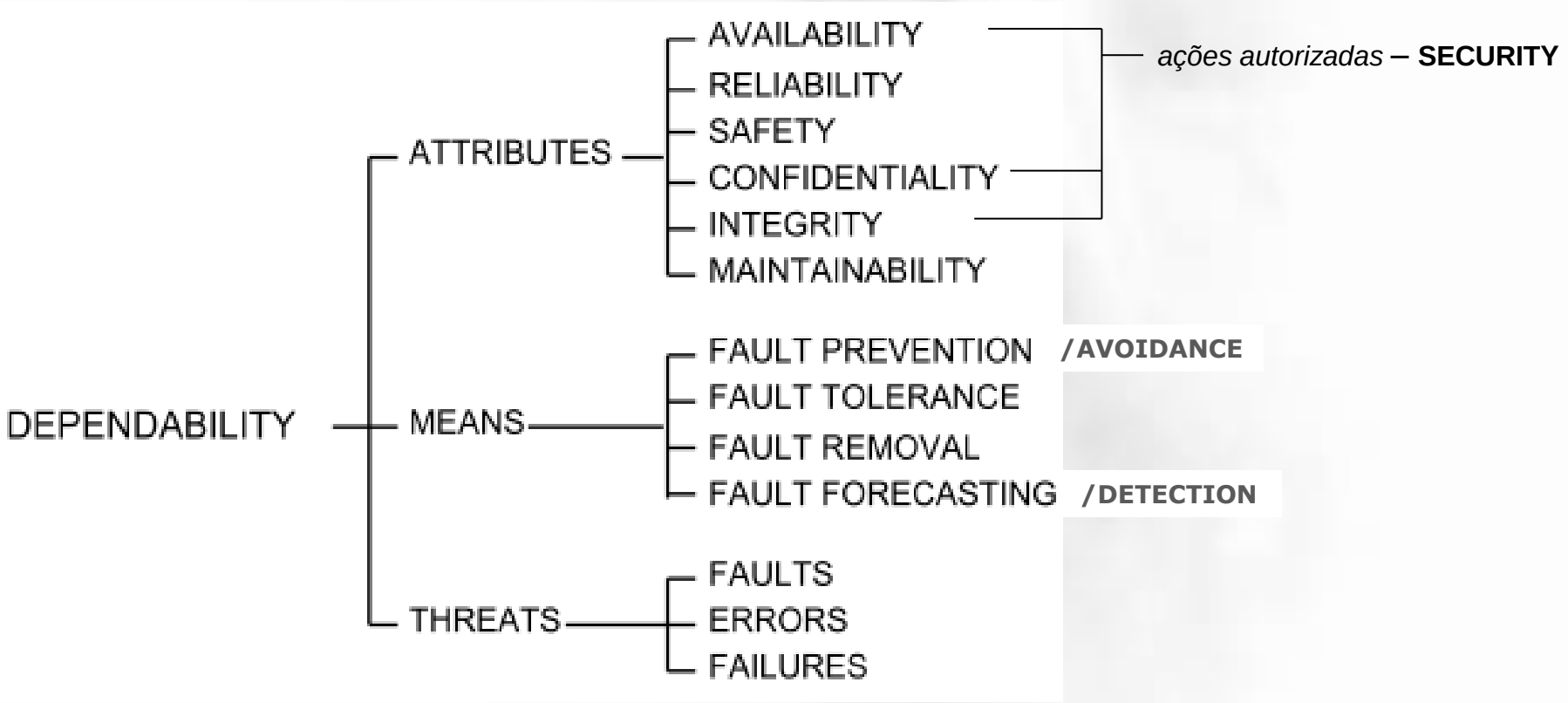


2. SISTEMAS COMPUTACIONAIS CRÍTICOS: **DEPENDABILIDADE**

"Entrega de serviço que pode ser justificadamente confiável (trusted)"

*"Evita falhas que podem ser **inaceitavelmente** frequentes e/ou severas"*

(Alto Risco de Perdas e Danos)

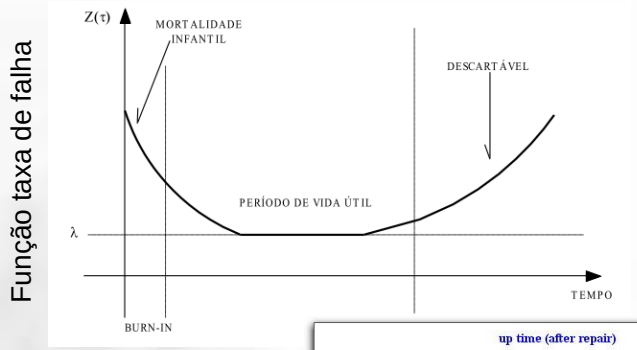


A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

ATRIBUTOS (*Attributes*)

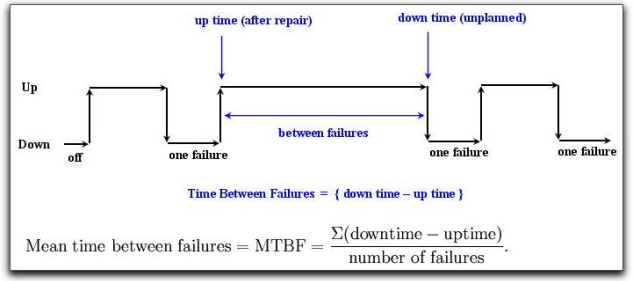
Reliability: continuity of correct service;

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



Availability: readiness for correct service;

A(assint.) = MTTF / MTBF



Maintainability: ability to undergo repairs and modifications.

Safety: absence of catastrophic consequences on the user(s) and the environment;

S(t) = R(t) considerando 'falhas' inseguras

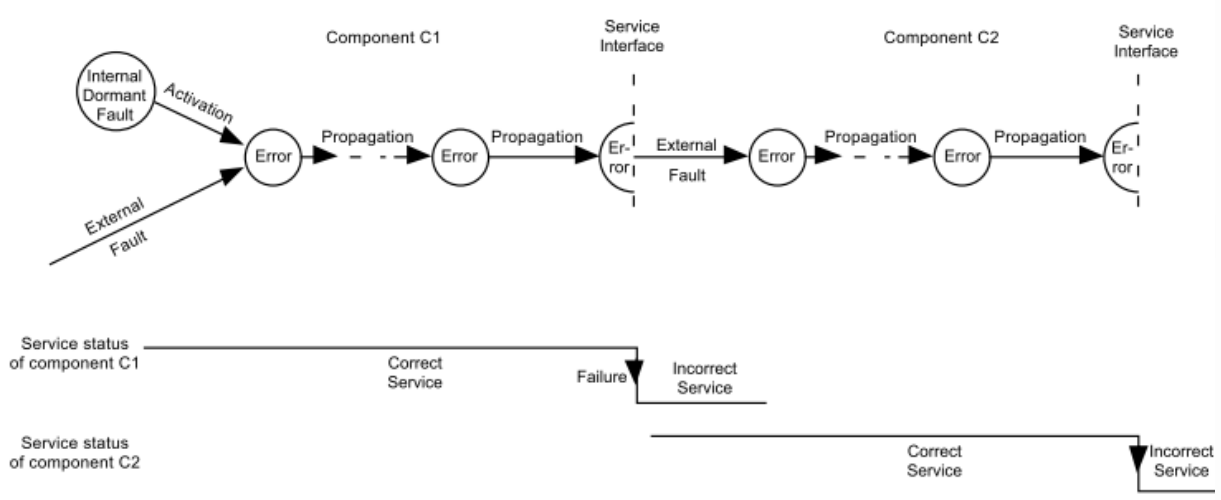
Confidentiality: absence of unauthorized disclosure of information;

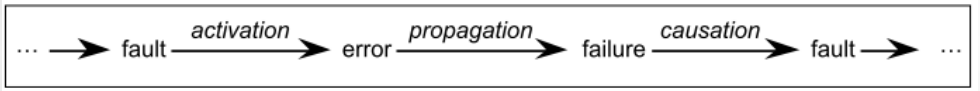
Integrity: absence of improper system state alterations;

'R.A.M.S.' [CENELEC EN 50126]

AMEAÇAS (*Threats*)

*(definições dependentes de nível de abstração – "**fronteiras do sistema**")*





FALHA (Fault): condição anormal que pode causar a redução ou perda da capacidade de um componente realizar sua função ('errar'). Provocado (ativado) por alguma vulnerabilidade no sistema (hardware ou software), podendo ter origem em erros de especificação e implementação; componentes defeituosos, etc; tornando o sistema vulnerável à distúrbios.

ERRO (*Error*): manifestação da *falha*. Discrepância entre o realizado e o especificado (desejado) e que pode levar o sistema a uma disfunção (failure).

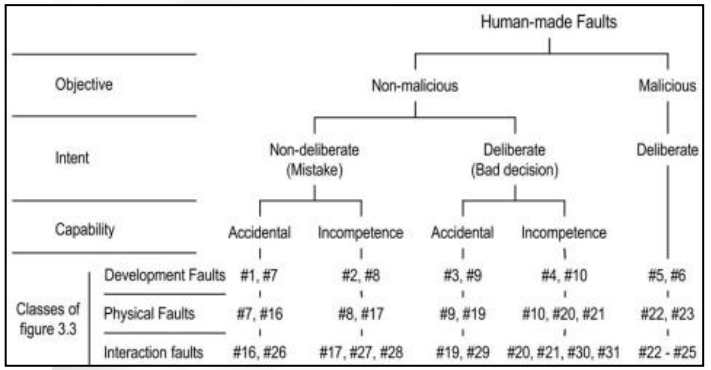
DISFUNÇÃO (*Failure*): término da habilidade do sistema prover ao seu usuário uma função, operação ou serviço da forma como foi especificado/desejado. É a manifestação (consequência) do *erro* na interface de serviço do sistema.

A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

AMEAÇAS (*Threats*)

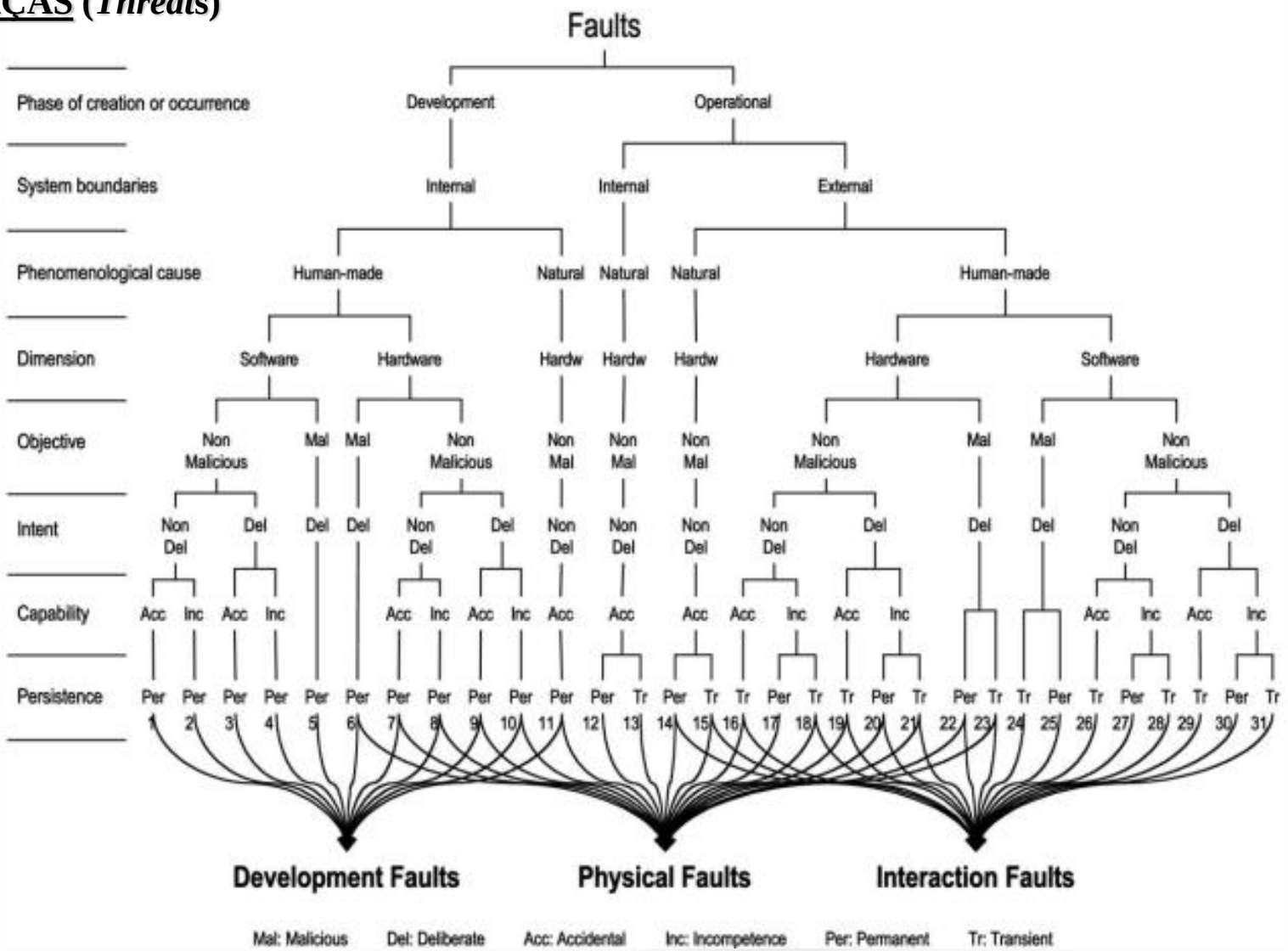
Faults

- Phase of creation or occurrence
  - **Development faults** [occur during (a) system development, (b) maintenance during the use phase, and (c) generation of procedures to operate or to maintain the system]
  - **Operational faults** [occur during service delivery of the use phase]
- System boundaries
  - **Internal faults** [originate inside the system boundary]
  - **External faults** [originate outside the system boundary and propagate errors into the system by interaction or interference]
- Phenomenological cause
  - **Natural faults** [caused by natural phenomena without human participation]
  - **Human-Made faults** [result from human actions]
- Dimension (Equipment)
  - **Hardware faults** [originate in, or affect, hardware] ..... (Physical)
  - **Software faults** [affect software, i.e., programs or data] ..... (Information)
- Objective
  - **Malicious faults** [introduced by a human with the malicious objective of causing harm to the system]
  - **Non-Malicious faults** [introduced without a malicious objective]
- Intent
  - **Deliberate faults** [result of a harmful decision]
  - **Non-Deliberate faults** [introduced without awareness]
- Capability
  - **Accidental faults** [introduced inadvertently]
  - **Incompetence faults** [result from lack of professional competence by the authorized human(s), or from inadequacy of the development organization
- Persistence (duration)
  - **Permanent faults** [presence is assumed to be continuous in time]
  - **Transient faults** [presence is bounded in time]
- **Nature**. systematic or random; accidental or intentional



A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

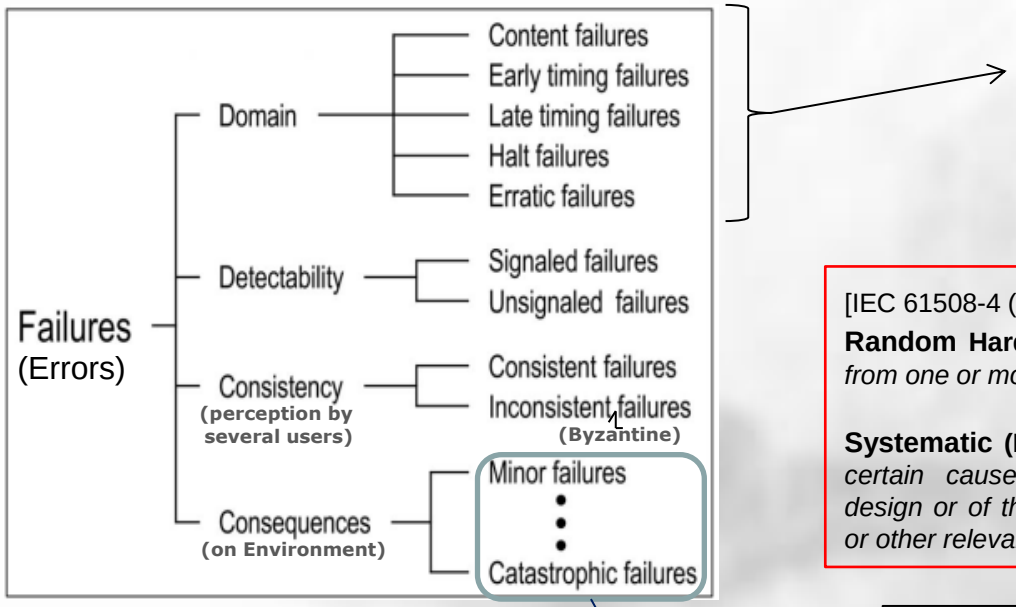
AMEAÇAS (*Threats*)

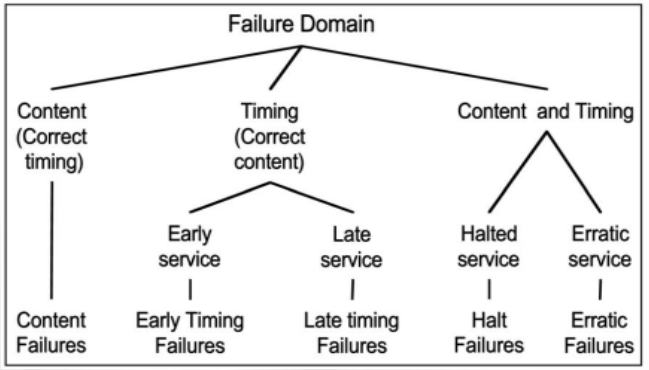


A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

AMEAÇAS (*Threats*)

A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

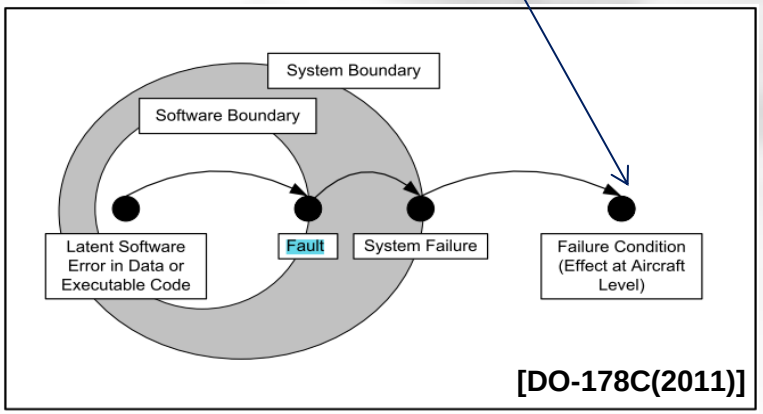




[IEC 61508-4 (2010)]

Random Hardware failure: *failure, occurring at a random time, which results from one or more of the possible degradation mechanisms in the hardware.*

Systematic (Hw/Sw) failure: *failure, related in a deterministic way to a certain cause, which can only be eliminated by a modification of the design or of the manufacturing process, operational procedures, documentation or other relevant factors.*



[DO-178C(2011)]

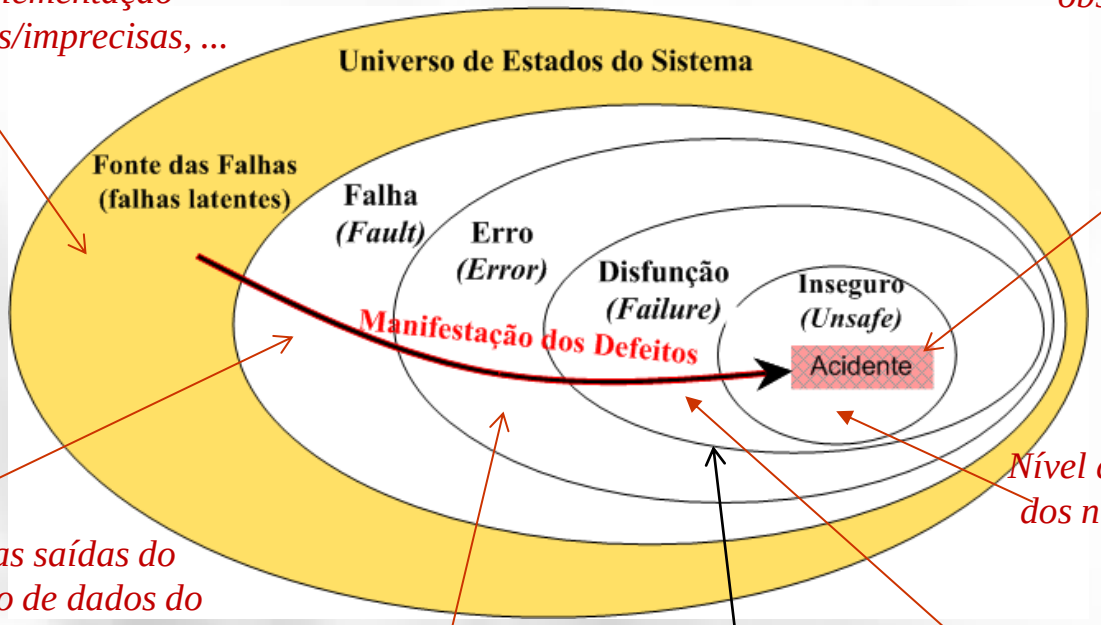
[DO-254 (2000)]
Failure - The inability of a system or system component to perform a required function within specified limits. A failure may be produced when a fault is encountered.
Fault - (1) A manifestation of a flaw in hardware due to an error or random event. A fault, if it occurs, may cause a failure. (2) An undesired anomaly in an item.
(3)[DO-178C(2011)] A manifestation of an **error** in software.

[DO-178C(2011)]
Error – With respect to software, a mistake in requirements, design, or code.

AMEAÇAS (*Threats*)

Componentes má qualidade/com defeito, EMI, Especificação/Implementação incorretas/incompletas/imprecisas, ...

Aeronave colide com obstáculos no final da pista durante a aterrissagem.



Nível de frenagem abaixo dos níveis especificados

curto-circuito em uma das saídas do controlador do barramento de dados do cartão processador.

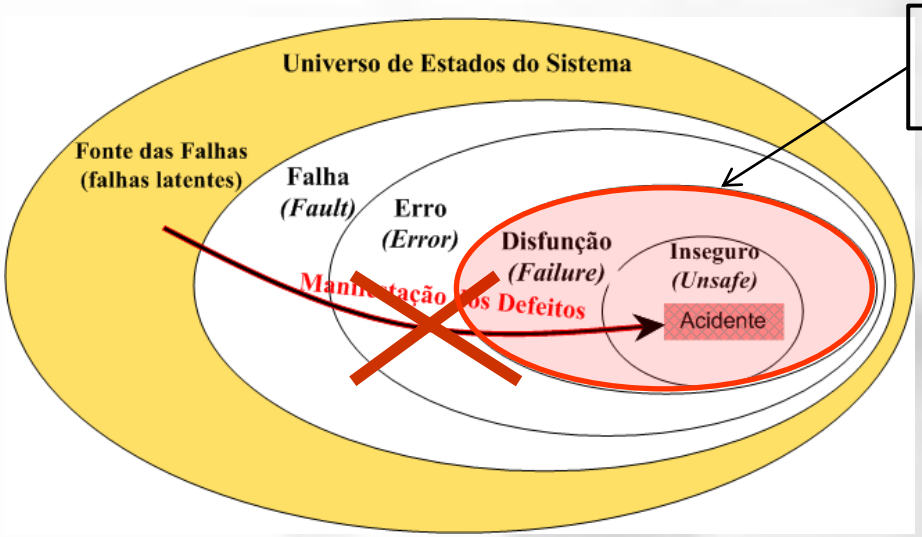
bit de um barramento de dados fica preso em nível lógico "1".

Não é possível executar o comando de abertura de uma válvula do controle hidráulico dos freios.

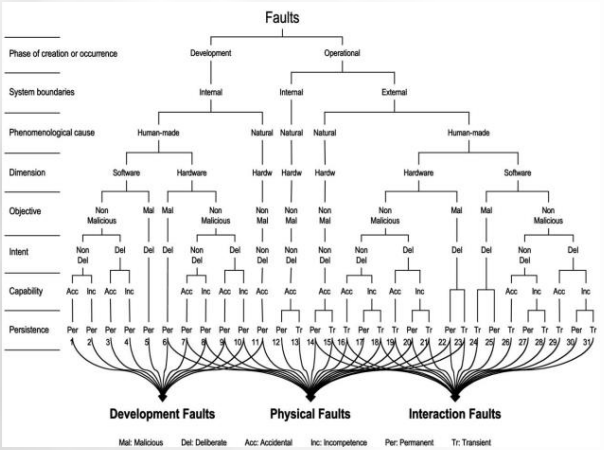
CRITICAL (ex.: *unsafe*) *versus NON-CRITICAL* (ex.: *safe*)
*FAILURES (@**consequences**)*

MEIOS (*Means*)

Garantia de Dependabilidade em Sistemas (computacionais) críticos → **Gerenciamento de FALHAS *(Fault Management)***



Risco associado a
*"FAILURES.**consequences*** ← **Cathastrophic**



(1) **Durante o projeto do sistema, <u>evitando-as</u> (prevenindo suas inserções) e/ou <u>tolerando-as</u> (lidando de forma apropriada com suas ocorrências) enquanto o sistema estiver operacional.**

➔ **Fault Avoidance / Fault Tolerance.**

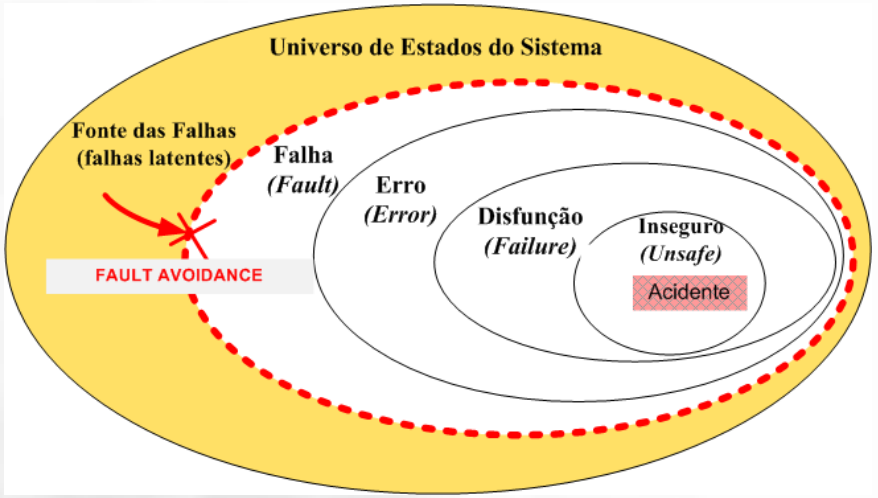
(2) **Durante a validação ou operação do sistema, <u>removendo-as</u> antes do sistema estar operacional e/ou prevendo-as (detectando-as e removendo-as antes de suas ocorrências) enquanto o sistema estiver operacional.**

➔ **Fault Removal / Fault Detection.**

MEIOS (***Means***)

GERENCIAMENTO DE FALHAS:

Fault Avoidance/Removal



Fault Avoidance/Removal: uso de técnicas e procedimentos que **previnem** a inserção de falhas durante as fases de desenvolvimento do sistema ("**construção**") ou que as **remove** antes de o sistema se tornar operacional ("**validação**"). Minimizam a quantidade (e severidade) de falhas que possam estar presentes e serem ativadas durante a operação do sistema.

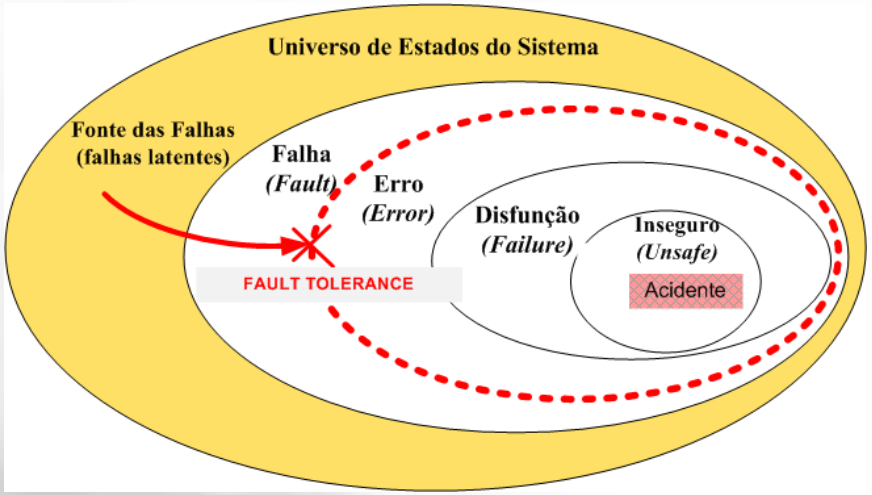
Exemplos (aplicáveis tanto em Hw quanto Sw):

- USO DE MÉTODOS FORMAIS DE DESENVOLVIMENTO (PARA ESPECIFICAÇÃO, PROJETO, VERIFICAÇÃO).
- USO DE PROJETO ESTRUTURADO, MÉTODOS DE PROGRAMAÇÃO SEGURA, EMC, …
- USO PROCESSOS DE CONTROLE DE QUALIDADE NA AQUISIÇÃO DE PARTES / REUSO.
- USO DE MÉTODOS DE V&V (INSPEÇÃO, PROVAS DE PROJETO FORMAIS, TESTES, …).

MEIOS (*Means*)

GERENCIAMENTO DE FALHAS:

Fault Tolerance



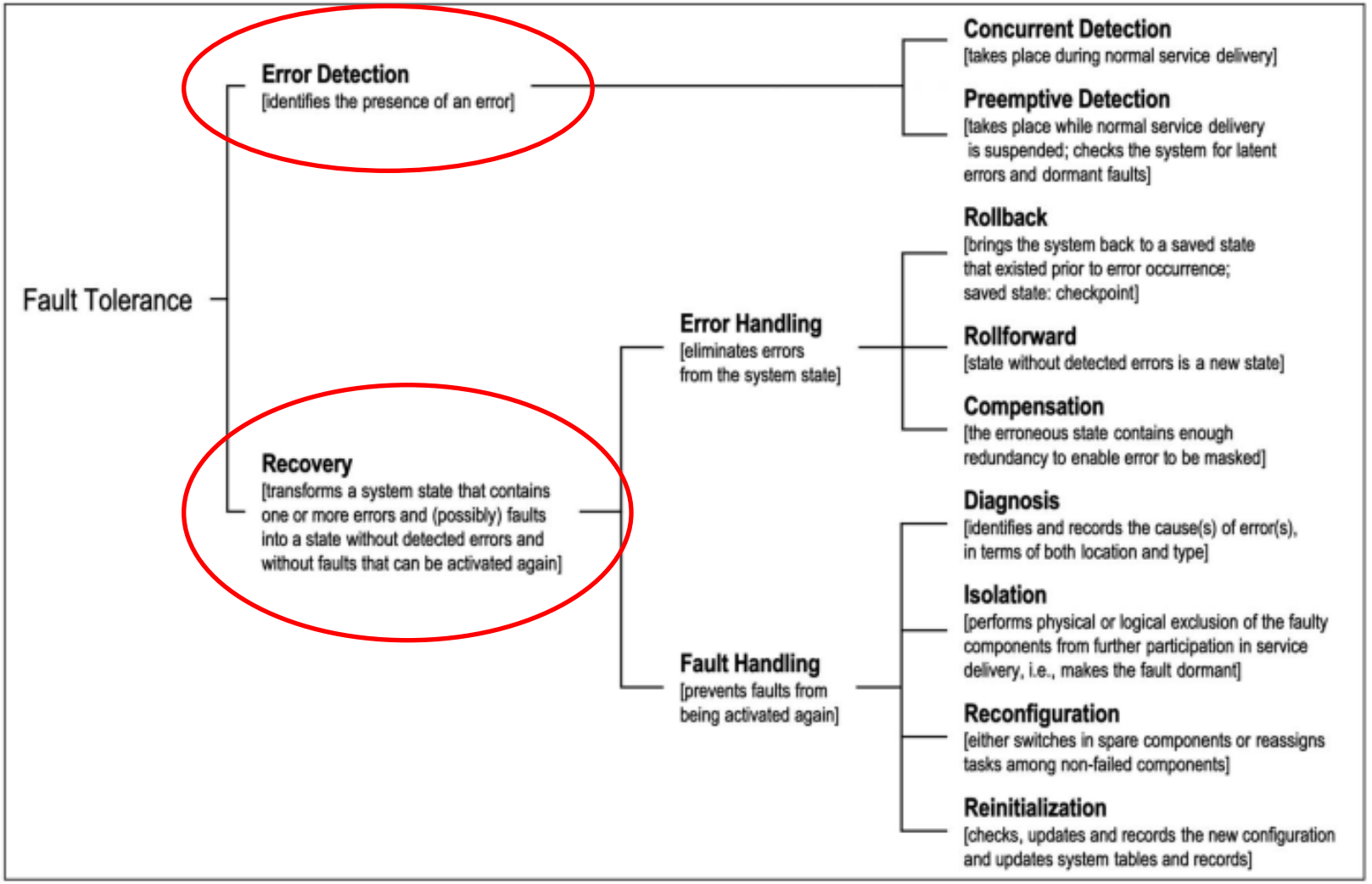
Fault Tolerance: uso de abordagens (arquiteturas, técnicas e procedimentos) durante o desenvolvimento do sistema ("**construção**") de forma que permita que ele continue operando, de forma apropriada ("*dependable*"), quando da presença falhas ativas **("*fail-safe*" / "*fail-operational*")**.

Exemplos de abordagens:

- USO DE **REDUNDÂNCIA** DE HARDWARE, SOFTWARE, DE INFORMAÇÃO/DADOS e/ou TEMPORAL.
- USO DE **DIVERSIDADE** (de PROJETO, de INFORMAÇÃO, TEMPORAL).
- **USO DE TÉCNICAS DE PROGRAMAÇÃO DEFENSIVA.**

MEIOS (*Means*)

TOLERÂNCIA A FALHAS (*Fault Tolerance*)



A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, no. 1, pp. 11–33, Jan. 2004.

Técnicas e medidas para TOLERÂNCIA A FALHAS (*Fault Tolerance*) ... exemplos

REDUNDÂNCIA: adição de informação, recursos ou tempo além do que é necessário para a **operação normal** do sistema.

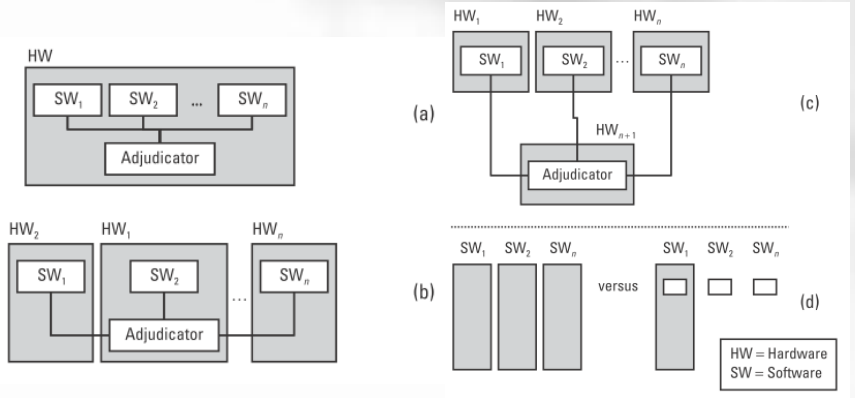
Redundância de Dados/Informação
. Código de Paridade
. Códigos m de n
. Códigos Duplicados
. Checksums (Código Separável)
. Códigos Cíclicos (Não Separável)
. Códigos Aritméticos
. Códigos Berger
. Paridade Horizontal e Vertical
. Código de Correção de Erro Hamming
. Circuitos Integrados com Correção de Erro

Redundância por Software
. Verificação de Consistência
. Verificação de Capacidade
. N_ Versões de Software

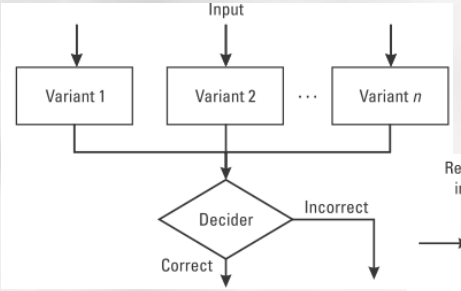
Redundância de Hardware
. Redundância de Hardware Passiva
. Redundância de Hardware Ativa
. Redundância de Hardware Hibrida

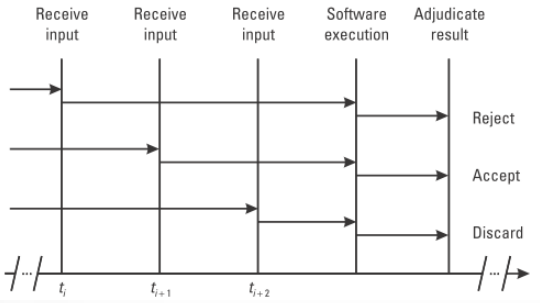
Redundância Temporal
. Detecção de Defeito Transiente
. Detecção do Defeito Permanente.
.. Lógica Alternada
.. Recomputação com Operandos Deslocados
.. Recomputação com Operando Trocados
.. Recomputação com Duplicação e Comparação (REDWC).
.. Recomputação para Correção de Erro

DIVERSIDADE: apenas redundância pode não ser suficiente para detectar e tolerar falhas (ex.: falha de modo comum).



Software Fault Tolerance Techniques and Implementation. Laura L. Pullum. 2001



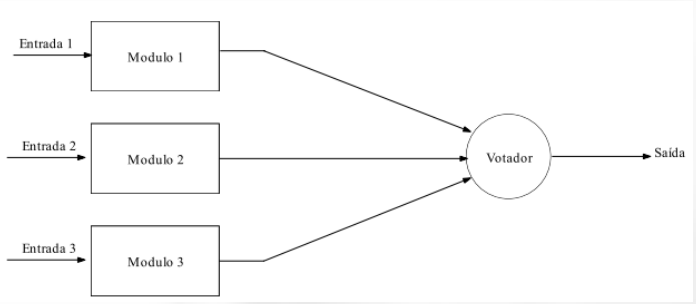


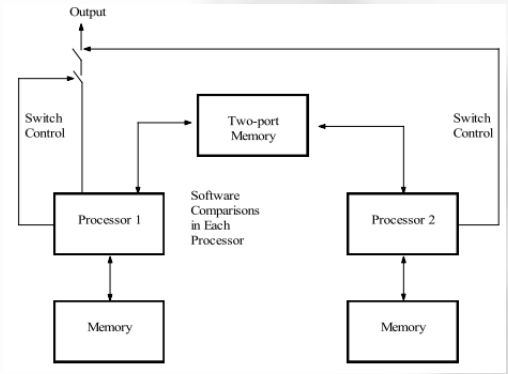
2. SISTEMAS COMPUTACIONAIS CRÍTICOS: **DEPENDABILIDADE**

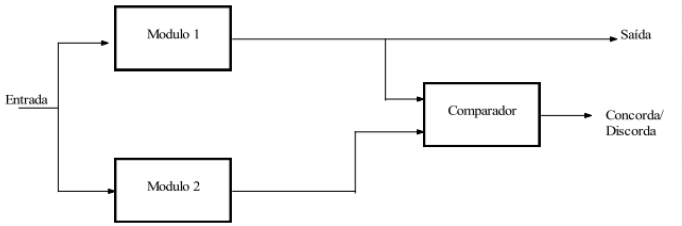
REDUNDÂNCIA: adição de informação, recursos ou tempo além do que é necessário para a **operação normal** do sistema.

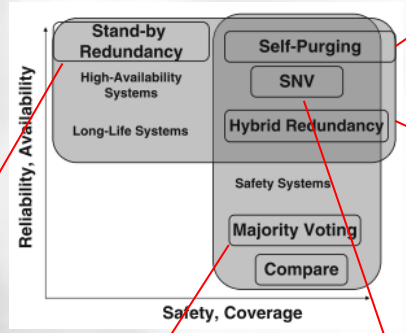
TOLERÂNCIA A FALHAS: detectar e lidar com a falha.

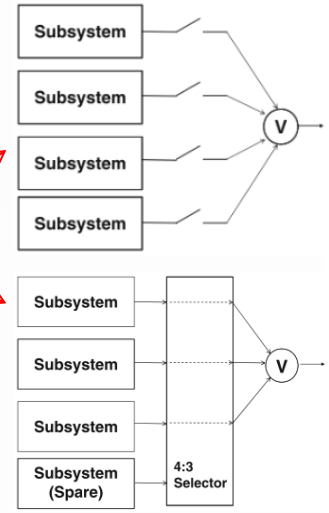
"Safety x Availability " ; "Fault x Coverage"

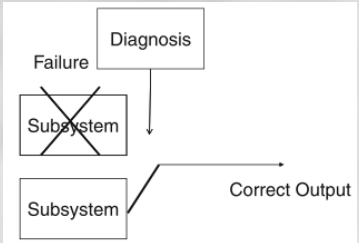


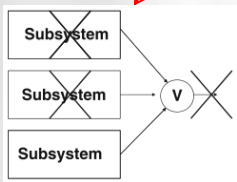




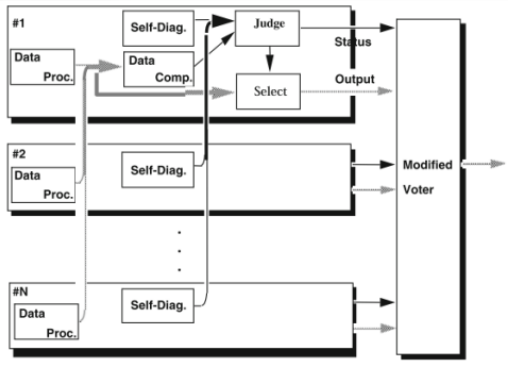








TMR



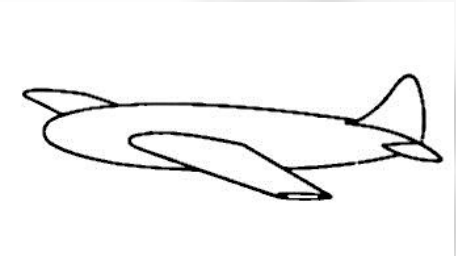
N. Kanekawa, E. H. Ibe, T. Suga, and Y. Uematsu, *Dependability in Electronic Systems*. New York, NY: Springer New York, 2011.

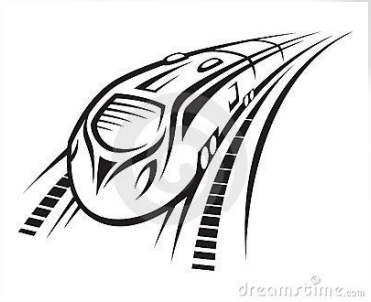
DEPENDABILIDADE

"Entrega de serviço que pode ser justificadamente confiável (trusted)"

"Evita falhas que podem ser ***inaceitavelmente*** *frequentes e/ou severas"*

(Sistemas com Alto Risco de Perdas e Danos)





???

DESENVOLVIMENTO de Sistemas Computacionais Críticos em Dependabilidade

(exemplo de Modelo de Desenvolvimento (*M. Kaâniche, J.Laprie, J.Blanquart.*))

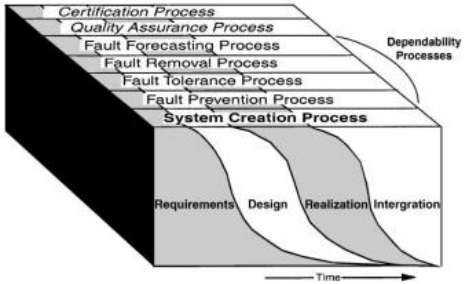


Fig. 1. Dependability-explicit development model.

System Creation Process
- Functional specifications
  - Definition of system functions (expected values and timing behavior)
  - Description of mission phases and their sequencing (phased systems)
  - Preliminary task distribution between the operators and the computing system
- Definition of the operational environment
  - System boundaries, utilization environment and user profiles
  - Operation and maintenance modes
- Development, validation and operating constraints
  - Physical constraints (weight, technology, etc.), Maintenance and operation constraints
  - Foreseeable evolutions, Reusability, Portability, Interoperability, Testability

Fault Prevention Process
- Formalisms and languages
  - Standards, rules and certification requirements, development environment, formalisms and tools
- Project organization
  - Definition of life-cycle models, Assignment of tasks to project teams, Resource management
- Project planning and project risk assessment
  - Identification of risks and means for risk reduction
  - Selection of development strategies and technologies
  - Planning of development stages and definition of transition criteria between stages
  - Project reviews planning, Configuration management planning

Fault Tolerance Process
- Description of system behavior in presence of failures
  - Identification of relevant dependability attributes and necessary tradeoffs
  - Failure modes and acceptable degraded operation modes
  - Maximum tolerable duration of service interruption for each degraded operation mode
  - Number of consecutive and simultaneous failures to be tolerated for each degraded operation mode

Fault Removal Process
- Verification Planning
  - Static verification techniques, Testing strategies (testing criteria, test generation techniques)
  - Specification of test-beds and environment simulators
- Verification assumptions
  - Classes of functions, behavior and expected faults to be analyzed by each verification technique
  - Predicates to be verified
- Verification of the requirements
  - Functional and behavioral analyses , Reviews and inspections of the specification
  - Prototyping, User-based validation, Expert reviews

Fault Forecasting Process
- Expression of dependability objectives
  - Definition of quantification measures and assignment of quantified targets
- Analysis of failure modes and their consequences on delivered service
  - Identification of failure modes
  - Classification of failures by severity
  - Specification of the classes of faults and failures to be addressed
- Fault forecasting assumptions
  - Modeling assumptions and parameters
- Function-by-function dependability allocation
  - Classification of system functions by criticality levels
- Fault forecasting planning
  - Selection of appropriate methods and tools for qualitative analysis and quantitative evaluation
  - Definition of a data collection environment
- Data collection and analysis
  - Feed-back from existing products , Follow-up of the product under development

Fig. 3. Checklist and guidelines corresponding to the Requirements phase.

System Creation Process
- Architecture definition
  - Structure: Decomposition into layers and/or components (human operators, hardware, software)
  - Behavior: System states and events for each layer and interaction between layers
  - Data types, data flow and interfaces between components
- Selection of development technologies
  - Identification of hardware and software components
- Identification of reusable components
  - Definition of necessary adjustments
- Operation and maintenance procedures preparation
- Integration plan preparation
  - Architecture integration strategy
  - Planning of system integration in its operational environment

Fault Prevention Process
- Formalisms and languages
- Project management
- Project planning and project risk assessment

Fault Tolerance Process
- Description of system behavior in presence of faults
  - Fault assumptions (faults considered, faults discarded)
- System partitioning
  - Fault tolerance structuring: (Fault containment regions, Error confinement regions)
  - Fault tolerance application layers
- Fault tolerance strategies
  - Redundancy, Functional diversity, Defensive programming, Protection techniques, etc.
- Error handling mechanisms
  - Error detection, Error diagnosis, Error recovery
- Fault handling mechanisms
  - Fault diagnosis, Fault passivation, Reconfiguration
- Identification of single points of failure

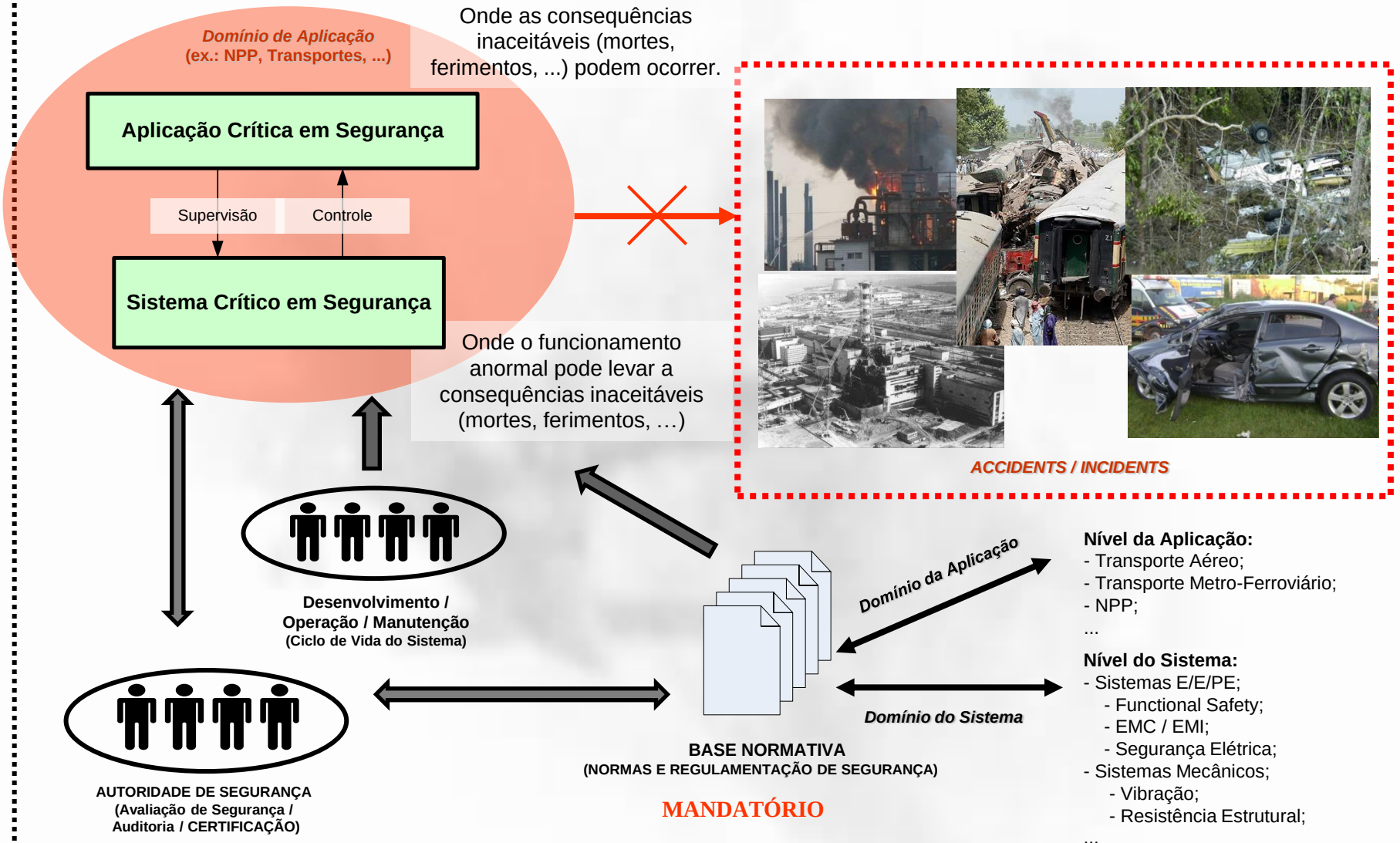
Fault Removal Process
- Verification assumptions
  - Classes of expected faults to be analyzed, Predicates that should hold
- Verification of the design
  - Behavioral analyses
  - Reviews and inspections of the design
  - Prototyping and simulation for human-machine interfaces verification
  - Design verification against the requirements
- Verification of fault tolerance mechanisms
  - Error and fault handling algorithms (formal) verification
  - Simulation-based fault injection (fault models)
- Unit and integration testing planning
  - Testing strategies (testing criteria, test case generation methods)
  - Fault injection strategies
- Definition of functional and structural verification scenarios
- Verification of evaluation results

Fault Forecasting Process
- Fault forecasting assumptions
- Analysis of failure modes and their consequences on the service to be delivered
- Component-by-component dependability allocation
- Preliminary assessment of the system dependability
  - Analytical models, Simulation
  - System scaling: Redundancies (human operators and computing system), Coverage
- Data collection and analysis

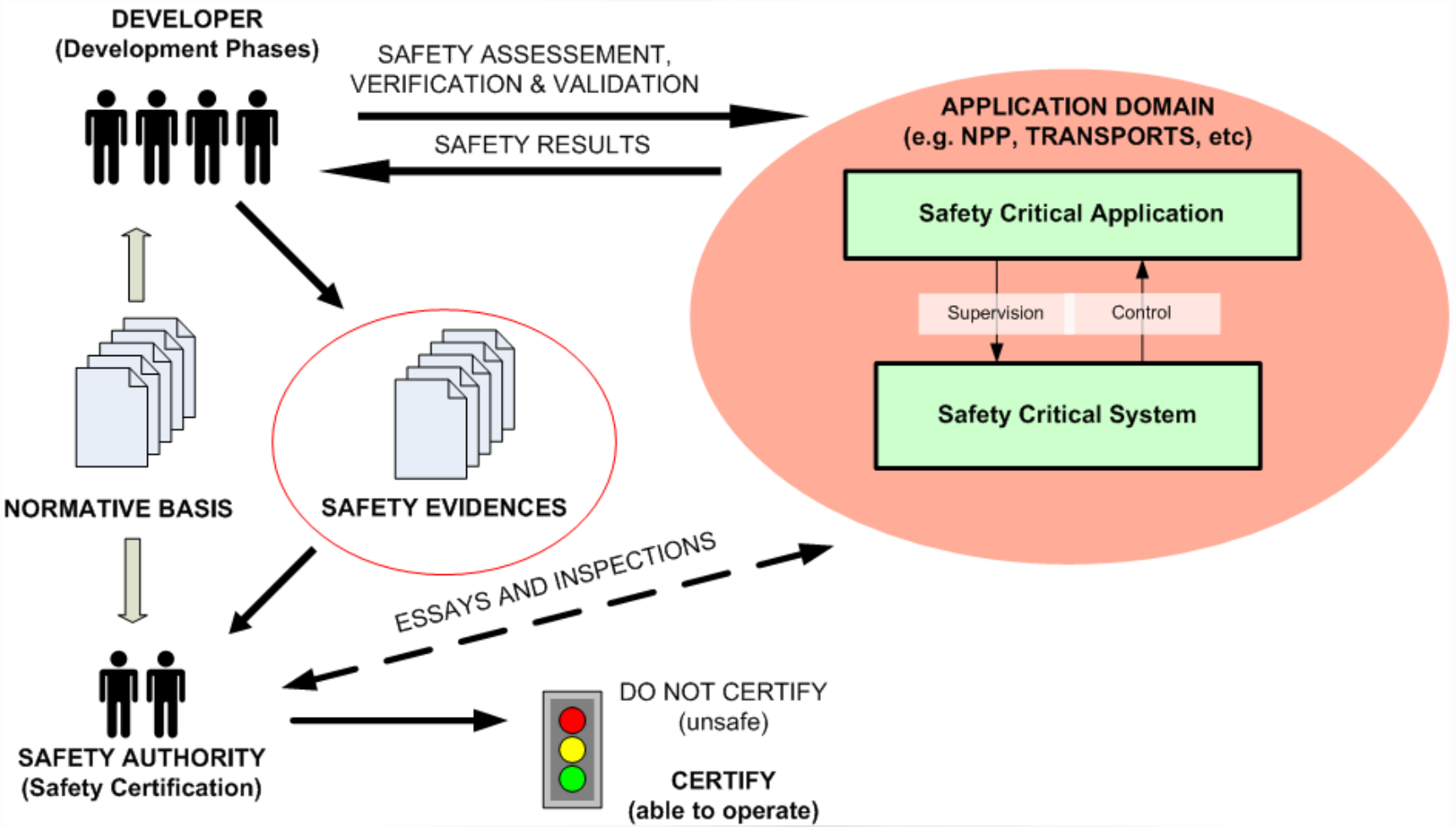
Fig. 4. Checklist and guidelines corresponding to the Design phase.

M. Kaâniche, J.Laprie, J.Blanquart. “A framework for dependability engineering of critical computing systems,” *Saf. Sci.*, vol. 40, no. 9, pp. 731–752, Dec. 2002.

O macro processo de segurança ....



O macro processo de segurança ....



General structure used by the Developers and Certifiers
to <u>identify unsafe conditions</u> and to <u>ensure the safety level</u> (Safety Authority)

DESENVOLVIMENTO de Sistemas Computacionais Críticos em Dependabilidade

Domínio de aplicação crítica (sobretudo em safety) possui bases normativas (e regulatórias) consolidadas

Exemplos de NORMAS relacionadas ao uso de Sistemas Computacionais Críticos em Transportes

Transporte Metro-Ferroviário

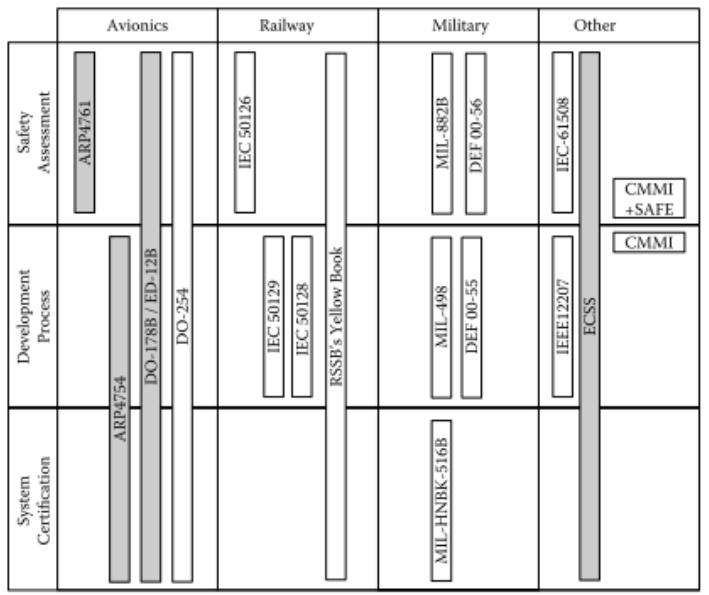
IEC 61508: Functional safety of E/E/PE safety-related systems
CENELEC EN 50126: Railway applications - the Specification and Demonstration of RAMS.(*)
CENELEC EN 50128: Railway applications - SW for Railway Control and Protection System.
CENELEC EN 50129: Railway applications - Safety related electronic systems for signaling.
FRA 49 CFR (parts 200-299) (209, 213, 234, 236): Federal Railroad Authority Regulations.
...

Transporte Aéreo – Aeronave

SAE ARP 4761: Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment.
SAE ARP 4754: Certification Considerations for Highly-Integrated or Complex Aircraft Systems.
RTCA DO-178C: SW considerations in Airborne Systems and Equipment Certification.
RTCA DO-254: Design Assurance Guidance for Airborne Electronic Hardware.
RTCA DO-297: Integrated Modular Avionics (IMA) Development
...

Transporte Aéreo – Gerenciamento de Tráfego Aéreo

ICAO Doc 4444: ATM – Procedures for Air Navigation Services.
ICAO Annex 11: Air Traffic Services.
ICAO Doc 9859: Safety Management Manual.
...



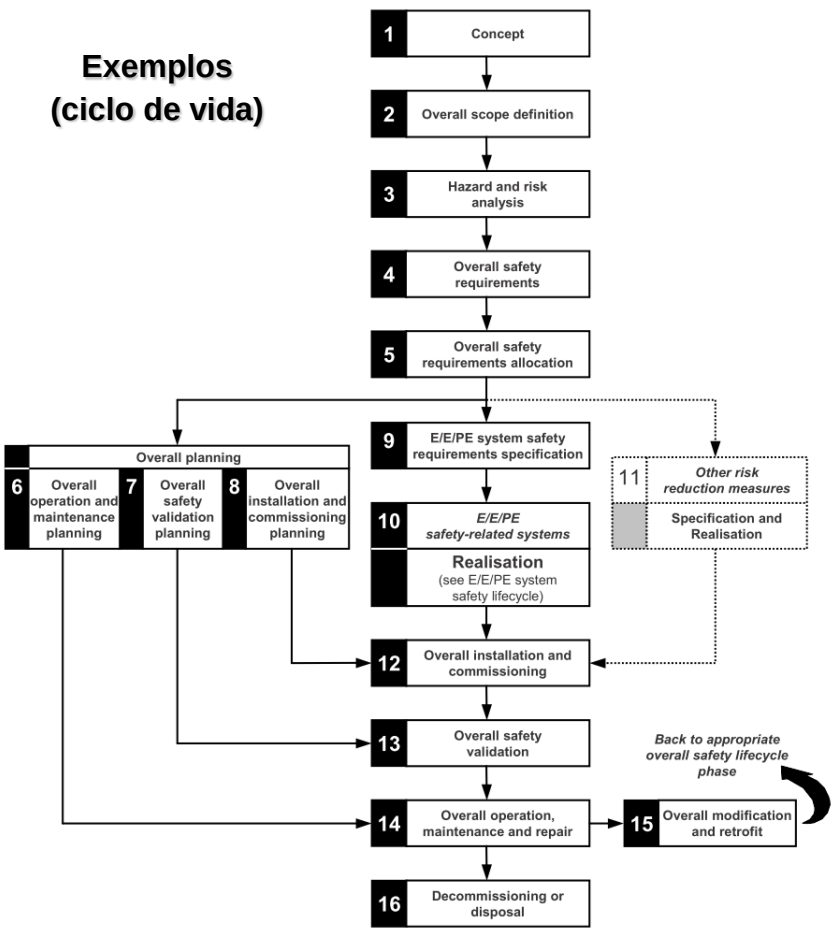
M. Bozzano and A. Villafiorita, *Design and safety assessment of critical systems*. CRC Press, 2011.

(*) CENELEC - European Committee for Electrotechnical Standardization

DESENVOLVIMENTO de Sistemas Computacionais Críticos em Dependabilidade

- ➔ **Sistemas críticos em segurança seguem ciclos de vida bem definidos.**
- ➔ **Normas lidam com o conceito de segurança em todas as fases do ciclo de vida do sistema.**

Exemplos (ciclo de vida)



IEC 61508 - Overall safety lifecycle

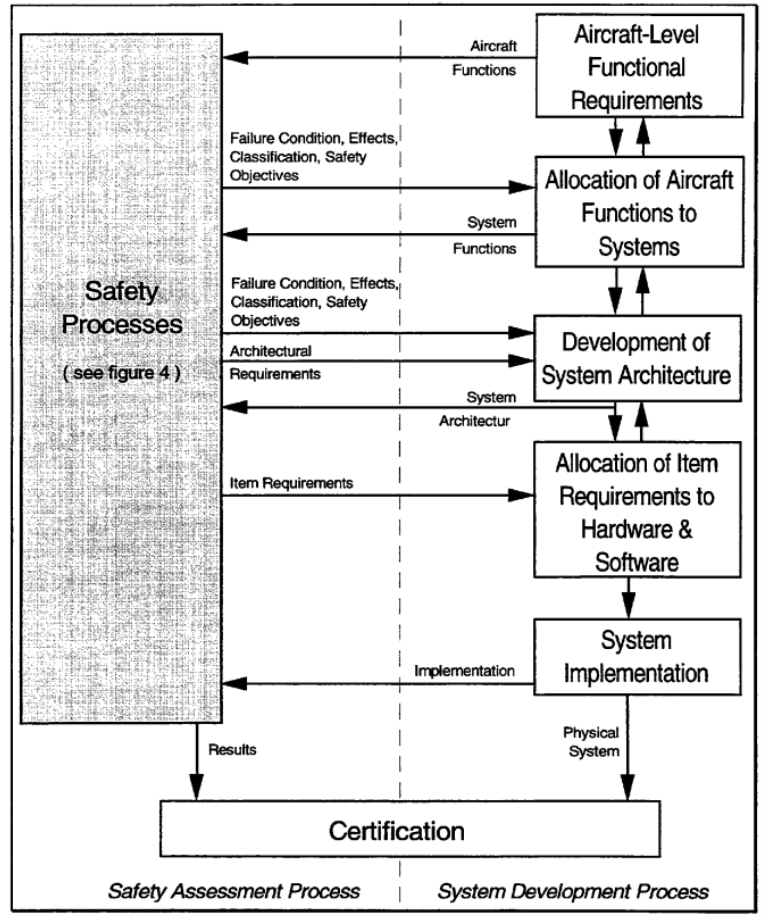


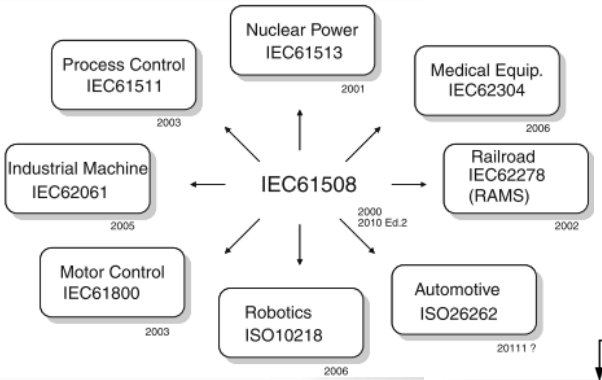
FIGURE 3 - System Development Process Model

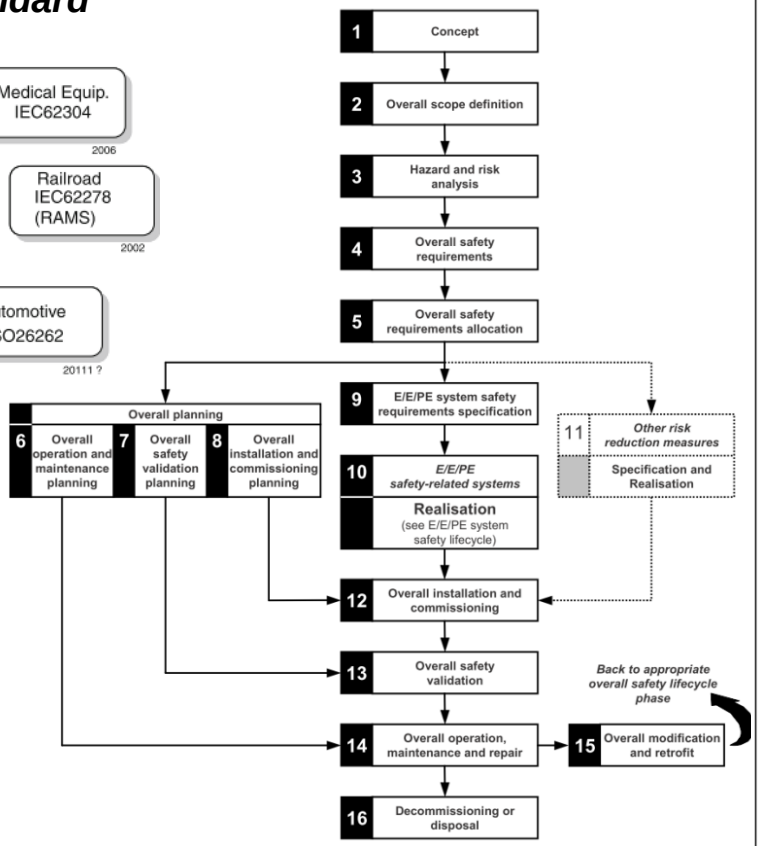
ARP 4754 - System Development Process Model

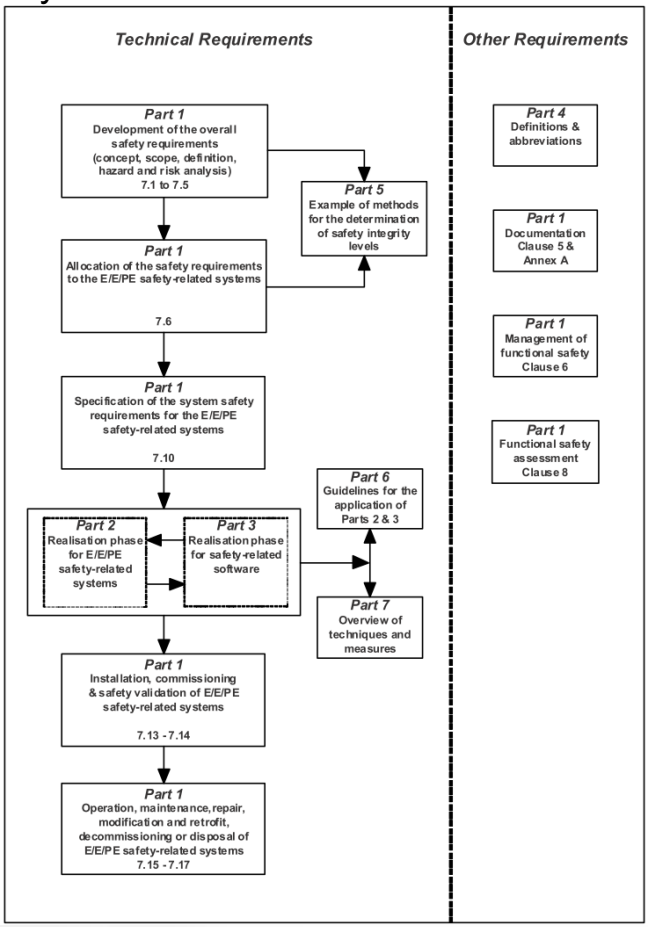
DESENVOLVIMENTO baseado na IEC61508 (2010) – E/E/PE *safety-related systems*

IEC 61508 (2010) [7 partes] "Functional Safety for E/E/PE safety-related systems"

➔ "***Basic Safety Standard***"

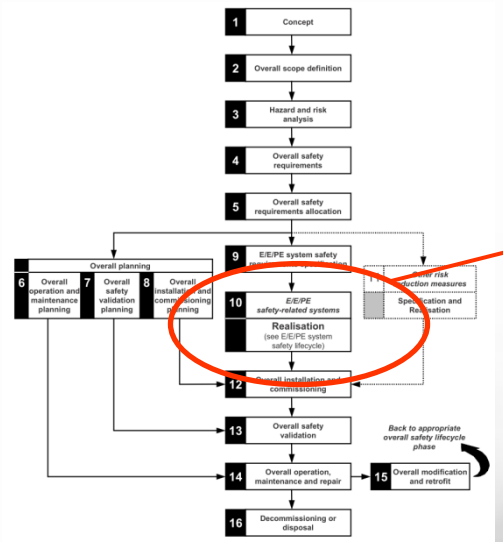


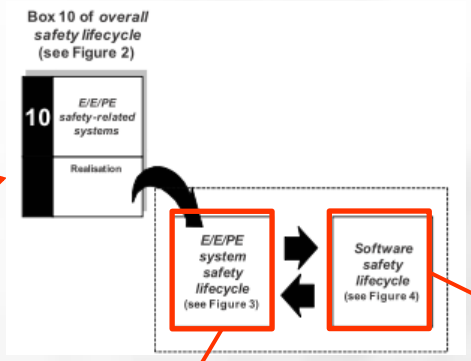


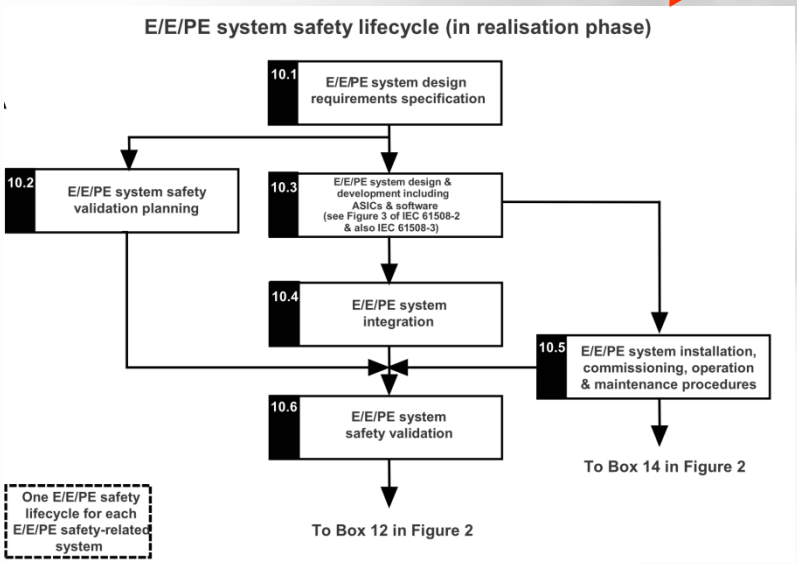


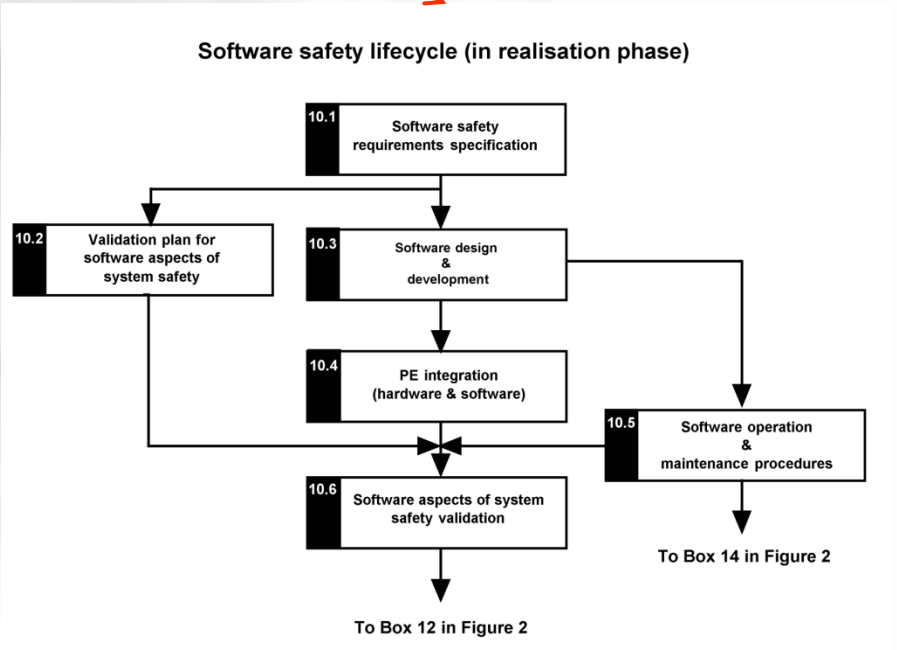
IEC 61508-1 (2010) Functional safety of electrical/electronic/programmable electronic safety-related systems – **Part 1: General requirements**
IEC 61508-2 (2010) " – **Part 2: Requirements** for E/E/PE safety-related systems
IEC 61508-3 (2010) " – **Part 3: Software requirements**
IEC 61508-4 (2010) " – **Part 4: Definitions and abbreviations**
IEC 61508-5 (2010) " – **Part 5: Examples of methods for the det. of SIL**
IEC 61508-6 (2010) " – **Part 6: Guidelines on the application of 61508-2&3**
IEC 61508-7 (2010) " – **Part 7: Overview of techniques and measures**

DESENVOLVIMENTO baseado na IEC61508 (2010) – E/E/PE *safety-related systems*



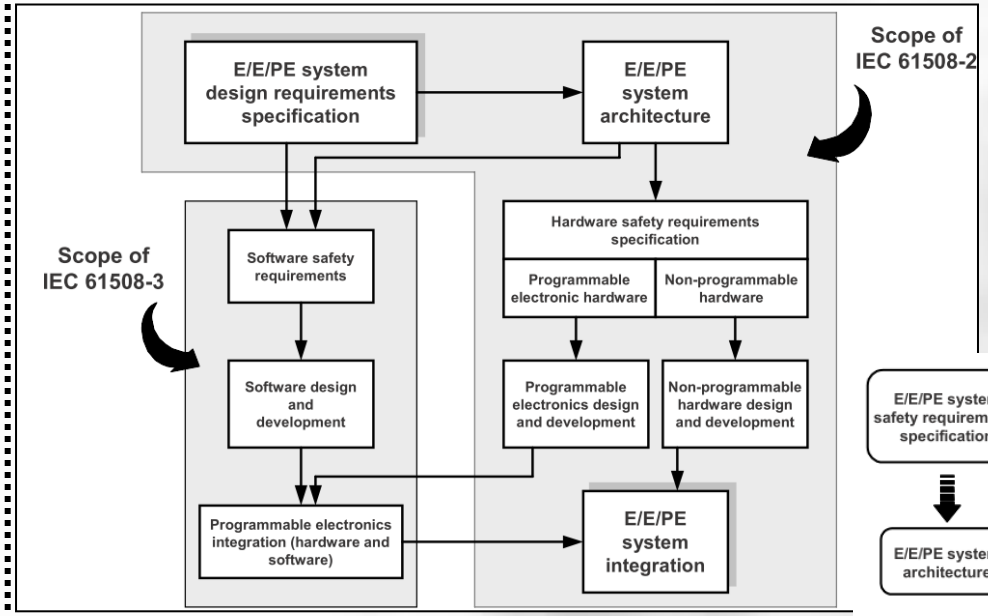


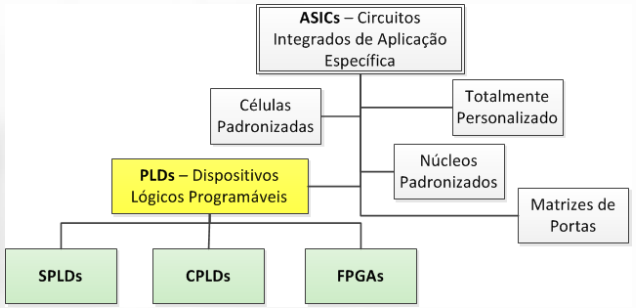


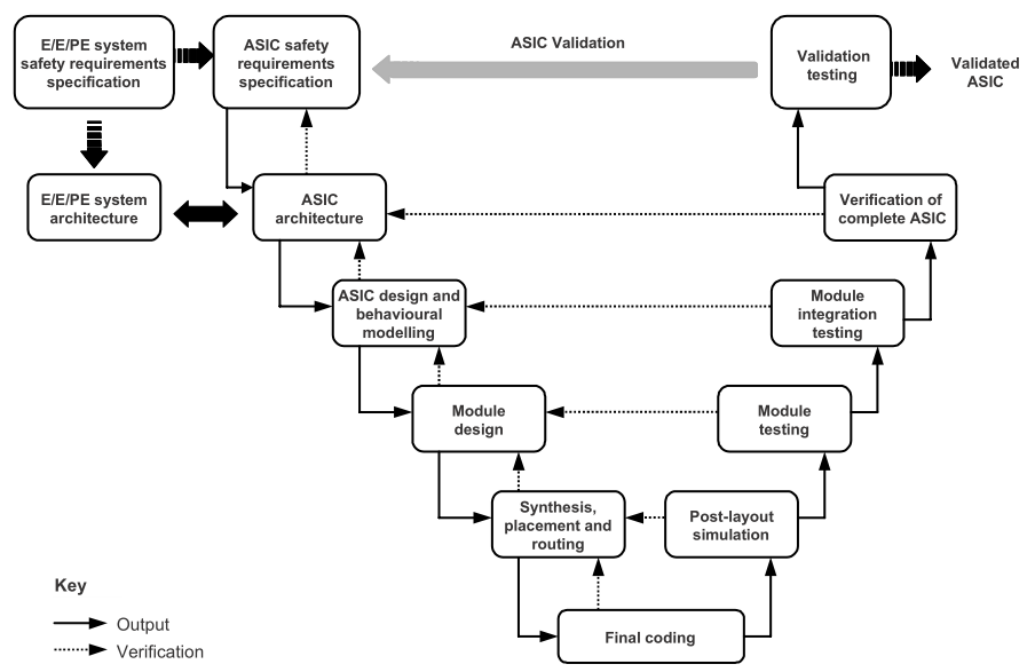


DESENVOLVIMENTO baseado na IEC61508 (2010) – E/E/PE *safety-related systems*

➔ *Hardware* (incluindo ASICs) e *Software*







IEC 61508-2 (2010) - Requirements for E/E/PE safety-related systems
IEC 61508-3 (2010) - Software requirements

Técnicas e Medidas para *FAULT AVOIDANCE/REMOVAL*

Segundo NORMA IEC 61508-2 (2010) "Functional Safety for E/E/PE safety-related systems - Part 2: Requirements for electrical/electronic/ programmable electronic safety-related systems"

- **Annex B** (normative) Techniques and measures for E/E/PE safety-related systems – **avoidance of systematic failures** during the different phases of the lifecycle.
- Annex F (informative) Techniques and measures for ASICs – **avoidance of systematic failures**.

Table B.1 – Techniques and measures to avoid mistakes during specification of E/E/PE system design requirements (see 7.2)

Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Project management	B.1.1	M low	M low	M medium	M high
Documentation	B.1.2	M low	M low	M medium	M high
Separation of E/E/PE system safety functions from non-safety functions	B.1.3	HR low	HR low	HR medium	HR high
Structured specification	B.2.1	HR low	HR low	HR medium	HR high
Inspection of the specification	B.2.6	– low	HR low	HR medium	HR high
Semi-formal methods	B.2.3, see also Table B.7 of IEC 61508-3	R low	R low	HR medium	HR high
Checklists	B.2.5	R low	R low	R medium	R high
Computer aided specification tools	B.2.4	– low	R low	R medium	R high
Formal methods	B.2.2	– low	– low	R medium	R high

Table B.3 – Techniques and measures to avoid faults during E/E/PE system integration (see 7.5)

Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Functional testing	B.5.1	M high	M high	M high	M high
Project management	B.1.1	M low	M low	M medium	M high
Documentation	B.1.2	M low	M low	M medium	M high
Black-box testing	B.5.2	R low	R low	R medium	R high
Field experience	B.5.4	R low	R low	R medium	R high
Statistical testing	B.5.3	– low	– low	R medium	R high

Table B.5 – Techniques and measures to avoid faults during E/E/PE system safety validation (see 7.7)

Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Functional testing	B.5.1	HR high	HR high	HR high	HR high
Functional testing under environmental conditions	B.6.1	HR high	HR high	HR high	HR high
Interference surge immunity testing					
Fault insertion testing (when required diagnostic coverage ≥ 90 %)					
Project management					
Documentation					
Static analysis, dynamic analysis and failure analysis					
Simulation and failure analysis					
Worst-case analysis, dynamic analysis and failure analysis					
Static analysis and failure analysis (see Note 4)					
Expanded functional testing					
Black-box testing					
Fault insertion testing (when required diagnostic coverage < 90 %)					
Statistical testing					
Worst-case testing					
Field experience					

Table B.2 – Techniques and measures to avoid introducing faults during E/E/PE system design and development (see 7.4)

Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Observance of guidelines and standards	B.3.1	M high	M high	M high	M high
Project management	B.1.1	M low	M low	M medium	M high
Documentation	B.1.2	M low	M low	M medium	M high
Structured design	B.3.2	HR low	HR low	HR medium	HR high
Modularisation	B.3.4	HR low	HR low	HR medium	HR high
Use of well-tried components	B.3.3	R low	R low	R medium	R high
Semi-formal methods	B.2.3, see also Table B.7 of IEC 61508-3	R low	R low	HR medium	HR high
Checklists	B.2.5	– low	R low	R medium	R high
Computer-aided design tools	B.3.5	– low	R low	R medium	R high
Simulation	B.3.6	– low	R low	R medium	R high
Inspection of the hardware or walk-through of the hardware	B.3.7 B.3.8	– low	R low	R medium	R high
Formal methods	B.2.2	– low	– low	R medium	R high

Técnicas e Medidas para *FAULT AVOIDANCE/REMOVAL*

Segundo NORMA IEC 61508-2 (2010) "Functional Safety for E/E/PE safety-related systems - Part 2: Requirements for electrical/electronic/ programmable electronic safety-related systems"

- Annex B (normative) Techniques and measures for E/E/PE safety-related systems – **avoidance of systematic failures** during the different phases of the lifecycle.
- **Annex F** (informative) Techniques and measures for ASICs – **avoidance of systematic failures**.

Table F.2 – Techniques and measures to avoid introducing faults during ASIC design and development: User programmable ICs (FPGA/PLD/CPLD) (see 7.4.6.7)

Design phase	Ref	Technique/Measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Design entry	1	Structured description	E.3	HR high	HR high	HR* high	HR* high
	2	Design description in (V)HDL (see Note)	E.1	HR high	HR high	HR* high	HR* high
	3	Schematic entry	E.2	– high	– high	NR	NR
	4	Design description using boolean equations		R high	R high	NR	NR
	5a	For circuit descriptions that use boolean equations: manual inspection in designs with limited (low) complexity		HR high	HR high	HR* high	HR* high
	5b	For circuit descriptions that use boolean equations: simulation of state transitions in designs with higher complexity		HR high	HR high	HR* high	HR* high
	6	Application of a proven in use design environment	E.4	HR high	HR high	HR* high	HR* high
	7	Application of proven in use (V)HDL simulators (see Note)	E.4	HR high	HR high	HR* high	HR* high
	8	Functional test on module level (using for example (V)HDL test benches) (see Note)	E.6	HR high	HR high	HR* high	HR* high
	9	Restricted use of asynchronous constructs	E.9	HR high	HR high	HR* high	HR* high
	10	Design for testability (depending on the test coverage in percent)	E.11	R > 95 %	R > 98 %	R > 99 %	R > 99 %
	11	Modularisation	E.12	R medium	R medium	HR high	HR high
	12	Coverage of the verification scenarios (test benches)	E.13	R medium	R medium	HR high	HR high
	13	Observation of coding guidelines	E.14	HR high	HR high	HR* high	HR* high
	14	Documentation of simulation results	E.17	HR low	HR medium	HR high	HR* high
	15a	Code inspection	E.18	R medium	R high	HR high	HR* high
	15b	Walk-through	E.19	R medium	R high	HR high	HR* high
	16a	Application of validated soft-cores	E.20	R medium	R high	HR high	HR* high
	16b	Validation of soft-cores	E.21	R medium	R high	HR* high	HR* high

Table F.2 *(continued)*

Design phase	Ref	Technique/Measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Synthesis	17	Internal consistency checks (see for example IEC 61508-7, E.4)		HR high	HR high	HR* high	HR* high
	18a	Simulation of the gate netlist, to check timing constraints	E.22	R medium	R medium	R high	R high
	18b	Static analysis of the propagation delay (STA)	E.23	R medium	R medium	R high	R high
	19a	Verification of the gate netlist against a reference model by simulation	E.24	R medium	R medium	HR high	HR high
	19b	Comparison of the gate netlist with the reference model (formal equivalence check)	E.25	R medium	R medium	HR high	HR high
	20	For PLD/CPLD in complex designs: check of the design by simulation		R medium	R medium	HR high	HR high
	21	Check of IC vendor requirements and constraints					
	22	Documentation of synthesis constraints results and tools					
	23	Application of proven in use synthesis tools					
	24	Application of proven in use libraries/CPLD technologies					
	25	Script based procedure					
Placement, routing, layout generation	26a	Justification of proven in use for applie hard cores					
	26b	Application of validated hard cores					
	26c	Online testing of hard cores					
	27a	Simulation of the gate netlist, to check timing constraints					
	27b	Static analysis of the propagation dela (STA)					
	28a	Verification of the gate netlist against a reference model by simulation					
	28b	Comparison of the gate netlist with the reference model (formal equivalence check)					
	29	Design rule check (DRC)					
	30	Application of a proven in use design environments, application of proven in use cell libraries	E.4	HR* high	HR* high	HR* high	HR* high
	31	Additional slack (>20 %) for process technologies in use for less than 3 years	E.39	HR high	HR high	HR* high	HR* high

Table F.2 *(continued)*

Design phase	Ref	Technique/Measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
Manu-facturing	32	Application of a proven in use process technology		HR high	HR high	HR* high	HR* high
	33	Application of proven in use device-series	E.41	HR high	HR high	HR* high	HR* high
	34	Proven in use manufacturing process	E.42	HR low	HR medium	HR high	HR* high
	35	Quality control of the manufacturing process	E.43	HR high	HR high	HR high	HR* high
	36	Manufacturing quality pass of the device	E.44	R low	R medium	HR high	HR* high
	37	Functional quality pass of the device	E.45	HR high	HR high	HR* high	HR* high
	38	Quality standards	E.46	HR low	HR medium	HR high	HR* high
	39	Quality management, for example according to ISO 9000		HR high	HR high	HR high	HR* high
	40	Final verification and validation of the FPGA/PLD prototype in the system		HR high	HR high	HR* high	HR* high
	41	Final verification and validation during mass manufacturing, per-unit-check		R high	R high	HR* high	HR* high
	42	Burn-in test	E.40	R low	R low	R medium	HR* high

Técnicas e medidas para TOLERÂNCIA A FALHAS (*Fault Tolerance*)

Segundo NORMA IEC 61508-2 (2010) "Functional Safety for E/E/PE safety-related systems - Part 2: Requirements for electrical/electronic/ programmable electronic safety-related systems"

- Annex A (normative) Techniques and measures for E/E/PE safety-related systems – **control of failures during operation**.

Table A.15 – Techniques and measures to control systematic failures caused by hardware design

	Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
	Program sequence monitoring	A.9	HR low	HR low	HR medium	HR high
	Failure detection by on-line monitoring (see Note 4)	A.1.1	R low	R low	R medium	R high
	Tests by redundant hardware	A.2.1	R low	R low	R medium	R high
	Standard test access port and boundary-scan architecture	A.2.3	R low	R low	R medium	R high
	Code protection	A.6.2	R low	R low	R medium	R high
	Diverse hardware	B.1.4	– low	– low	R medium	R high

At least one of the techniques in the light grey shaded group, or one of the techniques specified in Table A.3 of IEC 61508-3, is required.

Table A.16 – Techniques and measures to control systematic failures caused by environmental stress or influences

	Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
	Measures against voltage breakdown, voltage variations, overvoltage, low voltage and other phenomena such as a.c. power supply frequency variation that can lead to dangerous failure	A.8	M low	M medium	M medium	M high
	Separation of electrical energy lines from information lines (see Note 4)	A.11.1	M	M	M	M
	Increase of interference immunity	A.11.3	M low	M low	M medium	M high
	Measures against the physical environment (for example, temperature, humidity, water, vibration, dust, corrosive substances)	A.14	M low	M high	M high	M high
	Program sequence monitoring	A.9	HR low	HR low	HR medium	HR high
	Measures against temperature increase	A.10	HR low	HR low	HR medium	HR high
	Spatial separation of multiple lines	A.11.2	HR low	HR low	HR medium	HR high
	Idle current principle (where continuous control is not needed to achieve or maintain a safe state of the EUC)	A.1.5	R	R	R	R
	Measure to detect breaks and shorts in signal lines		R	R	R	R
	Failure detection by on-line monitoring (see Note 5)	A.1.1	R low	R low	R medium	R high
	Tests by redundant hardware	A.2.1	R low	R low	R medium	R high
	Code protection	A.6.2	R low	R low	R medium	R high
	Antivalent signal transmission	A.11.4	R low	R low	R medium	R high
	Diverse hardware (see Note 6)	B.1.4	– low	– low	– medium	R high
	Software architecture	7.4.3 of IEC 61508-3	See Tables A.2 and C.2 of IEC 61508-3			

Table A.17 – Techniques and measures to control systematic operational failures

	Technique/measure	See IEC 61508-7	SIL 1	SIL 2	SIL 3	SIL 4
	Modification protection	B.4.8	M low	M medium	M high	M high
	Failure detection by on-line monitoring (see Note 4)	A.1.1	R low	R low	R medium	R high
	Input acknowledgement	B.4.9	R low	R low	R medium	R high
	Failure assertion programming	C.3.3	See Tables A.2 and C.2 of IEC 61508-3			

At least one of the techniques in the light grey shaded group is required.

A IEC 61508 e os Safety Integrity Levels (SILs)

safety-related system
designated system that both

- implements the required safety functions necessary to achieve or maintain a safe state for the EUC; and
- is intended to achieve, on its own or with other E/E/PE safety-related systems and other risk reduction measures, the necessary safety integrity for the required safety functions

A safety-related system may
a) be designed to **prevent** the hazardous event.
b) be designed to **mitigate** the effects of the harmful event, thereby reducing the risk by reducing the consequences;
c) be designed to achieve a combination of a) and b).

safety function
function to be implemented by an E/E/PE safety-related system or other risk reduction measures, that is intended to achieve or maintain a safe state for the EUC, in respect of a specific hazardous event (see 3.4.1 and 3.4.2)

safety integrity
probability of an E/E/PE safety-related system satisfactorily performing the specified safety functions under all the stated conditions within a stated period of time

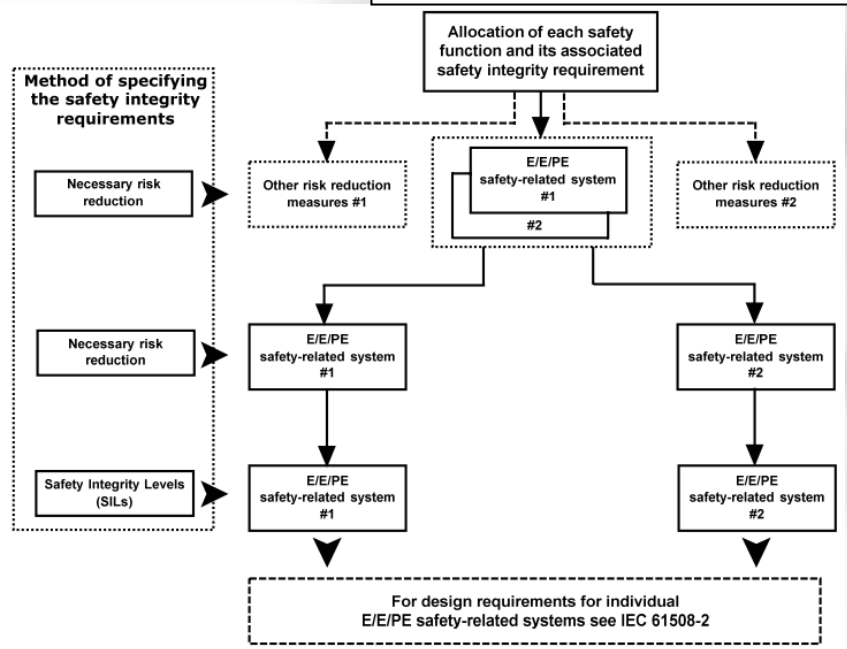


Table 3 – Safety integrity levels – target failure measures for a safety function operating in high demand mode of operation or continuous mode of operation

Safety integrity level (SIL)	Average frequency of a dangerous failure of the safety function [h^{-1}] (PFH)
4	$\geq 10^{-9}$ to $< 10^{-8}$
3	$\geq 10^{-8}$ to $< 10^{-7}$
2	$\geq 10^{-7}$ to $< 10^{-6}$
1	$\geq 10^{-6}$ to $< 10^{-5}$

Table 2 – Safety integrity levels – target failure measures for a safety function operating in low demand mode of operation

Safety integrity level (SIL)	Average probability of a dangerous failure on demand of the safety function (PFD_{avg})
4	$\geq 10^{-5}$ to $< 10^{-4}$
3	$\geq 10^{-4}$ to $< 10^{-3}$
2	$\geq 10^{-3}$ to $< 10^{-2}$
1	$\geq 10^{-2}$ to $< 10^{-1}$

PFD	Probability of Dangerous Failure on Demand
PFD_{avg}	Average Probability of dangerous Failure on Demand
PFH	Average frequency of dangerous failure [h^{-1}]

ENGENHARIA DE SEGURANÇA DE SISTEMAS

System Safety Engineering: *an engineering discipline that employs specialized knowledge and skills in applying scientific and engineering principles, criteria, and techniques to* ***identify hazards and then to eliminate the hazards or reduce the associated risks when a hazard cannot be eliminated****.*

Hazard: a real or potential condition that could lead to an unplanned event or series of events (i.e. mishap) resulting in death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment.

Hazard
potential source of harm

harm
physical injury or damage to the health of people or damage to property or the environment

Safety. Freedom from conditions that can cause death, injury, occupational illness, damage to or loss of equipment or property, or damage to the environment.

Safety. Feedom from unacceptable risk

risk (risk)
combination of the probability of occurrence of harm (mishap) and the severity of that harm (mishap)

SYSTEM SAFETY: the application of engineering and management principles, criteria, and techniques to **achieve acceptable risk** within the constraints of operational effectiveness and suitability, time, and cost, throughout all phases of the system life cycle.

[MIL-STD-882E "System Safety"]
[IEC 61508 (2010)]

AGENDA

1. O Grupo de Análise de Segurança da Poli-USP
2. Sistemas Computacionais em Aplicações Críticas
3. **Programação Defensiva (PD)**

... e explorar **técnicas de programação defensiva** aplicáveis no desenvolvimento de **sistemas computacionais tolerantes a falhas**."

4. Exemplos de aplicação de PD em sistemas críticos
5. Comentários Finais e Discussão

BOTTOM-UP APPROACH

GERENCIAMENTO DE FALHAS
(abordagem da Dependabilidade):

`<Fault>` ➔ `<Failure>` ➔ `<Consequence = Catastrophic>`

- **EVITAR introdução de <u>falhas</u> durante desenvolvimento;**
- **REMOVER as <u>falhas</u> introduzidas no sistema (antes de suas manifestações);**
- **TOLERAR a ocorrência de <u>falhas</u> durante operação.**

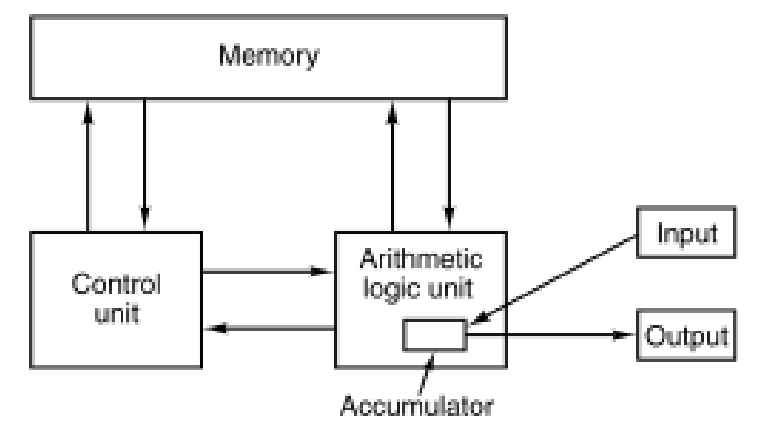
TOP-DOWN APPROACH

GERENCIAMENTO DE RISCOS
(abordagem da Engenharia de Segurança de Sistemas):

`Risco = Prob[↑Hazard] x Consequence[Hazard]`

- **ELIMINAR (evitar ou remover) os *Hazards*** (`Prob[↑Hazard]=0,0...`)
- **REDUZIR SEVERIDADE (tolerar) os *Hazards*** (`min{Consequence[Hazard]|↑Hazard}`)

Sistemas baseados em Computadores:



software [IEC61508-4(2010)]
*"intellectual creation comprising the **<u>programs</u>**, **procedures**, **data**, **rules** and any associated documentation pertaining to the operation of a **data processing system**"*

3. PROGRAMAÇÃO DEFENSIVA

OBJETIVO: produzir programas que **DETECTEM** anomalias nos **FLUXOS DE CONTROLE**, **FLUXOS DE DADOS** ou **VALORES DE DADOS** durante sua execução e **REAJAM** **de maneira predeterminada e aceitável**." [IEC 61508-7 (2010)]

Nome genérico atribuído às diversas técnicas de projeto (PROGRAMAÇÃO) do SOFTWARE que podem ser implementadas em um sistema programado (baseado em computador) visando incrementar o seu nível integridade de segurança (SIL).

Duas áreas sobrepostas de técnicas defensivas podem ser identificadas:

- **Prevenção contra requisitos errôneos e/ou incompletos;**

  Ex.: falha na especificação de faixas de valores de entradas; definição incorreta dos tipos de parâmetros funcionais, entre outros.

- **Prevenção contra deficiências oriundas de erros de projeto ou na codificação.**

  Ex.: nível de robustez de HW incompatível com o ambiente de operação, uso de compiladores com problemas, entre outros.

3. PROGRAMAÇÃO DEFENSIVA

OBJETIVO: produzir programas que **DETECTEM** anomalias nos **FLUXOS DE CONTROLE**, **FLUXOS DE DADOS** ou **VALORES DE DADOS** durante sua execução **e REAJAM de maneira predeterminada e aceitável**." [IEC 61508-7 (2010)]

CONCEITUALMENTE, as técnicas de Programação Defensiva podem ser classificadas em 2 grupos:

ESTÁTICAS:

- **Utilizadas em tempo de programação e compilação.**
- **Padronização na proteção contra erros ("*Coding Standards*").**
- **Uso das convenções dos compiladores como programação defensiva.**

DINÂMICAS (ou de *runtime*)

- **presentes em tempo de execução do software (sistema em operação).**
- **Certificações em Tempo de Execução (*garante a funcionalidade com base em determinados pontos do programa*).**

Detecção de Anomalias em Sistemas Críticos utilizando PD

Dado que os programas, segundo a definição de PD, **devem detectar anomalias** em **Fluxos de Controle**, **Fluxos de Dados** ou **Valores de Dados** durante sua execução **e** reagir de maneira predeterminada e aceitável.

⇩

Exemplos de Técnicas para Detecção de Falhas

Functionality Check ➔ verificação de funcionalidade dos módulos de Hardware do Sistema (por meio de testes de leitura e escrita de memória, de registradores, entre outros).

Consistency Checking ➔ verifica consistência de informação a ser processada (ex.: *range checking, teste de passagem de parâmetros*) ➔ "Preconditions" e "Postconditions".

Signal Comparison ➔ em sistemas com redundância, valida informação comparando múltiplas fontes.

Information Redundancy ➔ campos redundantes (paridade, M-out-of-N codes, checksum, CRC), códigos corretores de erro, RAID, Replicação de Dados (variáveis duais)

Loopback Testing ➔ teste de informações escritas por meio de sua leitura (loopback) e comparação.

Watchdog Timer ➔ detecta inoperância de processamento por meio de mecanismos de "sw refresh".

Bus and Power Supply Monitoring ➔ monitoramento da condição funcional de barramentos e fontes de energia, verificando se estão operando segundo valores nominais.

Testes de Início e Fim➔ testa execução completa de rotinas/módulos (evita reentrâncias).

Detecção de Anomalias em Sistemas Críticos utilizando PD

Softwares Tolerantes a Falhas são projetados tanto para "aguardar o surgimento de falhas dentro de seu próprio ambiente" quanto para "utilizar condições externas (nominais ou esperadas), e se comportar de uma maneira pré-definida".

Algumas das técnicas que podem ser utilizadas para estes fins são:

Verificação de Variáveis de Entrada e Variáveis com significado físico ➔ valores verificados quanto à plausibilidade de ocorrência.

Verificação de Variáveis de Saída ➔ seus efeitos devem ser verificados, preferencialmente, pela observação direta de mudanças associadas aos estados do sistema.

Configuração do Programa ➔ o programa deve "auto-verificar" sua configuração. Isto pode incluir tanto a existência e a acessibilidade de um hardware esperado como a verificação de que ele mesmo está completo (importante na garantia de integridade após manutenções).

3. PROGRAMAÇÃO DEFENSIVA

PD NA NORMA IEC 61508-3 (2010)

Annex A (normative) - Guide to the selection of techniques and measures

Table A.4 – Software design and development – detailed design

(See 7.4.5 and 7.4.6)

(Includes software system design, software module design an

	Technique/Measure *	Ref.	SIL 1	SI
1a	Structured methods **	C.2.1	HR	H
1b	Semi-formal methods **	Table B.7	R	H
1c	Formal design and refinement methods **	B.2.2, C.2.4	---	
2	Computer-aided design tools	B.3.5	R	
3	Defensive programming	C.2.5	---	
4	Modular approach	Table B.9	HR	H
5	Design and coding standards	C.2.6 Table B.1	R	H
6	Structured programming	C.2.7	HR	H
7	Use of trusted/verified software elements (if available)	C.2.10	R	H
8	Forward traceability between the software safety requirements specification and software design	C.2.11	R	

Table B.1 – Design and coding standards

(Referenced by Table A.4)

	Technique/Measure *	Ref.	SIL 1	SIL 2	SIL 3	
1	Use of coding standard to reduce likelihood of errors	C.2.6.2	HR	HR	HR	
2	No dynamic objects	C.2.6.3	R	HR	HR	
3a	No dynamic variables	C.2.6.3	---	R	HR	
3b	Online checking of the installation of dynamic variables	C.2.6.4	---	R	HR	
4	Limited use of interrupts	C.2.6.5	R	R	HR	
5	Limited use of pointers	C.2.6.6	---	R	HR	
6	Limited use of recursion	C.2.6.7	---	R	HR	
7	No unstructured control flow in programs in higher level languages	C.2.6.2	R	HR	HR	HR
8	No automatic type conversion	C.2.6.2	R	HR	HR	HR

Table A.2 – Software design and development – software architecture design

(see 7.4.3)

	Technique/Measure *	Ref.	SIL 1	SIL 2	SIL 3	SIL 4
	Architecture and design feature					
1	Fault detection	C.3.1	---	R	HR	HR
2	Error detecting codes	C.3.2	R	R	R	HR
3a	Failure assertion programming	C.3.3	R	R	R	HR
3b	Diverse monitor techniques (with independence between the monitor and the monitored function in the same computer)	C.3.4	---	R	R	----
3c	Diverse monitor techniques (with separation between the monitor computer and the monitored computer)	C.3.4	---	R	R	HR
3d	Diverse redundancy, implementing the same software safety requirements specification	C.3.5	---	---	---	R
3e	Functionally diverse redundancy, implementing different software safety requirements specification	C.3.5	---	---	R	HR
3f	Backward recovery	C.3.6	R	R	---	NR
3g	Stateless software design (or limited state design)	C.2.12	---	---	R	HR
4a	Re-try fault recovery mechanisms	C.3.7	R	R	---	---
4b	Graceful degradation	C.3.8	R	R	HR	HR
5	Artificial intelligence - fault correction	C.3.9	---	NR	NR	NR
6	Dynamic reconfiguration	C.3.10	---	NR	NR	NR
7	Modular approach	Table B.9	HR	HR	HR	HR
8	Use of trusted/verified software elements (if available)	C.2.10	R	HR	HR	HR
9	Forward traceability between the software safety requirements specification and software architecture	C.2.11	R	R	HR	HR
10	Backward traceability between the software safety requirements specification and software architecture	C.2.11	R	R	HR	HR
11a	Structured diagrammatic methods **	C.2.1	HR	HR	HR	HR
11b	Semi-formal methods **	Table B.7	R	R	HR	HR
11c	Formal design and refinement methods **	B.2.2, C.2.4	---	R	R	HR
11d	Automatic software generation	C.4.6	R	R	R	R
12	Computer-aided specification and design tools	B.2.4	R	R	HR	HR
13a	Cyclic behaviour, with guaranteed maximum cycle time	C.3.11	R	HR	HR	HR
13b	Time-triggered architecture	C.3.11	R	HR	HR	HR
13c	Event-driven, with guaranteed maximum response time	C.3.11	R	HR	HR	-
14	Static resource allocation	C.2.6.3	-	R	HR	HR
15	Static synchronisation of access to shared resources	C.2.6.3	-	-	R	HR

PD NA NORMA IEC EN50128 (2001)

Railway applications - Communications, signalling and processing systems - Software for railway control and protection systems

Table A.3 – Software Architecture (clause 9)

TECHNIQUE/MEASURE	Ref	SWS ILO	SWS IL1	SWS IL2	SWS IL3	SWS IL4
1. Defensive Programming	B.15	-	R	R	HR	HR
2. Fault Detection & Diagnosis	B.27	-	R	R	HR	HR
3. Error Correcting Codes	B.20	-	-	-	-	-
4. Error Detecting Codes	B.20	-	R	R	HR	HR
5. Failure Assertion Programming	B.25	-	R	R	HR	HR
6. Safety Bag Techniques	B.54	-	R	R	R	R
7. Diverse Programming	B.17	-	R	R	HR	HR
8. Recovery Block	B.50	-	R	R	R	R
9. Backward Recovery	B.5	-	NR	NR	NR	NR
10. Forward Recovery	B.32	-	NR	NR	NR	NR
11. Re-try Fault Recovery Mechanisms	B.53	-	R	R	R	R
12. Memorising Executed Cases	B.39	-	R	R	HR	HR
13. Artificial Intelligence - Fault Correction	B.1	-	NR	NR	NR	NR
14. Dynamic Reconfiguration of software	B.18	-	NR	NR	NR	NR
15. Software Error Effect Analysis	B.26	-	R	R	HR	HR
16. Fault Tree Analysis	B.28	R	R	R	HR	HR

Requirements

1. Approved combinations of techniques for Software Safety Integrity Levels 3 and 4 shall be as follows:
   a) 1, 7 and one from 4, 5 or 12
   b) 1, 4 and 5
   c) 1, 4 and 12
   d) 1, 2 and 4
   e) 1 and 4, and one of 15 and 16

Table A.4 – Software Design and Implementation (clause 10)

TECHNIQUE/MEASURE	Ref	SWS ILO	SWS IL1	SWS IL2	SWS IL3	S
1. Formal Methods including for example CCS, CSP, HOL, LOTOS, OBJ, Temporal Logic, VDM, Z and B	B.30	-	R	R	HR	
2. Semi-Formal Methods	D.7	R	HR	HR	HR	
3. Structured. Methodology including for example JSD, MASCOT, SADT, SDL, SSADM and Yourdon.	B.60	R	HR	HR	HR	
4. Modular Approach	D.9	HR	M	M	M	
5. Design and Coding Standards	D.1	HR	HR	HR	M	
6. Analysable Programs	B.2	HR	HR	HR	HR	
7. Strongly Typed Programming Language	B.57	R	HR	HR	HR	
8. Structured Programming	B.61	R	HR	HR	HR	
9. Programming Language	D.4	R	HR	HR	HR	
10. Language Subset	B.38	-	-	-	HR	
11. Validated Translator	B.7	R	HR	HR	HR	
12. Translator Proven in Use	B.65	HR	HR	HR	HR	
13. Library of Trusted/Verified Modules and Components	B.40	R	R	R	R	

AGENDA

1. O Grupo de Análise de Segurança da Poli-USP
2. Sistemas Computacionais em Aplicações Críticas
3. Técnicas de Programação Defensiva (PD)
4. **Exemplos de PD em sistemas críticos**
5. Comentários Finais e Discussão

4. EXEMPLOS DE PD EM SISTEMAS CRÍTICOS

Original code	*Modified Code*

`a = b;`	`a0 = b0;`
	`a1 = b1;`
	`if (b0 != b1)`
	`error();`

`a = b + c;`	`a0 = b0 + c0;`
	`a1 = b1 + c1;`
	`if ((b0!=b1) || (c0!=c1))`
	`error();`

PD aplicada a erros que afetem os dados

Original code	*Modified code*

`res = search (a);`
…
`int search (int p)`
`{ int q;`
…
`q = p + 1;`
…
`return(1);`
`}`
	`search(a0, a1, &res0, &res1);`
	…
	`void search (int p0,int p1,int *r0,int *r1)`
	`{ int q0, q1;`
	…
	`q0 = p0 + 1;`
	`q1 = p1 + 1;`
	`if (p0 != p1)`
	`error();`
	…
	`*r0 = 1;`
	`*r1 = 1;`
	`return;`
	`}`

PD aplicada a erros que afetem passagem de parâmetros de funções

Rebaudengo et al (1999). Soft-error Detection through Software Fault-Tolerance techniques. Int. Symp. On Defect and Fault Tolerance in VLSI Systems, 1999. DFT'99

4. EXEMPLOS DE PD EM SISTEMAS CRÍTICOS

Original code	*Modified Code*

`if (condition)`
`{/* Block A */`
`  …`
`}`
`else`
`{/* Block B */`
`  …`
`}`
	`if (condition)`
	`{/* Block A */`
	`  if(!condition)`
	`  error();`
	`  …`
	`}`
	`else`
	`{/* Block B */`
	`  if(condition)`
	`  error();`
	`  …`
	`}`

PD aplicada a testes de declaração

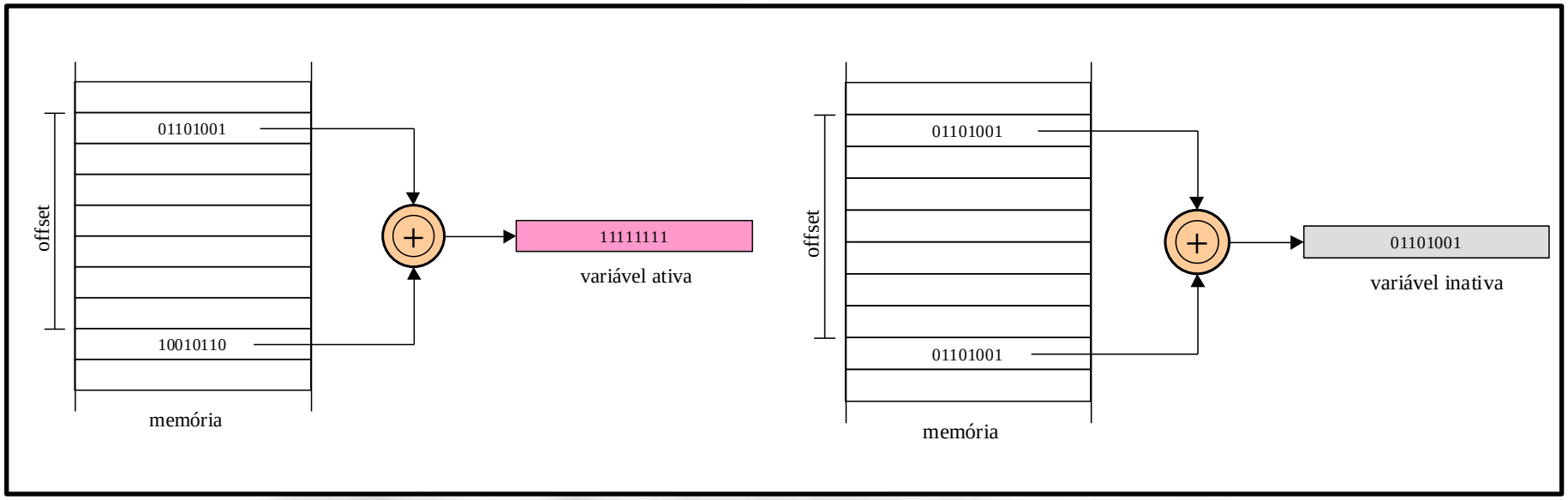
Original code	*Modified Code*

`…`
`ret = my_proc(a);`
`/* procedure call */`
`…`
`/* procedure definition */`
`int my_proc(int a)`
`{`
`  /* procedure body */`
`  …`
`  return(0);`
`}`
	`…`
	`/*call of procedure #790 */`
	`ret = my_proc( a);`
	`if( ecf != 790)`
	`  error();`
	`…`
	`/* procedure definition */`
	`int my_proc(int a)`
	`{`
	`  /* procedure body */`
	`  …`
	`  ecf = 790;`
	`  return (0);`
	`}`

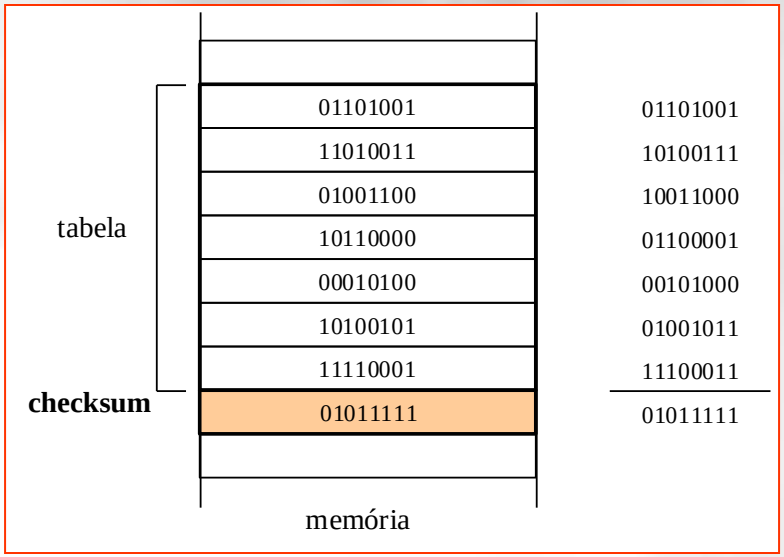
PD aplicada a procedimento de chamada e retorno de declaração

Rebaudengo et al (1999). Soft-error Detection through Software Fault-Tolerance techniques. Int. Symp. On Defect and Fault Tolerance in VLSI Systems, 1999. DFT'99

4. EXEMPLOS DE PD EM SISTEMAS CRÍTICOS



Uso de variáveis duais (redundância espacial)



Uso de campos redundantes (Checksum)

4. EXEMPLOS DE PD EM SISTEMAS CRÍTICOS

```
/* variável de controle global */
int InicioFIm_RotinaX;

void RotinaX(WORD smsg, WORD omsg, WORD teste)
{
                /* Teste de Inicio-Fim: Inicio. */
                if ((InicioFim_RotinaX & I_ROTINAX) != 0)
                {
                  Erro();
                }
                InicioFim_RotinaX = InicioFimRotinaX |  I_ROTINAX;

                /* Teste de parâmetros. */
                if ((smsg ^ omsg) != teste)
                {
                     Erro();
                }

                            <PROGRAMA>

                /* Teste de Inicio-Fim: Fim. */
                if ((InicioFim_RotinaX & I_ROTINAX) == 0)
                {
                     Erro();
                }
                InicioFim_RotinaX = InicioFim_RotinaX  & (~I_ROTINAX);

                return;
}
```

PD aplicada a testes de Início/Fim e Passagem de Parâmetro

4. EXEMPLOS DE PD EM SISTEMAS CRÍTICOS

J.M. Secall (2007). Avaliação comparativa do impacto do emprego de técnicas de programação defensiva na segurança de sistemas críticos. Tese Doutorado. EPUSP.
http://www.teses.usp.br/teses/disponiveis/3/3141/tde-06072007-154142/

1. **Duplicação de variáveis**: cada variável é guardada em duas posições de memória que são comparadas antes do uso. Para cada variável serão efetuadas as mesmas operações em tempo de execução e a verificação se dará por consistência

2. **Polinômio de recuperação tipo BCH** (Bose, Ray-Chaudhuri, Hocquenghem): cada variável é guardada com informação de recuperação de corrupção através de código BCH;

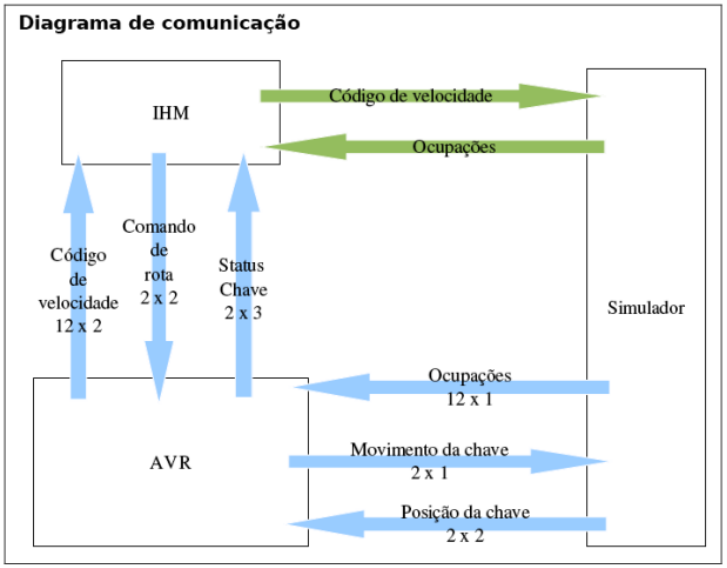
3. **Block Recovery, tipo Forward Recovery:** um estado de segurança para todas as variáveis do sistema está programado no programa e, na detecção de uma falha, este estado é restabelecido e utilizado como novo start-up para o sistema;

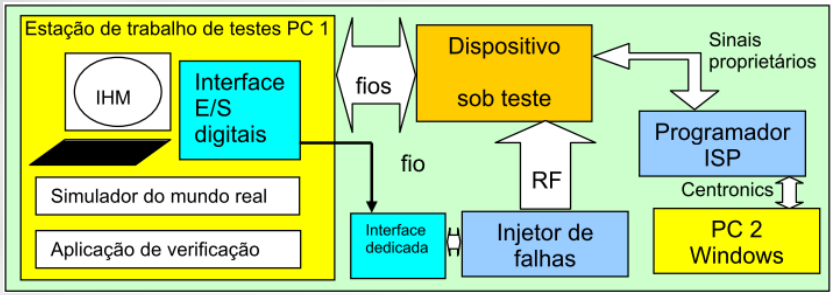
4. **Block Recovery, tipo Backward ou Back Recovery**: a cada intervalo, um estado seguro para todas as variáveis do sistema é armazenado e, na detecção de uma falha, este estado é restabelecido e utilizado como novo start-up para o sistema;

5. **Redundância Temporal**: cada linha de código é executada duas vezes e o resultado comparado;

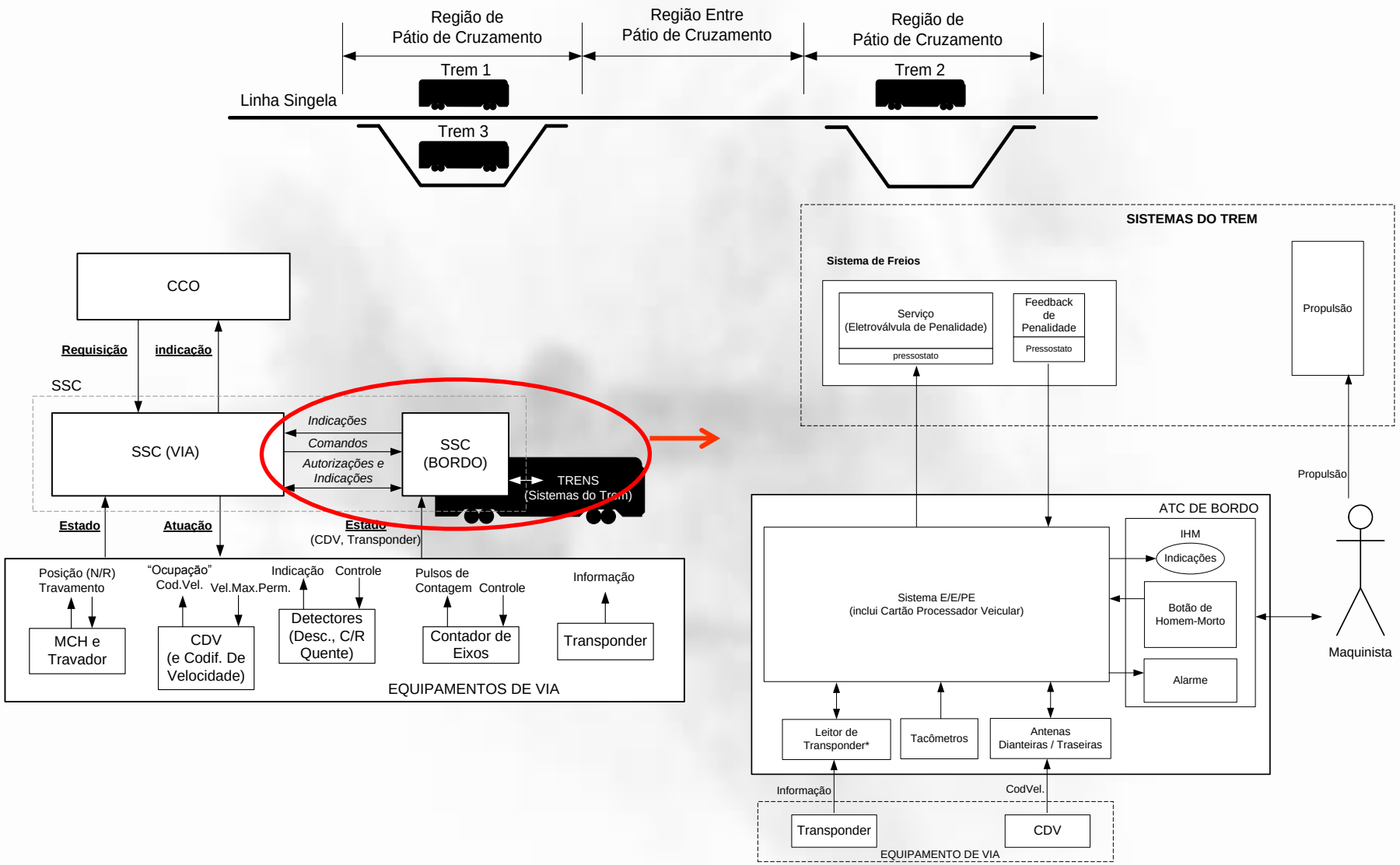
6. **Checksum (soma binária)** da área de dados: a cada operação de escrita o check sum da área de dados é recalculado e armazenado. A cada operação de leitura o check sum é antes calculado e comparado com o valor armazenado, validando o dado armazenado;

7. **Paridade:** cada variável é guardada com informação de recuperação de corrupção através de código de paridade;



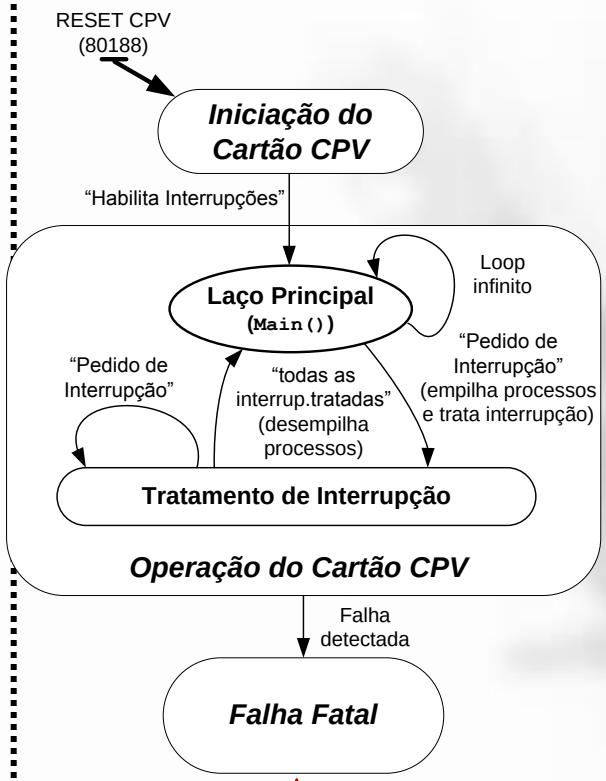


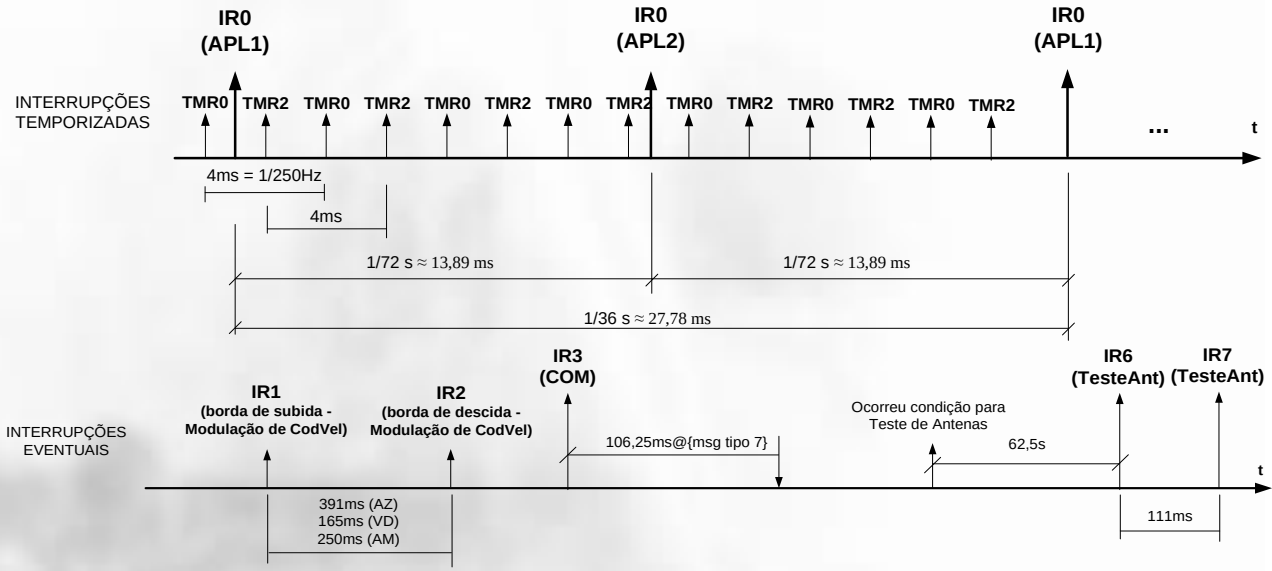
Sistema de Sinalização Ferroviário (ATC)

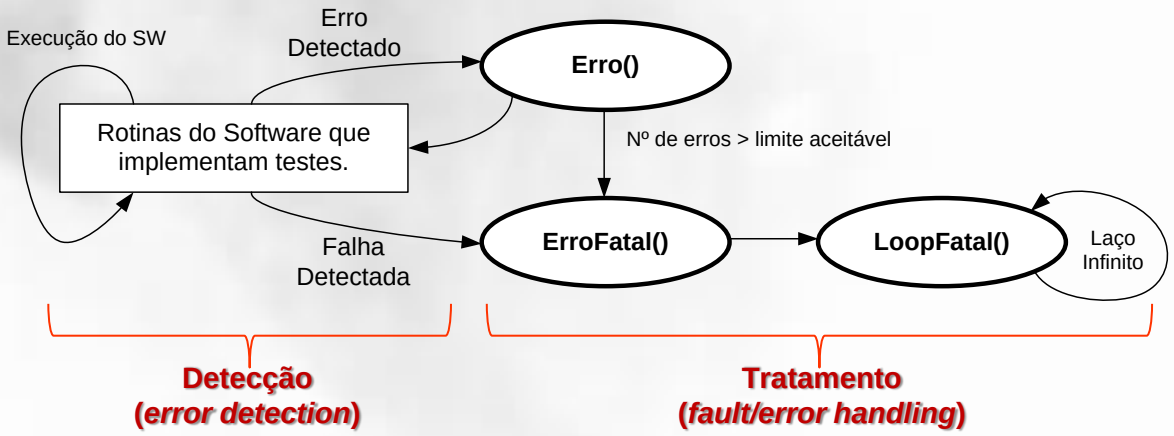


Sistema de Sinalização Ferroviário (ATC)

➔ SW do CPV







ESTADO SEGURO (Fail-safe)

Detecção (*error detection*)

Tratamento (*fault/error handling*)

Sistema de Sinalização Ferroviário (ATC)

➔ SW do CPV: técnicas de PD empregadas

VARIÁVEIS REDUNDANTES (ou "Duplicadas")

Variáveis de programa armazenadas na memória de modo redundante. **Periodicamente**, o valor de uma dessas posições é comparado com o da outra, e, caso haja discordância, é acionado o tratamento de falha.

CAMPOS REDUNDANTES

Valores de CRC são calculados em função dos valores armazenados em áreas de **memória EPROM** (um campo de redundância por página de 1kbyte), **vetores de interrupção** (na memória RAM), **parâmetros do programa** (na memória FLASH), e **mensagens de comunicação** com outros módulos (na memória compartilhada). Sempre que um desses valores é alterado, o valor de redundância da respectiva área é recalculado. **Periodicamente**, o valor da redundância é verificado, e, caso seja detectada discordância entre o valor calculado e o armazenado na memória, é acionado o tratamento de falha.

TESTES DE ENTRADA E DE SAÍDA DE UMA ROTINA

Em rotinas do programa existem variáveis que indicam se a respectiva rotina **está sendo executada**, ou se a sua **última execução já foi concluída**. A cada execução de rotina, tanto na entrada quanto na saída, sua respectiva variável é verificada, a fim de garantir a execução integral da rotina, e impedir sua reentrância. Caso alguma incoerência seja detectada, é acionado o tratamento de falha.

TESTE DA PASSAGEM DE PARÂMETROS

Rotinas recebem, das rotinas que as chamam, parâmetros passados através da pilha do programa. Adiciona-se um parâmetro extra, gerado pela rotina acionadora, em função dos valores dos demais parâmetros. A rotina que recebe esses parâmetros, cada vez que é acionada, verifica o valor do parâmetro de redundância, e, caso detecte uma eventual inconsistência na passagem de parâmetros através da pilha, é acionado o tratamento de falha.

TEMPORIZAÇÃO ENTRE EXECUÇÕES DE UMA ROTINA

Associadas a rotinas do programa, existem variáveis contadoras (os chamados **temporizadores críticos**) que indicam o tempo decorrido desde que cada uma dessas rotinas foi executada pela última vez. A cada nova execução de uma rotina, seu respectivo contador é reiniciado. **Periodicamente**, esses contadores são verificados, e, caso algum deles tenha atingido um valor máximo, previamente definido, indicando a não execução da respectiva rotina por um intervalo de tempo maior do que o esperado, é acionado o tratamento de falha.

PREENCHIMENTO DE POSIÇÕES DAS MEMÓRIAS

Todas as posições da memória EPROM não utilizadas para armazenar o código do programa (instruções e tabelas) são preenchidas com o valor 0CCH. Caso ocorra um desvio indevido do programa para uma dessas posições, é solicitada a interrupção 3, cujo atendimento aciona o tratamento de erro.

Já as posições da memória RAM não utilizadas são preenchidas com o valor 67H (macro UNOPCODE). Analogamente, caso ocorra um desvio indevido do programa para uma dessas posições, é solicitada a interrupção 6, cujo atendimento também aciona o tratamento de erro.

➔ Outras técnicas de TF aplicadas:

Watch-Dog Timer (WDT)
Módulo de Diagnóstico Independente por Hardware [BIT]
Redundância Temporal + Diversidade de Módulos de Sw

AGENDA

1. O Grupo de Análise de Segurança da Poli-USP
2. Sistemas Computacionais em Aplicações Críticas
3. Técnicas de Programação Defensiva (PD)
4. Exemplos de aplicação de PD em sistemas críticos
5. **Comentários Finais e Discussão**

5. COMENTÁRIOS FINAIS E DISCUSSÃO

SISTEMAS CRÍTICOS e DEPENDABILIDADE

(*SAFETY, CONFIABILIDADE, DISPONIBILIDADE, ...*):

➔ SISTEMAS COM ATRIBUTOS (E REQUISITOS) ESPECIAI$ (Perda$ e Dano$).

➔ A GARANTIA DA DEPENDABILIDADE EXIGE UMA **VISÃO HOLÍSTICA**, QUE VAI ALÉM DAS FRONTEIRAS DO SISTEMA EM DESENVOLVIMENTO OU MESMO DA RELAÇÃO SISTEMA-APLICAÇÃO ("*SISTEMAS SOCIOTÉCNICOS*").

➔ A GARANTIA DA DEPENDABILIDADE DEMANDA (inclusive) O **GERENCIAMENTO DE FALHAS** AO LONGO DE **TODO SEU CICLO-DE-VIDA**. FUNDAMENTAL o "E se ...?!?" E O "*Vai que ...?!*"

➔ (tende a ser fortemente) **REGULADA E NORMATIZADA** pelo domínio de aplicação.

"NÃO EXISTE ALMOÇO GRÁTIS"

➔ GANHO EM DEPENDABILIDADE versus PERDA EM **$$ (DESENVOLVIMENTO)**, **DESEMPENHO**, "***TIME-TO-MARKET***", ...

?!?!? *"POUCA" DEPENDABILIDADE == "MUITA" DEPENDABILIDADE == PERDAS E DANOS* ?!?!

- "POUCA"? "MUITA"? **MENSURAR** ATRIBUTOS (SOBRETUDO SEGURANÇA) É UMA **'ATIVIDADE COMPLEXA'**

- **IMPACTO** SOCIAL/POLÍTICO/REGULATÓRIO **ECONOMICAMENTE INCOMENSURÁVEL (ex.: caso "UK railway")**

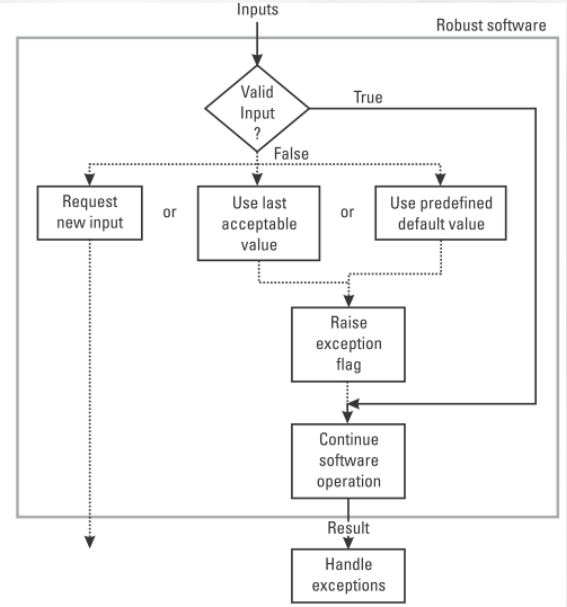
5. COMENTÁRIOS FINAIS E DISCUSSÃO

PROGRAMAÇÃO DEFENSIVA:

➔ UMA DAS ABORDAGENS (↓$?) NA CULTURA DE GERENCIAMENTO DE FALHAS.

➔ **VAI ALÉM** DO ATO DE 'MUNICIAR' O SOFTWARE DURANTE SUA **CODIFICAÇÃO**. DEVE SER PENSADO DESDE O INÍCIO DO DESENVOLVIMENTO, NA DEFINIÇÃO DA ARQUITETURA DO SISTEMA".

➔ **ENROBUSTECER** / AUMENTAR IMUNIDADE DO SISTEMA (SIL).



Software Fault Tolerance Techniques and Implementation.
Laura L. Pullum. 2001

DÚVIDAS? COMENTÁRIOS?

João Batista Camargo Junior
Professor Associado (Poli-USP)
joaocamargo@usp.br

Lúcio F. Vismari
Pesquisador, B.Eng., M.Eng. (GAS/Poli-USP)
lucio.vismari@usp.br
Engº de Segurança de Sistemas, Sr. (FDTE)
lucio.vismari@fdte.org.br

Grupo de Análise de Segurança (GAS)
Departamento de Engenharia da Computação e Sistemas Digitais (PCS)
Escola Politécnica da Universidade de São Paulo (Poli-USP)
São Paulo - Brasil