Pós-Graduação em Ciência da Computação

André Luís Ribeiro Didier

[TBD]

Ph.D. Thesis

Federal University of Pernambuco
posgraduacao@cin.ufpe.br
www.cin.ufpe.br/~posgraduacao

Recife

December 2014

Federal University of Pernambuco
Center of Informatics
Graduate in Computer Science

André Luís Ribeiro Didier

[TBD]

A Ph.D. Thesis presented to the Center of Informatics of Federal University of Pernambuco in partial fulfillment of the requirements for the degree of Philosophy Doctor in Computer Science.

Advisor: *Alexandre Cabral Mota*
Co-Advisor: *Alexander Romanovsky*

Recife
December 2014

André Luís Ribeiro Didier
[TBD]/ André Luís Ribeiro Didier. – Recife, December 2014-
50 p. : il. (algumas color.) ; 30 cm.

Advisor Alexandre Cabral Mota

Ph.D. Thesis – Universidade Federal de Pernambuco, December 2014.

1. Palavra-chave1. 2. Palavra-chave2. I. Orientador. II. Universidade xxx. III. Faculdade de xxx. IV. Título

CDU 02:141:005.7

Tese de doutorado apresentada por **André Luís Ribeiro Didier** ao programa de Pós-Graduação em Ciência da Computação do Centro de Informática da Universidade Federal de Pernambuco, sob o título **[TBD]**, orientada pelo **Prof. Alexandre Cabral Mota** e aprovada pela banca examinadora formada pelos professores:

Prof. André Luis de Medeiros Santos
Centro de Informática/UFPE

Prof. Alexandre Marcos Lins de Vasconselos
Centro de Informática/UFPE

Prof. Christina von Flach Garcia Chavez
Departamento de Ciência da Computação/UFBA

Prof. Leonardo Gresta Paulino Murta
Instituto de Computação/UFF

Prof. Jones Oliveira de Albuquerque
Departamento de Estatística e Informática/UFRPE

Recife
December 2014

I dedicate this thesis to Juliana, Luciana and Snowflake.

Acknowledgements

I would like to thank to the following people:

1. Alexandre
2. Sascha
3. Augusto Sampaio
4. Zoe
5. John Fitz.

...

[Text]

[More text. . .] (title)

Resumo

Resumo. . .

Palavras-chave:

Abstract

Abstract. . .

Keywords:

List of Figures

List of Tables

List of Acronyms

Contents

1

Introduction

Texto
Text (Título)

OLD:{

1. Reliability engineering
    (a) Describe safety (related to lives), reliability (related to cost).
2. Fault tolerance
    (a) Definition
    (b) Patterns
3. Functions to describe component behaviour. Include time
4. Make a ft-pattern replaceable by its function refinement.

}

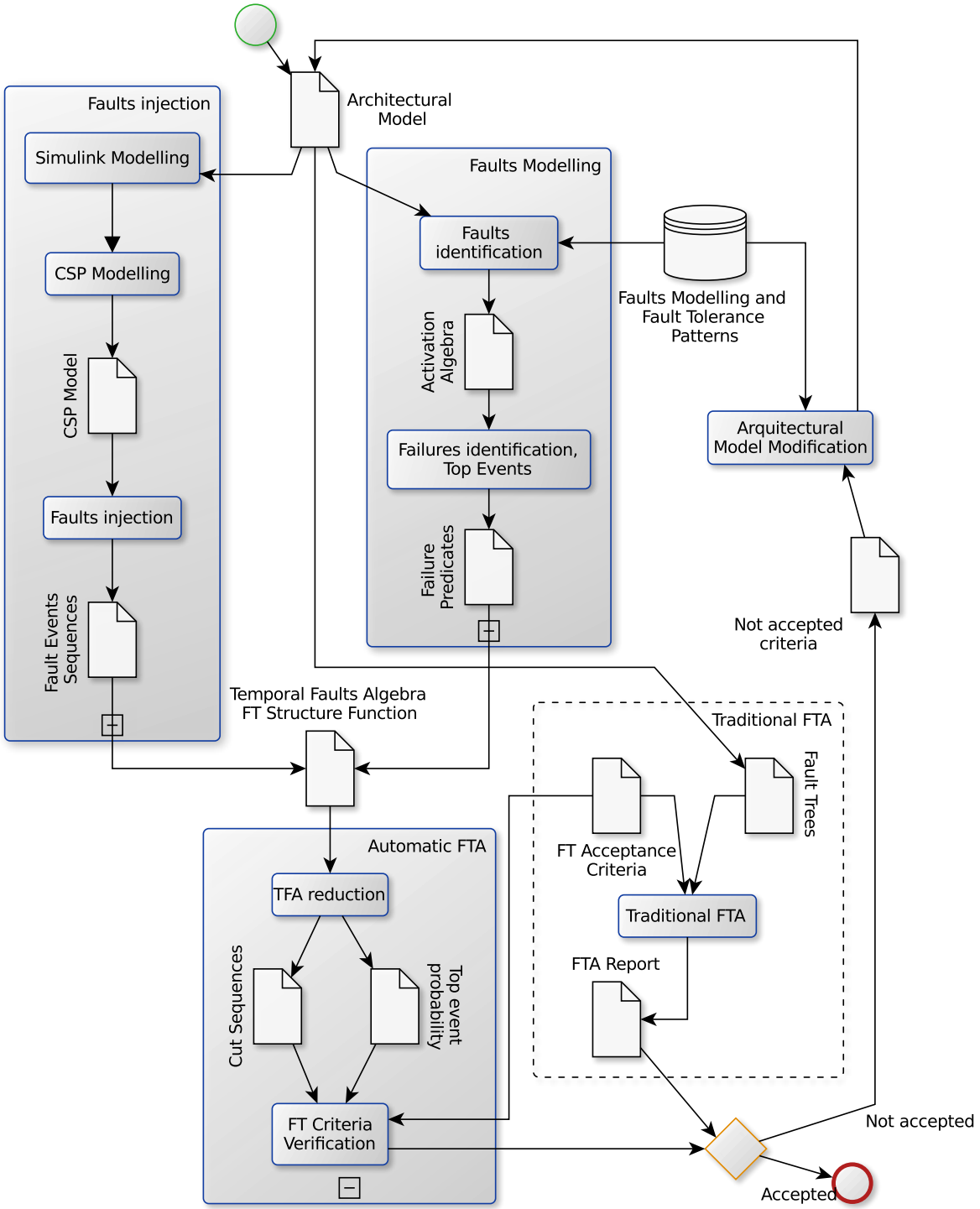

Figura 1.1: Overview

2

Modelling faults

Texto
Text (Título)

In this section we show how to model faults using functions in a component-level failure logic. We define two relations to establish our axioms: (i) failure value and failure state, (ii) order of occurrence. Using these definitions we are able to create a library of basic components that can be used as building blocks to create new components or to compose a system model.

2.1 Modelling a system

A system model is a block diagram (nodes) with directed connected edges. An edge's origin is the output of some component and the destination is one or more inputs of other component. Two output ports cannot be directly connected, nor does two inputs. It is required an output and at least one input to connect two components.

Edges and nodes contain fault-related information. An edge contains the current value of the output of the originating node. It is a Boolean indicating a nominal or an erroneous value. In the last case, the edge also contains the error mode. A node contains: (i) state information, indicating whether the component is in failure, and (ii) a logic for each output that expresses how the output reacts to internal failures and failures on the inputs. Item (ii) is very similar to the cause of output deviation in Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS) [TODO: citar def. HH]. It also includes the order of occurrence of the failures.

Differently from HiP-HOPS, an output expression also contains the nominal case to cover all possibilities on the components, to avoid missing a case.

Definition 1 (Values)**.** *The set of possible values in inputs and outputs is defined as:*

$$\text{Values} = \text{N}.\mathbb{R} \cup \text{FMode}$$

where N means a nominal value and FMode *specifies a failure mode, such as omission, commission etc. and denotes all possible failure modes for any component.*

Definition 2 (Component)**.** $C_m = (I_m, O_m)$
where $I_m = \{\text{in}_{m,1}, \ldots, \text{in}_{m,n_{in}}\}$ *are inputs of type* Values, $O_m = \{\text{out}_{m,1}, \ldots, \text{out}_{m,n_{out}}\}$ *are output functions of type* ValueFunction, $\text{ValueFunction} : (\text{Values} \times \ldots \times \text{Values}) \times FMode \rightarrow \text{Values}$, *m is the index of the component in a model, and* n_{in} *and* n_{out} *are the number of inputs and outputs.*

Definition 3 (System model)**.** $S = (Cs, A)$ *where Cs is a set of components and* $A : \text{PortIndex} \times \text{PortIndex}$ *relates a component output to another component's input,* PortIndex *is a pair of indexes where the first element is a component's index and the second is the input or output index within the component, and*

$$\forall m, j \bullet \neg \exists k \bullet A\left((m, o_m), (k, i_k)\right) \wedge A\left(\left(j, o_j\right), (k, i_k)\right)$$

Definition 4 (Model inputs and outputs)**.** *Model inputs and outputs are all components' inputs and outputs that are not in A.*

$$\text{M}_{\text{inputs}} = \bigcup_m \bullet (m,k) \mid \forall m, j, k \bullet \neg A\left(\left(j, o_j\right), (m,k)\right)$$

and

$$\text{M}_{\text{outputs}} = \bigcup_m \bullet (m,k) \mid \forall m, j, k \bullet \neg A\left((m,k), \left(j, i_j\right)\right)$$

where $C_m = (I_m, O_m)$, $I_m = (i_{m,1}, \ldots, i_{m,n_{in}})$ *and* $O_m = (o_{m,1}, \ldots, o_{m,n_{out}})$

Lemma 1 (Inputs and Outputs completeness)**.** *In a model, every input and output is either in A, or in* $\text{M}_{\text{inputs}} \cup \text{M}_{\text{outputs}}$*:*

$$\forall m, i \bullet (m,i) \in \text{M}_{\text{inputs}} \vee (m,i) \in \text{ran} A$$

$$\forall m, o \bullet (m,o) \in \text{M}_{\text{outputs}} \vee (m,o) \in \text{dom} A$$

Demonstração. TODO: sorry □

The output functions are defined using a *valued tautology* logic. It adds value information to a Boolean expression and can be generally understood as a guarded-value expression. The evaluation of the expression of all guards is always *true* (tautology). The most basic example of a valued tautology is a $\top$ expression and its associated value V : Values:

$$\top^V$$

Definition 5 (Valued tautology)**.** *A valued tautology is defined as:*

$$\begin{aligned} ValuedTautology &\mathrel{\widehat{=}} E_0^{V_0}\, op_0 \ldots op_m\, E_m^{V_m} \\ E_0\, op_0 \ldots op_m\, E_m &= \top \\ &\forall i, j \bullet E_i \wedge E_j \Rightarrow V_i = V_j \end{aligned}$$

where op_m is a Boolean operator, E_m is a Boolean expression and V_m is a value of type Values.

Lemma 2 (Valued tautology *OR*)**.** *Given two Boolean variables A and B, their respective values U and V, and a third value Q, $A^U \vee B^V \vee (\neg A \wedge \neg B)^Q$ is a valued tautology:*

$$\begin{aligned} A^U \vee B^V \vee (\neg A \wedge \neg B)^Q \mathrel{\widehat{=}} &(A \wedge \neg B)^U \vee (\neg A \wedge B)^V \vee \\ &(A \wedge B \wedge (U = V))^U \vee \\ &(\neg A \wedge \neg B)^Q \end{aligned}$$

Demonstração. Assume $E = A \vee B$, $\neg E = (\neg A \wedge \neg B)$ is also in the expression, thus:

$$E \vee \neg E = \top$$

TODO: completar □

Lemma 3 (Valued tautology *AND*)**.** *Give two Boolean variables A and B, their respective values U and V and a third value Q, $A^U \wedge B^V \vee (\neg A \vee \neg B)^Q$ is a valued tautology:*

$$\begin{aligned} \left(A^U \wedge B^V\right) \vee (\neg A \vee \neg B)^Q \mathrel{\widehat{=}} &(A \wedge \neg B)^Q \vee (\neg A \wedge B)^Q \vee \\ &(A \wedge B \wedge U = V)^U \vee \\ &(\neg A \wedge \neg B)^Q \end{aligned}$$

Demonstração. Assume $E = A \wedge B$, $\neg E = (\neg A \vee \neg B)$ is also in the expression, thus:

$$E \vee \neg E = \top$$

TODO: completar □

TODO: Citação do capítulo ou da seção: to be or not to be.

Definition 6 (Value of a valued tautology)**.** *The value of a valued tautology is the i-eth value when the i-eth expression evaluates to* true*. Given a valued tautology $T = E_0^{V_0}\, op_0 \ldots op_m\, E_m^{V_m}$, the value of T is defined by the function ρ:*

$$\exists i \bullet E_i \wedge \rho\,(T) = V_i$$

If an $E_j (i \neq j)$ also evaluates to *true*, the value is the same, accordingly to definition 5.

Definition 7 (Rules for ρ)**.** *Given a predicate $P(.)$, and Boolean expressions E and E_i, $i \in [1,n]$:*

$$P\left(\rho\left(E_1^{V_1} \vee \ldots \vee E_n^{V_n}\right)\right) = (P(V_1) \wedge E_1) \vee \ldots \vee (P(V_n) \wedge E_n) \quad (2.1)$$

$$E^{\rho\left(E_1^{V_1} \vee \ldots \vee E_n^{V_n}\right)} = (E \wedge E_1)^{V_1} \vee \ldots \vee (E \wedge E_n)^{V_n} \quad (2.2)$$

TODO: Acrescentar teorema sobre como as funções de saída geram as equações para as árvores.

Accordingly to the work reported in **?**), a fault tree structure function is not only a Boolean expression. It exemplifies with the expression $true \wedge \neg closed(barriers)$, where we cannot just replace '$\wedge$' to an AND-gate, because $true$ is not a basic event, nor $\neg closed(barriers)$ expresses a fault. The work reported in **?**) defines a formal model of static fault trees considering operands as events.

Definition 8 (Fault tree gate)**.** *Given F_f, $f \in [0,n_F], n_F \geq 0$, Boolean variables where if F_f evaluates to $true$, then a basic fault event occurs, a fault tree gate G combines all F_f with a Boolean operator op:*

$$G = F_0\, op \ldots op\, F_{n_F}$$

If $n_F = 0$ then the gate reduces to the variable F_0.

Definition 9 (Fault tree structure function)**.** *Given F_f, $f \in [0,n_F]$ Boolean variables, and fault tree gates G_g, $g \in [0,n_G]$, a structure function $SF : (Boolean \times \ldots \times Boolean) \rightarrow Boolean$ is also a fault tree gate with operator op:*

$$SF(F_0,\ldots,F_{n_F}) = G_0(F_0,\ldots,F_{n_F})\, op \ldots op\, G_{n_G}(F_0,\ldots,F_{n_F})$$

where if F_f evaluates to $true$, then a basic fault event occurs. Then SF represents a fault tree.

Theorem 1 (A predicate over a valued tautology is a structure function of a fault tree)**.** *Given F_f, $f \in [1,n_F]$ Boolean variables that represent basic fault events, G_g, $g \in [0,n_G]$ fault tree gates that contains only F_f, and $T = G_0^{V_0}\, op_0 \ldots op_{n_G}\, G_{n_G}^{V_{n_G}}$ a valued tautology, then a predicate P over F_f is a structure function of T.*

Demonstração. TODO: sorry. □

2.1.1 Example of valued tautology

A single battery b_x is defined as:

$$C_{b_x} = (\emptyset, \{\text{out}_{b_x,1}\})$$
$$\text{out}_{b_x,1} = F_{b_x}^{\text{Omission}} \vee \neg F_{b_x}^{\text{N.5}}$$

A simple monitor *mon*:

$$
\begin{aligned}
C_{mon} &= \left(\left\{\mathrm{in}_{mon,1}, \mathrm{in}_{mon,2}\right\}, \left\{\mathrm{out}_{mon,1}\right\}\right) \\
\mathrm{out}_{mon,1} &= \left(\neg F_{mon} \wedge \rho\left(\mathrm{in}_{mon,1}\right) \geq \mathrm{N}.2\right)^{\rho\left(\mathrm{in}_{mon,1}\right)} \vee \\
&\quad \left(\neg F_{mon} \wedge \rho\left(\mathrm{in}_{mon,1}\right) < \mathrm{N}.2\right)^{\rho\left(\mathrm{in}_{mon,2}\right)} \vee \\
&\quad \left(F_{mon} \wedge \rho\left(\mathrm{in}_{mon,1}\right) \geq \mathrm{N}.2\right)^{\rho\left(\mathrm{in}_{mon,2}\right)} \vee \\
&\quad \left(F_{mon} \wedge \rho\left(\mathrm{in}_{mon,1}\right) < \mathrm{N}.2\right)^{\rho\left(\mathrm{in}_{mon,1}\right)}
\end{aligned}
$$

Combining both components into a system S_{mon} results in the following system model:

$$
\begin{aligned}
S_{mon} &= \left(\left\{C_{b_1}, C_{b_2}, C_{mon}\right\}, A_{mon}\right) \\
&\quad A_{mon}\left(\left(b_1, 1\right), \left(mon, 1\right)\right) \\
&\quad A_{mon}\left(\left(b_2, 1\right), \left(mon, 2\right)\right)
\end{aligned}
$$

And, accordingly to definition 4:

$$
\begin{aligned}
\mathrm{M}_{\mathrm{inputs}} &= \emptyset \\
\mathrm{M}_{\mathrm{outputs}} &= \left\{\left(mon, 1\right)\right\}
\end{aligned}
$$

The system model output is the result of expanding $\mathrm{out}_{mon,1}$ with the output functions of

the connected single batteries $\text{out}_{b_1,1}$ and $\text{out}_{b_2,1}$:

$$\begin{aligned}
\text{out}_{mon,1} = &\left(\neg F_{mon} \wedge \rho\left(F_{b_1}^{\text{Omission}} \vee \neg F_{b_1}^{\text{N.5}}\right) \geq \text{N.2}\right)^{\rho\left(F_{b_1}^{\text{Omission}} \vee \neg F_{b_1}^{\text{N.5}}\right)} \vee \\
&\left(\neg F_{mon} \wedge \rho\left(F_{b_1}^{\text{Omission}} \vee \neg F_{b_1}^{\text{N.5}}\right) < \text{N.2}\right)^{\rho\left(F_{b_2}^{\text{Omission}} \vee \neg F_{b_2}^{\text{N.5}}\right)} \vee \\
&\left(F_{mon} \wedge \rho\left(F_{b_1}^{\text{Omission}} \vee \neg F_{b_1}^{\text{N.5}}\right) \geq \text{N.2}\right)^{\rho\left(F_{b_2}^{\text{Omission}} \vee \neg F_{b_2}^{\text{N.5}}\right)} \vee \\
&\left(F_{mon} \wedge \rho\left(F_{b_1}^{\text{Omission}} \vee \neg F_{b_1}^{\text{N.5}}\right) < \text{N.2}\right)^{\rho\left(F_{b_1}^{\text{Omission}} \vee \neg F_{b_1}^{\text{N.5}}\right)} \\
\text{out}_{mon,1} = &\left(\neg F_{mon} \wedge \neg F_{b_1}\right)^{\rho\left(F_{b_1}^{\text{Omission}} \vee \neg F_{b_1}^{\text{N.5}}\right)} \vee \text{ by definitions 6 and 7} \\
&\left(\neg F_{mon} \wedge F_{b_1}\right)^{\rho\left(F_{b_2}^{\text{Omission}} \vee \neg F_{b_2}^{\text{N.5}}\right)} \vee \left(F_{mon} \wedge \neg F_{b_1}\right)^{\rho\left(F_{b_2}^{\text{Omission}} \vee \neg F_{b_2}^{\text{N.5}}\right)} \vee \\
&\left(F_{mon} \wedge F_{b_1}\right)^{\rho\left(F_{b_1}^{\text{Omission}} \vee \neg F_{b_1}^{\text{N.5}}\right)} \\
\text{out}_{mon,1} = &\left(\neg F_{mon} \wedge \neg F_{b_1}\right)^{\text{N.5}} \vee \text{ by definition 7} \\
&\left(\left(\neg F_{mon} \wedge F_{b_1}\right) \wedge F_{b_2}\right)^{\text{Omission}} \vee \left(\left(\neg F_{mon} \wedge F_{b_1}\right) \wedge \neg F_{b_2}\right)^{\text{N.5}} \vee \\
&\left(\left(F_{mon} \wedge \neg F_{b_1}\right) \wedge F_{b_2}\right)^{\text{Omission}} \vee \left(\left(F_{mon} \wedge \neg F_{b_1}\right) \wedge \neg F_{b_2}\right)^{\text{N.5}} \vee \\
&\left(F_{mon} \wedge F_{b_1}\right)^{\text{Omission}}
\end{aligned}$$

Finally, we obtain the system fault tree from a predicate (see theorem 1):

$$\begin{aligned}
\rho\left(\text{out}_{mon,1}\right) = \text{Omission} \mathrel{\widehat{=}} \rho\big(&\left(\neg F_{mon} \wedge \neg F_{b_1}\right)^{\text{N.5}} \vee \\
&\left(\left(\neg F_{mon} \wedge F_{b_1}\right) \wedge F_{b_2}\right)^{\text{Omission}} \vee \\
&\left(\left(\neg F_{mon} \wedge F_{b_1}\right) \wedge \neg F_{b_2}\right)^{\text{N.5}} \vee \\
&\left(\left(F_{mon} \wedge \neg F_{b_1}\right) \wedge F_{b_2}\right)^{\text{Omission}} \vee \\
&\left(\left(F_{mon} \wedge \neg F_{b_1}\right) \wedge \neg F_{b_2}\right)^{\text{N.5}} \vee \\
&\left(F_{mon} \wedge F_{b_1}\right)^{\text{Omission}} \big) = \text{Omission} \\
\rho\left(\text{out}_{mon,1}\right) = \text{Omission} \mathrel{\widehat{=}} \rho\big(&\left(\left(\neg F_{mon} \wedge F_{b_1}\right) \wedge F_{b_2}\right)^{\text{Omission}} \vee \\
&\left(\left(F_{mon} \wedge \neg F_{b_1}\right) \wedge F_{b_2}\right)^{\text{Omission}} \vee \\
&\left(F_{mon} \wedge F_{b_1}\right)^{\text{Omission}} \big) = \text{Omission} \\
\rho\left(\text{out}_{mon,1}\right) = \text{Omission} \mathrel{\widehat{=}} &\left(\neg F_{mon} \wedge F_{b_1} \wedge F_{b_2}\right) \vee \\
&\left(F_{mon} \wedge \neg F_{b_1} \wedge F_{b_2}\right) \vee \\
&\left(F_{mon} \wedge F_{b_1}\right) \\
\rho\left(\text{out}_{mon,1}\right) = \text{Omission} \mathrel{\widehat{=}} &\left(F_{b_1} \wedge F_{b_2}\right) \vee \left(F_{mon} \wedge \left(F_{b_1} \vee F_{b_2}\right)\right)
\end{aligned}$$

2.1.2 Failure value and failure state

To illustrate these definitions, suppose a component C that contains only one output, and its failure logic depends only on its internal state: if it is faulty, it produces an omission value O_M, otherwise it produces a nominal value N. The failure logic for its output O is: $O = F \wedge O_M \vee \neg F \wedge N$.

TODO: all Boolean operators relate to $FailureMode$ as: if the Boolean value is $true$, then it results in the $FailureMode$ value, otherwise it results in $false$. It creates an invariant that all output functions are tautologies.

2.1.3 Order of occurrence

The order of occurrence of events is defined as a sequence value as it is in HiP-HOPS: it is a natural value assigned to each input. The value does not contain gaps, so always there is a pair of input that the modulus of the difference of their sequence values is 1, or the modulus of the difference is 0 for all pairs of inputs. For example, if we have two variables, they can be assigned values 0 and 1 or 1 and 2, but not 0 and 2.

We define a relation $S : Input \rightarrow \mathbb{N}$ as the sequence value for the given input.

3

Temporal Fault Trees

Texto
Text (Título)

Compared to traditional Fault Trees, Temporal Fault Trees can describe the order of occurrence of events. A specific order of occurrence causes a top event, thus the concept of failure expressions is augmented. Instead of using traditional Boolean gates, they use special gates named temporal gates (see table 3.1). They differ on what kind of input they compare: Boolean gates compare Boolean values whilst temporal gates compare *sequence values*. A sequence value is a natural number in which each basic event happens on a specific value, but there are no gaps on the sequence (see table 3.2). Zero values indicate that the event has not happened. As in Boolean logic, temporal logic also has truth tables, which are named Temporal Truth Tables.

TFTs can be written as an expression where each basic event in a tree is a variable in the expression. Table 3.3 shows a TTT—a result of direct application of the equations shown in table 3.2—for each temporal gate basic formula, using events A and B.

3.1 Formalisation of Temporal Fault Trees

In this section, I propose a formalisation in Communicating Sequential Processes (CSP) of TFT to enable a comparison of TFTs and the calculation of the probability of occurrence of top events.

The formalisation occurs in three steps: (i) formalising sequence values calculations to achieve TTTs for a given temporal expression (section 3.1.1), (ii) converting these TTTs to sequences of events (section 3.1.2), and (iii) building a process to call these events to verify refinements (section 3.1.3).

Tabela 3.1: Temporal gates

Name	Abbrev.	Operator	Description
Or	OR	+	Similarly to the Boolean operator, whenever one of its inputs is higher than 0, it outputs its value.
And	AND	.	When both inputs are higher than zero, it outputs the value of the maximum sequence value.
Priority Or	POR	|	When the first input is higher than zero, it outputs its value and the second output may or may not occur. It will not output a value when only the second output is higher than the first output.
Priority And	PAND	<	It outputs the value of the second output if the first output occurs strictly before the second.
Synchronous And	SAND	&	It outputs the value of the sequence value when both inputs occur.

Tabela 3.2: Sequence value (SV) equations for each temporal gate

Gate	Sequence value formula
OR	$\mathrm{SV}(A+B) = \begin{cases} \min(\mathrm{SV}(A),\mathrm{SV}(B)) & ,\mathrm{SV}(A)>0 \wedge \mathrm{SV}(B)>0 \\ \max(\mathrm{SV}(A),\mathrm{SV}(B)) & \text{otherwise} \end{cases}$
AND	$\mathrm{SV}(A.B) = \begin{cases} \max(\mathrm{SV}(A),\mathrm{SV}(B)) & ,\mathrm{SV}(A)>0 \wedge \mathrm{SV}(B)>0 \\ 0 & \text{otherwise} \end{cases}$
POR	$\mathrm{SV}(A\|B) = \begin{cases} \mathrm{SV}(A) & ,\mathrm{SV}(B)=0 \vee \mathrm{SV}(A)<\mathrm{SV}(B) \\ 0 & \text{otherwise} \end{cases}$
PAND	$\mathrm{SV}(A<B) = \begin{cases} \mathrm{SV}(B) & ,\mathrm{SV}(A)>0 \wedge \mathrm{SV}(A)<\mathrm{SV}(B) \\ 0 & \text{otherwise} \end{cases}$
SAND	$\mathrm{SV}(A\&B) = \begin{cases} \mathrm{SV}(A) & ,\mathrm{SV}(A)=\mathrm{SV}(B) \\ 0 & \text{otherwise} \end{cases}$

Tabela 3.3: TTT for each temporal gate

A	B	$A+B$	$A.B$	$A\|B$	$A<B$	$A\&B$
0	0	0	0	0	0	0
0	1	1	0	0	0	0
1	0	1	0	1	0	0
1	1	1	1	1	0	1
1	2	1	2	1	2	0
2	1	1	2	0	0	0

Tabela 3.4: TTT for the expression $(A+B)<C$

A	B	C	$A+B$	$(A+B)<C$
0	0	0	0	0
0	0	1	0	0
0	1	0	1	0
0	1	1	1	0
0	1	2	1	2
0	2	1	2	0
1	0	0	1	0
1	0	1	1	0
1	0	2	1	2
1	1	0	1	0
1	1	1	1	0
1	1	2	1	2
1	2	0	1	0
1	2	1	1	0
1	2	2	1	2
1	2	3	1	3
1	3	2	1	2
2	0	1	2	0
2	1	0	1	0
2	1	1	1	0
2	1	2	1	2
2	1	3	1	3
2	2	1	2	0
2	3	1	2	0
3	1	2	1	2
3	2	1	2	0

3.1.1 Sequence Value Calculus

Given two sequence values as inputs, each temporal gate expression is encoded as a CSP function as following:

$$OR(a,b) = \begin{cases} min(a,b) & \text{if } a>0 \wedge b>0 \\ max(a,b) & \text{otherwise} \end{cases} \quad (3.1)$$

$$AND(a,b) = \begin{cases} max(a,b) & \text{if } a>0 \wedge b>0 \\ 0 & \text{otherwise} \end{cases} \quad (3.2)$$

$$POR(a,b) = \begin{cases} a & \text{if } a>0 \wedge (b=0 \vee a<b) \\ 0 & \text{otherwise} \end{cases} \quad (3.3)$$

$$PAND(a,b) = \begin{cases} b & \text{if } a>0 \wedge a<b \\ 0 & \text{otherwise} \end{cases} \quad (3.4)$$

$$SAND(a,b) = \begin{cases} a & \text{if } a=b \\ 0 & \text{otherwise} \end{cases} \quad (3.5)$$

Combining these functions, any temporal expression can be written. For example, given the expression $(A+B)<C$, it can be written as: $PAND(OR(A,B),C)$. Table 3.4 shows the truth table for this example formula, by applying the function definitions for each row.

3.1.2 From TTTs to sequences of events

Obtaining (a set of) sequences of events is an intermediate step to get a process that represents a TFT. This step is an optimisation to remove non-determinism, trimming the paths that lead to the top-level event. It combines paths that have the same initial event, building an hierarchical structure of choices.

The function TTT below creates a set o tuples of size $n+1$, where n is the number of basic events on the expression and the value on position $n+1$ is the result of the application of the temporal expression:

$$TTT :: TExp \Longrightarrow \mathscr{P}\left(\underbrace{\mathrm{SV} \times \ldots \times \mathrm{SV}}_{n+1}\right)$$

$$TTT(expression) = \Big\{(a_1,\ldots,a_n, expression(a_1,\ldots,a_n)) \;\Big|\; (a_1,\ldots,a_n) \in TTT_{inputs}(n)\Big\} \tag{3.6}$$

where $1,\ldots,n$ are the indexes of basic events, $\mathrm{SV} = \{0,\ldots,n\}$ are sequence values and TTT_{inputs} defines a set of tuples of size n that represents the events (inputs) for each row in a TTT:

$$TTT_{inputs} :: I \Longrightarrow \mathscr{P}\left(\underbrace{\mathrm{SV} \times \ldots \times \mathrm{SV}}_{n}\right) \tag{3.7}$$

$$TTT_{inputs}(n) = \Big\{(a_1,\ldots,a_n) \;\Big|\; max(a_1,\ldots,a_n) = card\left(\{a_1,\ldots,a_n\} \setminus \{0\}\right)\Big\} \tag{3.8}$$

where $I = \{1,\ldots,n\}$. Note that the clause $max(a_1,\ldots,a_n) = card\left(\{a_1,\ldots,a_n\} \setminus \{0\}\right)$ guarantees that there are no gaps between two a_i's, satisfying the TFT property for sequence values.

For the example expression $(A+B) < C$, the functions TTT and TTT_{inputs} return a set with cardinality 26.

Finally, we create a data structure to avoid non-determinism and optimise the final process creation. This data structure is created in two steps:

1. Each tuple is converted into a set of pairs recursively where the first element is an available event and the second element is a set of options of events to choose. This set of options may contain others pairs recursively.

   $$SoP(sync_{ev}, (a_1,\ldots,a_n)) = \tag{3.9}$$

   where $sync_{ev}$ is an event that indicates that the following events occur with the same sequence value.

2. These pairs are then merged with respect to the first element. If there two pairs with

the same first element, they are merged, making a union of the second element sets.

3.1.3 Checking process refinements

Using the tuples defined by eq. (3.8), we now define a process that represents the execution of a temporal expression.

4

Formalisation of Hazard Management

This chapter formalises our hazard management approach using CSP.
Hazard properties, extracted from Ericson II (2005):

1. Hazardous element (HE);
2. Initiating mechanisms (IM): sequence of events to activate the hazard;
3. Target and Threat (T/T): the person or system and its corresponding threat;
4. Status: open or closed. A hazard can only be closed if it has been verified through analysis, inspection or testing that the safety requirements are implemented in the design and successfully tested for effectiveness.
5. Mitigation: lists the recommended actions obtained by all analysis
    - Eliminate through design selection;
    - Incorporate safety devices;
    - Provide warning devices;
    - Develop procedures and training;
    - Control hazard through design methods.
6. Mishap Risk Index (initial, current, final) – qualitative measurement:
    - Severity
        (a) Catastrophic;
        (b) Critical;
        (c) Marginal;
        (d) Negligible.
    - Probability
        (a) Frequent;

(b) Probable;

(c) Occasional;

(d) Remote;

(e) Improbable.

Scope of system to which Hazard Analysis make sense: those that have signal, material and energy. All system functional elements (KOSSIAKOFF et al., 2011, p. 47) are:

Signal. Generate, transmit, distribute, and receive signals used in passive or active sensing and in communications;

Data. Analyse, interpret, organize, query, and/or convert data and information into forms desired by the user or other systems;

Material. Provide system structural support or enclosure, or transform the shape, composition, or location of material substances;

Energy. Provide and convert energy or propulsive power to the system.

Definition 10 (Identifier)**.** *It is modelled as any user-defined data type. It can be a sequence of alphanumeric symbols (a descriptive text of the element) or a natural number (for example: an identification of a system element). It is a basic data type* Id *used in other definitions.*

Definition 11 (Component)**.** *A component is any functional or physical block within a system that may contain a hazard. It is defined as an identifier and a set of identifiers:*
$\mathrm{C} : \mathrm{Id} \times \mathbb{P}\,\mathrm{Id}$.
Every subcomponent cannot contain any ancestor. Given a component $C = (c_0, CS_0)$*, the subcomponents set is obtained by a folding operation.*
$\mathrm{subcomponents}\, C = \{\{\mathrm{first}\,(C_i)\} \cup \mathrm{subcomponents}\, C_i | C_i \in \mathrm{second}\,(C)\}$.
For every component C: $\forall C_i \in CS_0 \bullet c_0 \notin \mathrm{subcomponents}\, C_i$

Definition 12 (Hazardous element)**.** *A hazardous element represents an element that can potentially harm someone or something.* $\mathrm{HE} : \mathrm{Id}$.

Definition 13 (Initiating mechanism)**.** *An initiating mechanism is an event*[1] *on the system.* $\mathrm{IM} : \mathrm{Id}$

Definition 14 (Target/Threat)**.** *A target and threat pair represents the person or system that is harmed if a hazardous element is activated by the initiating mechanisms and the nature of the harm.* $\mathrm{T/T} : \mathrm{Id} \times \mathrm{Id}$.

[1] It is not a CSP event; when we refer to a CSP event we will always refer to a channel.

Definition 15 (Hazard). *Given a hazardous element HE, a sequence of n initiating mechanisms $\langle IM_1, \ldots, IM_n \rangle$ and a target-threat pair T/T, a hazard H is defined as: $H : \mathrm{Id}$ and is obtained by a total, bijective function $\overset{\triangle}{\rightsquigarrow}$:*
$\overset{\triangle}{\rightsquigarrow} : \mathrm{HE} \times \mathrm{T}/\mathrm{T} \times \mathrm{seq}\,\mathrm{IM} \underset{\mathrm{TB}}{\rightarrow} \mathrm{Id}$
which is written as:
$H = HE \underset{\langle IM_1, \ldots, IM_n \rangle}{\overset{\triangle}{\rightsquigarrow}} T/T.$

Definition 16 (Probability function). *A probability function P is used to associate values to elements in the system. It is defined through analysis. $P : \mathrm{Id} \nrightarrow \mathbb{R}$*

Definition 17 (Hazard analysis type). *A hazard analysis type is a categorization of the analysis methods as follows:*

CD-HAT. *Conceptual design hazard analysis type*

PD-HAT. *Preliminary design hazard analysis type*

DD-HAT. *Detailed design hazard analysis type*

SD-HAT. *System design hazard analysis type*

OD-HAT. *Operations design hazard analysis type*

HD-HAT. *Health design hazard analysis type*

RD-HAT. *Requirements design hazard analysis type*

Definition 18 (Analysis method). *An analysis method $\mathbb{AM}$, such that $\mathbb{AM} : \mathrm{Id}$, describes a method used to analyse an element. It is defined as an enumerated set with the value correspondence shown in table 4.1.*

Definition 19 (Time). *Time $\mathscr{T}$ is abstracted with equal and before relations. It is defined as a sequence of time tags. $\mathscr{T} : \mathrm{seq}\,\mathbb{T}$.*

Definition 20 (Time tag). *A time tag represents an instantaneous observation of time. It is defined as an identification: $\mathbb{T} : \mathrm{Id}$*

Definition 21 (Analysis). *Given an analysis method M, such that $M : \mathbb{AM}$, a time tag t, such that $t : \mathbb{T}$, an element E, such that $E : \mathrm{Id}$, and a probability p, such that $p : \mathbb{R}$, an analysis A is defined as:*
$A : (\mathrm{Id} \times \mathbb{T}) \times (\mathbb{AM} \times \mathbb{R})$
$A = ((E,t),(M,p))$

The probability function is then updated accordingly to an analysis. For example, for the analysis above, the probability function P can be updated as: $P = P \oplus \{E \mapsto p\}$.

Seq.	Abbrev.	Hazard analysis type	Description
1	PHL*	CD-HAT	Preliminary Hazard List
2	PHA*	PD-HAT	Preliminary Hazard Analysis
3	SSHA*	DD-HAT	Subsystem Hazard Analysis
4	SHA*	SD-HAT	System Hazard Analysis
5	O&SHA*	OD-HAT	Operating and Support Hazard Analysis
6	HHA*	HD-HAT	Health Hazard Assessment
7	SRCA*	RD-HAT	Safety Requirements/Criteria Analysis
8	FTA	SD-HAT, DD-HAT	Fault Tree Analysis
9	ETA	SD-HAT	Event Tree Analysis
10	FMEA	DD-HAT	Failure Mode and Effects Analysis
11	FaHA	DD-HAT	Fault Hazard Analysis
12	FuHA	SD-HAT, DD-HAT	Functional Hazard Analysis
13	SCA	SD-HAT, DD-HAT	Sneak Circuit Analysis
14	PNA	SD-HAT, DD-HAT	Petri Net Analysis
15	MA	SD-HAT, DD-HAT	Markov Analysis
16	BA	SD-HAT	Barrier Analysis
17	BPA	DD-HAT	Bent Pin Analysis
18	HAZOP	SD-HAT, DD-HAT	Hazard and Operability Analysis
19	CCA	SD-HAT, DD-HAT	Cause–Consequence Analysis
20	CCFA	SD-HAT, DD-HAT	Common Cause Failure Analysis
21	MORT	SD-HAT, DD-HAT	Management Oversight Risk Tree Analysis
22	SWSCA		
23	SWHA		Software Safety Assessment
24	THA		Threat Hazard Assessment

Tabela 4.1: Analysis methods

Definition 22 (Hazard activation probability)**.** *Given a probability function P, such that* $P : \mathrm{Id} \nrightarrow \mathbb{R}$*, a hazardous element HE, a sequence of n initiating mechanisms* $\langle IM_1, \ldots, IM_n \rangle$ *and a target-threat pair* T/T*, the probability of the activation of the hazard* $H = HE \overset{\triangle}{\underset{\langle IM_1,\ldots,IM_n\rangle}{\rightsquigarrow}} T/T$ *is:*

$P(H) = P(HE) \times P(IM_1) \times \ldots \times P(IM_n)$

Definition 23 (Initiating mechanism)**.** *An initiating mechanism is a relation between components attributes and their unwanted behaviour or state. Some component attribute examples Ericson II (2005):*

Hardware *Failure modes, hazardous energy sources,* $IM_{H1} = (C_1, 1)$, $IM_{H2} = (C_1, 2)$*;*

Software *Design errors, design incompatibilities,* $IM_{S1} = (C_2, 3)$, $IM_{S2} = (C_2, 4)$

Personnel *Human error, human injury, human control interface,* $IM_{P1} = (C_3, 3)$*;*

Environment *Weather, external equipment;*

Procedures *Instructions, tasks, warning notes;*

Interfaces *Erroneous input/output, unexpected complexities;*

Functions *Fail to perform, performs erroneously;*

Facilities *Building faults, storage compatibility, transportation faults;*

By the relations between components and hazards, and hazards and initiating mechanisms, a dependency graph can be built, so if a components changes, the corresponding hazards changes as well.

Definition 24 (Mishap)**.** *A mishap is defined as a pair of the probability of a hazard and the severity in case it happens:* $M : \mathbb{R} \times \mathbb{S}v$*.*

Where $\mathbb{S}v$ is a finite set, such that $\mathbb{S}v \subset \mathbb{N}$. It contains the following values:

1 = Catastrophic;

2 = Critical;

3 = Marginal and

4 = Negligible.

Definition 25 (Hazard Model)**.** *A* HM*...*

4.1 Refinement

In this section we define a refinement relation on a hazard model of a system.

TODO: Add definitions on the hazard model

- Set of analyses;
- Probability function;
- Addition of analysis $\Rightarrow$ updates probability function;
- Set of hazards;
- Dependency graph (connection between components);

There a few situations in which a refinement is desirable:

Hazard analysis. A hazard model is said to be an analysis of another model if at least one new analysis was added.

Hazard mitigation. A hazard model is said to be a mitigation of another if at least one hazard is mitigated by decrement of probability value. If P is the probability function of the previous model and P' is the probability function of the refined model, then:
$\mathrm{dom}\, P \subseteq \mathrm{dom}\, P' \land \exists e \in \langle setofhazards \rangle \cap \mathrm{dom}\, P \bullet P'(e) < P(e)$.

Hazard discovery. A hazard model is said to be a discovery of another if the hazards set is augmented with a new discovered hazard.

Component replacement. A valid component replacement in a model is achieved if all analyses for the replaced component are updated with the new component and the probability of activation is less than or equal to the previous ones.

Component addition. A valid component addition in a model is achieved if the component is not in any other component hierarchy or, if it is, then the analyses of all affected component hierarchy and related components are updated and the probability of activation is less than or equal to the previous ones.

Component removal. A valid component removal in a model is achieved if the analyses of all affected component hierarchy and related components are updated and the probability of activation is less than or equal to the previous ones.

5

Related work

Texto
Text (Título)

5.1 Title

5.1.1 Subtitle

Plain text.

5.1.2 Another subtitle

More plain text.

References

Ericson II, C. A. **Hazard Analysis Techniques for System Safety**. [S.l.]: Wiley-Interscience, 2005.

KOSSIAKOFF, A. et al. **Systems Engineering Principles and Practice**. 2nd.ed. [S.l.]: Wiley-Interscience, 2011.

Appendix