FEDERAL UNIVERSITY OF RIO GRANDE DO SUL
INSTITUE OF INFORMATICS
BACHELOR OF COMPUTER SCIENCE

LEANDRO AVILA DA SILVA

HWSafetyToolbox:
A Tool for Modeling Safety-Related Hardware

Graduation Thesis.

Advisor: Prof. Dr. Taisy da Silva Weber
Co-advisor: Dipl.-Ing. Sebastian Wille
Co-advisor: Dipl.-Inf. Bastian Zimmer

Kaiserslautern
2014

AKNOWLEDGEMENTS

I would like to thank my parents, Eurides Michel da Silva and Rosângela Avila da Silva, and my sister, Débora de Cássia Avila da Silva, for all the support, friendship and companionship provided, that comfort me so much, even now that I'm far away from you. Thanks to my grandfather, Osvaldo Israel de Avila, and to my late grandmother, Norma Azevedo de Avila, for all the wonderful moments we had together, which I will always keep fondly in my mind.

Thank you to all my teachers from the Informatics Institute from UFRGS, for all the education passed throughout my graduation. A special thanks to Prof. Taisy da Silva Weber, for patiently advise me through the production of this thesis, and also for the dedication to kindly help students in everything that you can. Without your advices I would not be here in Kaiserslautern.

From the University of Kaiserslautern, firstly, I would like to thank the Prof. Dr.-Ing. Norbert Wehn for giving me the opportunity to work with his highly qualified group in this exchange program, it was a really great experience. Thank you to Dipl.-Ing. Sebastian Wille and Dipl.-Ing. Matthias Jung for being my advisors on this project and for providing all the support and know-how necessary.

From the Fraunhofer IESE, thanks to everyone that helped on this project, but mainly thank you to Dipl.-Inf Bastian Zimmer for all the knowledge passed, patience and contribution to this work.

Thank you to my girlfriend, Arghavan Hosseinzadeh, for all the good moments, for the personal support given and for understanding when I couldn't be present because of this thesis and project. This year was much better thanks to your presence. And to your foods.

Finally, thank you to all my friends from Kaiserslautern and from Brasil. This project would be impossible to do without the good times that we had in between workdays.

RESUMO

O desenvolvimento de sistemas embarcados para domínios de aplicação de segurança functional crítica está em ascensão. Considerando que os requisitos de segurança para esses sistemas são altos e difíceis de integrar ao processo de desenvolvimento, normas foram criadas, fornecendo um ciclo de vida seguro para simplificar a produção de sistemas críticos de segurança. Para o domínio de aplicação de veículos rodoviários, a norma ISO 26262 foi criada, preenchendo as necessidades específicas da indústria automotiva em alcançar a segurança funcional de seus equipamentos eletro-eletrônicos.

Este trabalho de graduação apresenta o design e implmentação da HWSafetyToolbox. HWSafetyToolbox é uma ferramenta que ajuda na avaliação das Métricas de Arquitetura de Hardware (*Hardware Architecture Metrics*), um processo descrito na fase de desenvolvimento de hardware da norma ISO 26262. A ferramenta apresenta um novo modo de modelar um sistema e seus *safety goals,* a abordagem do "topo para a base" (*top-down*), que oferece um nível de abstração na visualização e análise dos componentes do sistema e de seus modos de falha. A ferramenta também provê a avaliação automática das Métricas de Arquitetura de Hardware dos *safety goals* modelados; a avaliação é impressa num arquivo do Microsoft Excel. A HWSafetyToolbox foca na reusabilidade de informação, pela criação de bibliotecas de informação de confiabilidade de componentes e permitindo ao usuário a importação de qualquer informação previamente modelada na ferramenta.

Este documento detalha o processo de desenvolvimento da ferramenta, com um diagrama de casos de uso, lista de requisitos e o meta-modelo que representa a informação do sistema de hardware. Adicionalmente, as tecnologias usadas no projeto e um diagrama mostrando a arquitetura do sistema são fornecidos, descrevendo o processo de implementação. Finalmente, é mostrada a modelagem de um sistema real de hardware na ferrramenta, com o objetivo de validar o protótipo desenvolvido.

Palavras-chave: Segurança funcional. ISO 26262. Métricas da Arquitetura de Hardware. Modelagem de hardware.

HWSafetyToolbox: A Tool for Modeling Safety-Related Hardware

ABSTRACT

The development of embedded systems for safety-critical application domains is on the rise. Since the safety requirements for these systems are high and difficult to integrate in the development process, standards were created providing a safety life-cycle to streamline the production of safety-critical systems. For the road vehicles application domain, the standard ISO 26262 was created, fulfilling the specific needs of the automotive industry to achieve functional safety on their electric/electronic equipments.

This thesis presents the design and implementation of the HWSafetyToolbox. The HWSafetyToolbox is a tool to help with the evaluation of Hardware Architecture Metrics, a process described on the phase of hardware development of the ISO 26262 standard. The tool provides a new way to model your system and its safety goals, the top-down approach, which offers a layer of abstraction for viewing and analyzing the components of the system and its faults. The tool also provides the automatic evaluation of Hardware Architecture Metrics for the modelled safety goals; the evaluation is done and printed in a Microsoft Excel file for the user. The HWSafetyToolbox focus on high reusability of information, featuring the ability to create component reliability information libraries and allowing the user to import into a new model any information previous modelled into the tool.

This document features the design process of the work developed, with a use case diagram, a list of requirements and the meta-model which represents the hardware system information. Additionally, the technologies used on the implementation and a diagram, showing the system architecture, are provided, describing the implementation process. Finally, it is shown the modelling of a real hardware system with the tool, for the validation of the prototype developed.

Keywords: Functional Safety. ISO 26262. Hardware Architecture Metrics. Hardware modeling.

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS AND ACRONYMS

AAL	Ambient Assisted Living
EMF	Eclipse Modeling Framework
IESE	Institute for Experiments in Software Engineering
ISO	International Organization for Standardization
UFRGS	Universidade Federal do Rio Grande do Sul
UML	Unified Modeling Language
WSN	Wireless Sensor Network
XMI	XML Metadata Interchange
XML	Extended Markup Language

SUMMARY

1 INTRODUCTION

This chapter presents an introduction to the project developed for this bachelor thesis: HWSafetyToolbox, a tool for modelling safety related hardware aiming compliance with the ISO 26262 standard. The model of the system is automatically analysed by the tool, the output of this analysis is the evaluation of the Hardware Architectural Metrics.

This work was developed while the author participated in an exchange program with the University of Kaiserslautern. There, the author joined the Microelectronic Systems Design Research Group, headed by Prof. Dr.-Ing. Norbert Wehn, working in this project, which was part of an ongoing partnership between the aforementioned group and the Embedded Systems Quality Assurance department of the Fraunhofer Institute for Experimental Software Engineering (IESE). This partnership occurs through the contribution of their expertise in microelectronics design and functional safety to research projects where these areas overlap.

The chapter starts with the motivations behind the creation of the project, then lists the project goals and is finished by the structure adopted by this thesis.

1.1 Motivation

The ever increasing presence of computer systems around us has been leading companies to push the boundaries on the design of systems for more dangerous application domains. When the application domain being analysed is of safety-critical applications like medicine equipment, oil extraction systems or road vehicles electronics, it should be considered that any malfunction related to these computer systems can be directly responsible for the harm of the persons and/or the environment (DUNN, 2003; ADLER et al., 2011). Therefore, the research for means which could allow the design of better computer systems for safety-critical applications, or safety-critical systems, is of great importance.

Aiming to provide a framework which could allow companies to achieve functional safety with their electrical, electronic and programmable electronic safety-related systems, the International Electrotechnical Commission (IEC) created the standard “IEC 61508 - Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES)” (INTERNATIONAL ELECTROTECHNICAL COMMISSION, 2011). An offspring of the IEC 61508 standard, focusing on the road vehicles domain, “ISO 26262 - Road vehicles - Functional safety”, was created by the International Organization for Standardization (ISO) through customization of the IEC 61508 standard to comply with needs

specific to the design of electronic systems used in road vehicles (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011). The process described as evaluation of the Hardware Architectural Metrics on "ISO 26262-5: —1 Road vehicles – Functional Safety — Part 5: Product development: hardware level" will be one of the main focuses of this thesis.

The "ISO 26262 - Part 5" specifies the requirements for the hardware level of the product development. Described on this part is the evaluation of the Hardware Architecture Metrics, a meticulous process that revolves around the exhaustive analysis of the impact of a possible failure of each component on each safety goal (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011). This evaluation provides a view on the system so up-close, that it becomes difficult to visualize the overall impact of a single component failure or of multiple components on the safety goal. Furthermore, to be able to do this evaluation, a bureaucratic and difficult work, the "Part 5" should be well read and understood by the engineer creating and/or analysing the hardware system. Additionally, when working with similar projects, all the analysis should be done twice, since the framework provided by the standard doesn't focus on reusability of information (KRAMMER et al., 2010).

Thereby it is important to create an easier approach for the modelling and safety analysis of hardware systems that seek compliance with the ISO26262. A tool which could provide a higher level of abstraction when modelling the system, with automatic calculation of the hardware metrics and information re-usability shall be of great utility.

1.2 Goals

The main goal of this work is to design and develop a tool - the HWSafetyToolbox - to help with the modelling and analysis of hardware in the context of the ISO26262 standard. The tool should provide an easy approach to the evaluation of the hardware architecture metrics as described in the "ISO 26262-5: —1 Road vehicles – Functional Safety — Part 5: Product development: hardware level".

To accomplish this, the tool shall present the following features:

- Hardware system modelling with top-down approach: the top-down approach will allow users to group components of the system into component blocks, providing a high-level view of the system. Component blocks will be able to

have their own failure modes, which may be connected to the failure mode of atomic blocks, creating a cause-effect relation between them.

- Component – Failure Mode library: creation of a library with reliability data of components. This will allow the easy import of information when the user is setting up a new component while modelling the system.
- Information reusability: all information created inside the tool will be able to be copy-pasted or drag-and-dropped inside another project. This will allow, for an example, the reusability of component blocks.
- Hardware architecture metrics: it will allow users to generate the table with the evaluation of their hardware architecture metrics.

1.3 Results

This thesis is presented as a documentation of the developed solution. The documentation contains information regarding the modelling and development of the tool: use case diagram; conceptual view of the system; EMF model, showing the classes of the system; Eclipse Plug-in conceptual view.

The prototype was evaluated with the modelling of a real hardware system, the AmICA wireless sensor node. It was defined component blocks for the whole system. Finally, it was modelled the relation between failure modes of the atomic components and components blocks for the power supply components of the hardware.

1.4 Organization

The organization of the rest of this document follows: Chapter 2 contains a review of concepts related to this work, definitions related to the safety area and an overview of the ISO26262 standard, focusing on the Part 5 and the Hardware Architecture Metrics evaluation process; closing Chapter 2, it is analysed the related work on this area. Chapter 3 presents the proposed solution, showing the architecture of the system, a brief explanation on the Eclipse Modelling Framework, as well as details on the implementation process. Chapter 4 presents the AmICA node, the hardware used in the test of the prototype, showing details of processing of modelling this hardware inside the tool. Chapter 5 describes the conclusions achieved though this project and the proposed future work.

2 LITERATURE REVIEW

This chapter provides the conceptual background necessary for the reader to be able to fully understand this work.

Here is presented an overview of safety related concepts, an introduction to the ISO 26262 standard and to its evaluation of hardware architecture metrics. Closing the chapter, Section 2.4 describes work related to this project, showing a commercial solution and also a related standard.

2.1 Safety Related Concepts

Avizienis (2004) defines *safety* as the "absence of catastrophic consequences on the user(s) and the environment". While this definition is widely adopted in the area of Fault-Tolerance systems research, the definition used in this work is slightly different and is provided by The International Organization for Standardization (2011a, p. 13, p. 17), to which *safety* is the "absence of unreasonable risk". Since this work focus on the "ISO 26262: —1 Road vehicles – Functional Safety", it is important to follow the concepts as presented in the standard's glossary. Therefore, the following definitions presented on this section will also be extracted from the text of the ISO 26262.

2.1.1 Unreasonable Risk

To be able to understand the concept of safety, it is necessary to know the meaning of *unreasonable risk*, which is the risk determined to be unacceptable in a certain context according to society morals (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011). The International Organization for Standardization (2011) defines also *risk* as the probability of harm occurring and its severity and *harm* as the physical part of injury to a person's health.

2.1.2 Functional Safety

Functional Safety is the lack of unreasonable risk related to hazards caused by faulty behaviour of electronic systems and *hazard* is defined as the probable source of harm (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011). The concept of

functional safety is the main goal of standards like the ISO 26262 and the IEC 61508, since the framework provided in these standards aim to help achieving this state of absence of unreasonable risk on the systems seeking compliance.

2.1.3 Error, Fault and Failure

During the operational phase of a product, events that deviates the system from their intended behavior may happen (AVIZIENIS et al., 2004; DUNN, 2003). Definitions provided by (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011) and related to the chain of events that bring this improper behavior are:

- Error - difference between value or condition on the system and the expected valued or condition that the system should provide;
- Fault - abnormal condition that may lead to a fail;
- Failure - when an element loses its ability to perform a function as specified.

The relationship between the provided definitions can be described as: the *fault* can lead to an *error* that can lead to a *failure*

2.1.4 Failure Mode

Failure mode is the way by which an element of the system fails. A quick example would be a resistor, such element can fail in two manners: open-circuit and short-circuit (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011).

2.1.5 Safety Mechanism

Safety Mechanism is a measure implemented on the system to detect or control failures, allowing an item to achieve and/or maintain a safe state. Another way that a safety mechanism can operate is by warning the car driver of the failure, expecting that the driver will control the effect of the failure by himself (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011).

2.1.6 Safety Goal

Safety Goal is the name given to the top-level safety requirement. It is established after the system passed through hazard analysis and risk assessment. These two techniques identify and categorize hazardous events of items (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011).

2.1.7 Fault Classification

In order to understand better the effects that hardware faults can have on a safety goal, it is important to establish a classification, according to (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011), for the several different types of fault:

- Single point fault: fault in an element that leads to the direct violation of the safety goal, this element not being covered by a safety mechanism;
- Residual fault: portion of a fault that causes a corruption of the safety goal, this portion of the fault is not covered by existing safety mechanisms related to the hardware element where the fault occurs.
- Multiple point fault: one fault out a group of merged independent faults, leading to a multiple point failure.
- A latent multiple point fault is multiple point fault which presence is neither detected by a safety mechanism nor perceived by the driver and leads to a violation of a safety goal.

2.1.6 Automotive Safety Integrity Level - ASIL

Automotive Safety Integrity Level (ASIL) is one of the main concepts created by the ISO 26262 standard. The International Organization for Standardization defines ASIL as four levels, A being the least rigorous and D the most rigorous, specifying the elements' safety requirements and measures to avoid unnecessary risk provided by the ISO 26262 (2011a).

2.2 ISO 26262

In 1985, the International Electrotechnical Commission (IEC) introduced the standard "IEC 61508 - Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES)" in its first version. The standard contains a framework for achieving functional safety in safety-critical systems. The standard has been, since its publication, widely accepted in the industry as the state-of-the-art of functional safety (PANESAR-WALAWEGE et al., 2010).

Understanding the fact that automotive industry has specific needs related to a safety standard, the International Organization for Standardization (ISO) published an offspring of the IEC 61508 standard: "ISO 26262 - Road vehicles - Functional safety". This new standard was created through several changes done to the IEC 61508, in order to comply with the state-of-the-art of the design of safety related electronic systems used in road vehicles (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011).

"ISO 26262 - Road vehicles - Functional safety" proposes a safety life cycle for road vehicles. Providing an extensive list of safety measures to achieve functional safety, the International Organization for Standardization (2011) is composed by 10 parts:

- *ISO 26262-1: —1 Road vehicles – Functional Safety — Part 1: Vocabulary*
  Specifies definitions and vocabulary used through the standard
- *ISO 26262-2: —1 Road vehicles – Functional Safety — Part 2: Management of functional safety*
  Specifies the requirements on functional safety management;
- *ISO 26262-3: —1 Road vehicles – Functional Safety — Part 3: Concept phase*
  Specifies requirements on the concept phase of development. These include the hazard analysis and risk assessment, item definition, the initiation of the safety lifecycle, and the functional safety concept.
- *ISO 26262-4: —1 Road vehicles – Functional Safety — Part 4: Product development: system level*
  Specifies requirements for system level product development. Some of the requirements are related to: technical safety concept, system design, item integration and testing, safety validation and functional safety assessment.
- *ISO 26262-5: —1 Road vehicles – Functional Safety — Part 5: Product development: hardware level*

Specifies requirements for product development of hardware. Some of these requirements are: specification of the hardware safety requirements, hardware design, hardware architectural metrics, and evaluation of violation of the safety goal due to random hardware failures

- *ISO 26262-6: —1 Road vehicles – Functional Safety — Part 6: Product development: software level*

  Specifies requirements for software development. These include requirements for the: specification of software safety requirements, software architectural design, software unit design and implementation, software unit testing
- *ISO 26262-7: —1 Road vehicles – Functional Safety — Part 7: Production and operation*

  Specifies requirements for product production and operation. These include requirements for the: production, operation, service and decommissioning.
- *ISO 26262-8: —1 Road vehicles – Functional Safety — Part 8: Supporting processes*

  Specifies requirements for the process of product support.
- *ISO 26262-9: —1 Road vehicles – Functional Safety — Part 9: ASIL-oriented and safety-oriented analyses*

  Specifies the requirements for ASIL-oriented and safety-oriented analyses. Some of the requirements are for ASIL decomposition, analysis of dependent failures, criteria for coexistence of elements of different ASIL and safety analyses.

Through the above listed parts, the standard aims to provide a safety lifecycle for automotive systems, determining activities to be performed during all the steps of this lifecycle. By the usage of ASILs, it provides a risk-based determination of classes of risk in a system, proposing safety requirements according to the specified ASIL, with goal to avoid unacceptable risk (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011). Moreover, by providing these requirements and specifications, the standard provides, for the compliant systems, a way to achieve functional safety.

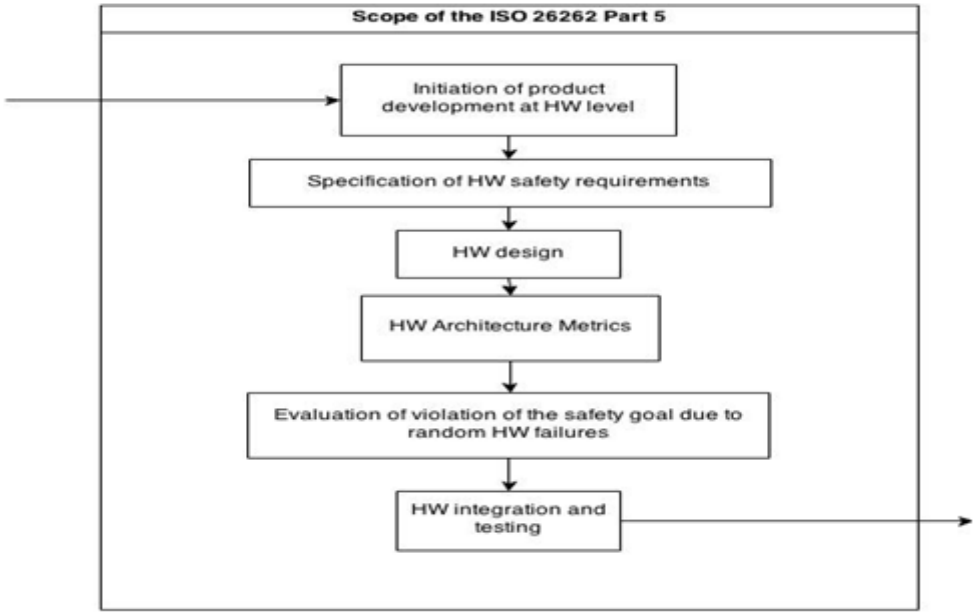
The determination of Automotive Safety Integrity Levels is done during the concept phase of the safety life-cycle. For this, the part is evaluated with regard to its functional safety, through hazard analysis and risk assessment. The output is the hazard situations present on the item, while the *Safety Goals* and their assigned *Automotive Safety Integrity Level* (ASIL) are determined by evaluation of these hazardous situations. Severity, probability of exposure and controllability are taken in consideration during ASIL determination. (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011). The safety measures proposed through the ISO 26262 standard are distributed with different layers of robustness to match the A, B, C and D ASILs.

The further exploration of the standard is beyond the scope of this work, hence, from now on, it will be here considered just the part 5, related to development on hardware level and the hardware architecture metrics.

2.2.2 ISO 26262: Part 5: Product development: hardware level.

The hardware development phase, illustrated on the Figure 2.2, starts with the "Initiation of the product development for the hardware", that seeks to determine and plan the functional safety activities during the sub-processes of the hardware development. After, begins the "Specification of hardware safety requirements", when it should be done a fully complete hardware specification that will be used on the development of the considered item. The requirements obtained are considered hardware safety requirements. Then, the sub-process of "Hardware design" starts, the objectives of this process are to design the hardware strictly following the specification and the hardware safety requirements, after the design is done, it should be done a verification of this design with respect to the specification and safety requirements. After the "Hardware design" is finished, the "Hardware architectural metrics" starts, the objective of this clause is to evaluate the hardware architecture of the hardware item against the safety requirements by the hardware architectural metrics. In the next sub-section of this work, 2.3.1, this clause will be discussed with greater detail. Next, begins the "Evaluation of violation of the safety goal due to random HW failures", which aims to evaluate if is sufficiently low the risk of a random hardware failure causing a violation on the safety goal violation. Finally, the last process is the "Hardware integration and testing", where the objective is to test the item to verify if it complies with the safety requirements created at the beginning. (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011)

Figure 2.2 Overview of the "Product Development: Hardware Level"



Source: (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011c)

2.2.3 Hardware Architecture Metrics of the ISO 26262 standard

The International Organization for Standardization establishes Hardware Architecture Metrics as metrics for the evaluating the robustness of the hardware architecture design with respect to safety (2011b), being described in the clause 8 and in the Annex C of the "Part 5: Product development: hardware level" of the ISO 26262 standard. These metrics should be applied to *every safety goals with related ASIL of C or D*, being optional, but recommend, to ASIL B.

The Hardware Architecture Metrics is the formed by the *single fault metric* and the *latent fault metric*, these two metrics represent the robustness of the hardware with respect to residual and single point faults (single fault metric) and to latent multiple point faults (latent fault metric) that may affect a safety goal. The robustness is shown by analyzing the failure modes of the hardware elements related to a safety goal and assessing the safety impact that they may have on the system, i.e. if these failure modes can cause the corruption of the safety goal.

The objectives of doing such evaluation are as follows:

- Reveal the robustness of the system to single point faults and to latent faults that may affect the safety goal;
- Reveal if it is sufficient the coverage of the safety mechanisms, to control hardware faults and to prevent risk from latent point faults in the E/E architecture; (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011c)

The target values for the hardware architecture metrics are represented in the Table 2.2.1

Table 2.2.1 Hardware Architecture Metrics' target values

	ASIL B	**ASIL C**	**ASIL D**
Single points fault metric	> 90 %	> 97 %	> 99 %
Latent points fault metric	> 60 %	> 80 %	> 90 %

Source: (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011)

The process of evaluating the hardware architecture metrics can be separated into stages:

1. It should be gathered a list of all hardware components related to the safety goal under evaluation.
2. With the components list, this step and the ones that follow should be repeated for each safety-related component. It should be gathered information related to the *Failure Modes, Failure Rate (in FIT values) and how the failure rate is divided (Failure Rate distribution, in percentage)* between the Failure Modes of the component. The information should be retrieved from a recognised industry source, such as the IEC 62380 or the Siemens norm SN 29500.
3. It should be evaluated if the *Failure Modes* of the component have the potential to violate the safety goal in absence of a *Safety Mechanisms*. If positive, then steps 5 and 6 should be performed
4. The *Safety Mechanisms* coverage with respect to the violation of the *Safety Goal* shall be evaluated, for each *Failure Modes* that had a positive the evaluation in

the 6th stage. The coverage value, in percentage, should be taken from a trusted industrial source, or by using the process described in the Annex D of the ISO 26262 Part 5. If there is no safety mechanism related to this failure mode, the coverage is considered as being 0%.

5. With the information and evaluation above, the analyst will be able to calculate the *Residual or Single Point Fault failure rate*. The calculation is done by the formula = (failure rate) * (failure rate distribution) * (1 - the safety mechanism coverage).

6. It should be evaluated if the *Failure Modes* may lead to the violation of safety goal in combination with an independent failure of another component.

7. The *Safety Mechanisms* coverage with respect to latent failures should be also evaluated, for each one the *Failure Mode* that had a positive the evaluation in the 6th stage. The coverage value, in percentage, should be taken from a trusted industrial source, or by using the process described in the Annex D of the ISO 26262 Part 5. If there is no safety mechanism related to this failure mode, the coverage is considered as being 0%.

8. With the information and evaluation above, the analyst will be able to calculate the *Latent Multiple Point Fault failure rate*. The calculation is done by the formula = (failure rate) * (failure rate distribution) * (1 - safety mechanism coverage with respect to latent failures).

Having completed the steps above for all the components related to a safety goal, the evaluation of the Hardware Architecture Metrics can be performed. The information gathered previously, is used in the following formulas:

$$SinglePointsFaultMetric = 1 - \frac{\sum_{SR}(\lambda SPF + \lambda RF)}{\sum_{SR}\lambda}$$

$$= \frac{\sum_{SR}(\lambda MPF + \lambda S)}{\sum_{SR}\lambda}$$

$$LatentFaultMetric = 1 - \frac{\sum_{SR}(\lambda MPFL)}{\sum_{SR}(\lambda - \lambda SPF - \lambda RF)}$$

$$= \frac{\sum_{SR}(\lambda MPFPoD + \lambda S)}{\sum_{SR}(\lambda - \lambda SPF - \lambda RF)}$$

Where:

λ: failure rate;

SR: safety related hardware components;

λSPF: failure rate associated to hardware component single point faults;

λRF: failure rate associated to hardware component residual faults;

λMPF: failure rate associated to hardware component multiple point faults;

λMPFPoD: failure rate associated to hardware component perceived or detected multiple point faults;

λMPFL: failure rate associated to hardware element latent multiple point faults

Finally, after the evaluation is done, the output of the equations should be compared with the target values (presented on the Table 2.2.1). If the result is positive, it means that the hardware design is robust enough to satisfy the safety goal. Otherwise, the hardware architecture should undergo structural changes. (INTERNATIONAL ORGANIZATION FOR STANDARDIZATION, 2011c)

Since the evaluation of the Hardware Architecture Metrics is a complex process, the Figure 2.2.1 and 2.2.2 shows an example where the steps from 1 to 8 are performed in a hypothetic safety goal of an hypothetic system and presented on a table.

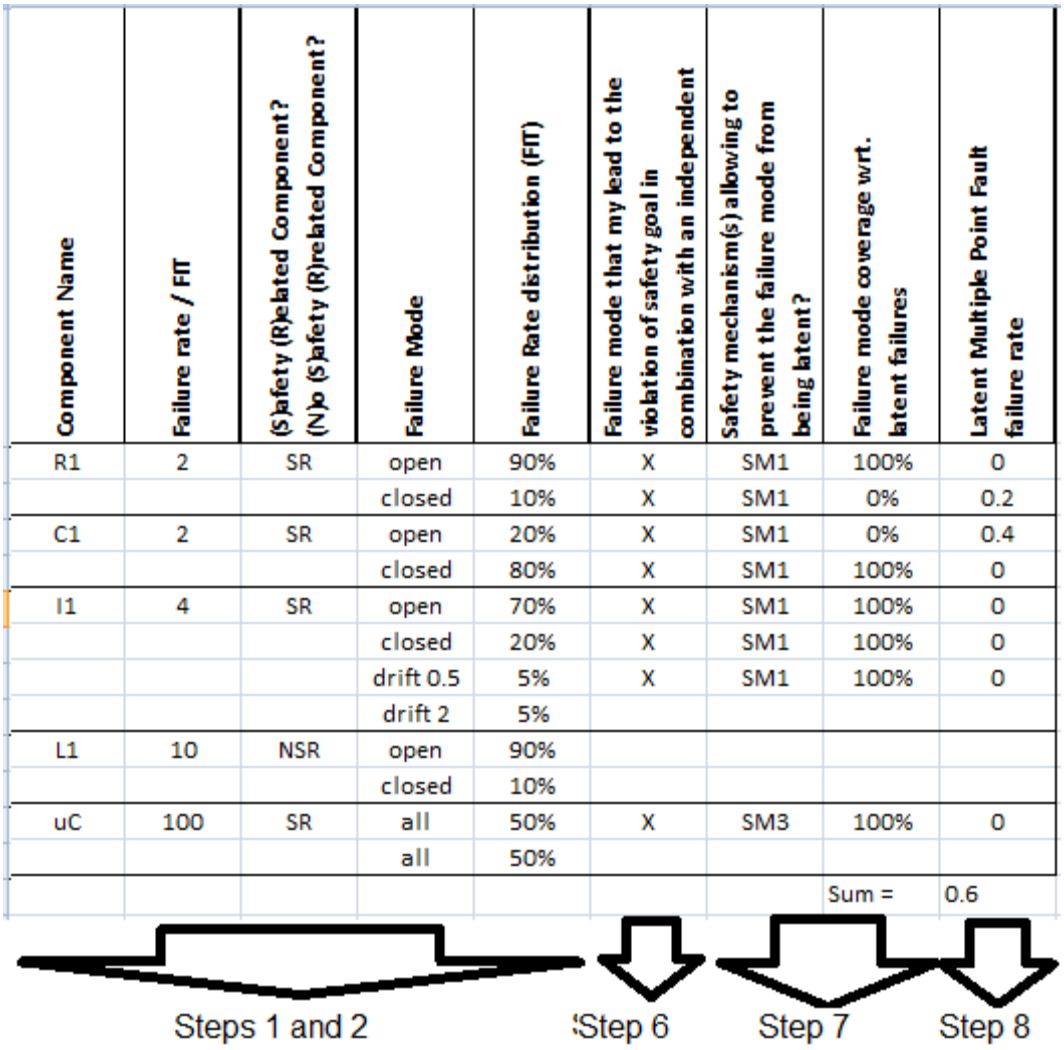
Figure 2.2.1 Example of the steps 1 to 5 performed in a hypothetic system, these steps are related to the evaluation of the residual or single point fault failure rate

Component Name	Failure rate / FIT	(S)afety (R)elated Component? (N)o (S)afety (R)elated Component?	Failure Mode	Failure Rate distribution (FIT)	Failure mode that has the potential to violate the safety goal in absence of Safety Mechanisms?	Safety mechanism(s) allowing to prevent the failure mode from violating the safety goal?	Failure mode coverage w/r. violation of safety goal	Residual or Single Point Fault failure rate
R1	2	SR	open	90%	X	none	90%	0.18
			closed	10%				
C1	2	SR	open	20%				
			closed	80%	X	SM1	99%	0.016
I1	4	SR	open	70%	X	SM1	99%	0.028
			closed	20%	X	SM1	99%	0.008
			drift 0.5	5%	X	SM1	99%	0.002
			drift 2	5%				
L1	10	NSR	open	90%				
			closed	10%				
uC	100	SR	all	50%	X	SM3	90%	5
			all	50%		SM3		
							Sum =	5.234

Steps 1 and 2 — Step 3 — Step 4 — Step 5

Source: (JEON et al., 2011)

Figure 2.2.2 Example of the steps 1, 2, 6, 7 and 8 performed in a hypothetic system, these steps are related to the evaluation of the residual or single point



Component Name	Failure rate / FIT	(S)afety (R)elated Component? (N)o (S)afety (R)elated Component?	Failure Mode	Failure Rate distribution (FIT)	Failure mode that may lead to the violation of safety goal in combination with an independent	Safety mechanism(s) allowing to prevent the failure mode from being latent?	Failure mode coverage wrt. latent failures	Latent Multiple Point Fault failure rate
R1	2	SR	open	90%	X	SM1	100%	0
			closed	10%	X	SM1	0%	0.2
C1	2	SR	open	20%	X	SM1	0%	0.4
			closed	80%	X	SM1	100%	0
I1	4	SR	open	70%	X	SM1	100%	0
			closed	20%	X	SM1	100%	0
			drift 0.5	5%	X	SM1	100%	0
			drift 2	5%				
L1	10	NSR	open	90%				
			closed	10%				
uC	100	SR	all	50%	X	SM3	100%	0
			all	50%				
							Sum =	0.6

Source: (JEON et al., 2011)

With the above example, we can calculate the Single Point Fault Metric and Latent Fault Metric. The sum of the failure rate of the safety related hardware components is 108. The Single Point Fault Metric = 1 – (5.234/108) = 95,2%. The Latent Fault Metric = 1 – (0.6 / (108 – 5.234)) = 99.42%. This means that if the safety goal being evaluated had related ASIL B, it would satisfy the target values. Since it is a conservative estimation, the target value of ASIL D achieved by the Latent Fault Metric is discarded, being used the lower estimative achieved by the Single Point Fault Metric. (JEON et al., 2011)

2.4 Related Work

Considering that the ISO 26262 standard was first published in 2011, there is already a considerable amount of study and research done on top of the norm. Some examples are (BORN; FAVARO; KATH, 2010), (JEON et al., 2011), (KRAMMER et al., 2010), (PALIN et al., 2011) and (STIRGWOLT, 2013).

After a lengthy research was done, no proposal was found with respect to turning the hardware architecture metrics evaluation a more straightforward process or to providing a layer of abstraction for it. Consequently, instead of presenting directly related work, this section will focus on two more broadly related works. Firstly, "medini analyze", a commercial application that helps on the hardware development phase. Finally, it will be provided an overview on the "Safe Automotive soFtware architEcture (SAFE) - WP3 - Deliverable D3.2.2", a model based approach for hardware systems seeking compliance with the ISO 26262 standard.

2.4.1 Medini Analyze

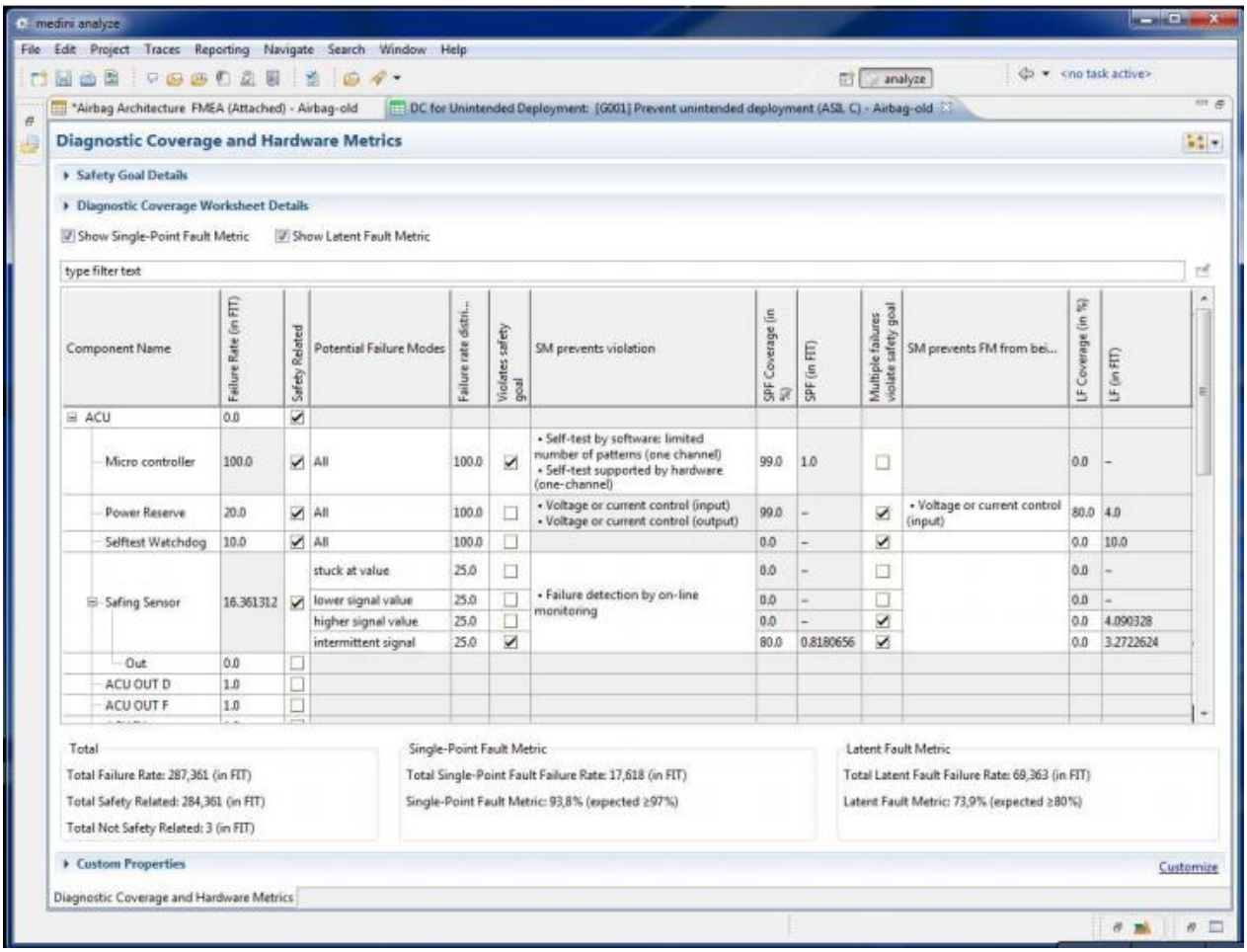
"medini™ analyze" is a tool developed and distributed by the German company ikv++ technologies ag. The application focus is the core activities of the functional safety analysis according to the ISO 26262 standard. The user base of the tool is mainly formed by safety managers, development engineers and quality managers that work with projects of safety-critial automotive systems aiming compliance with the ISO 26262 standard (IKV++ TECHNOLOGIES AG, [s.d.]a).

According to IKV++ TECHNOLOGIES AG, the main features for the hardware metrics part of the norm are as follows:

> "calculation of Single Point Fault Metric (SPF) and Latent Fault Metric (LF); Safety Element out of Context support evaluation of HW metrics ; automatic synchronization of failure mode and failure rate data from architecture model; specification of cause/effect chains and automatic calculation of failure rates; extensible catalog of safety mechanisms according to part 5 of ISO 26262; default SPF/LF coverage for safety mechanisms; rich validation and consistency checks; traceability of safety mechanisms to requirements and SW/HW implementation" ([s,d]b)

By the main features listed above, it is shown that a highly complete and efficient solution for the hardware architecture metrics (called failure metrics there) is present on the tool. The tool and this project have overlapping characteristics: the automatization of the evaluation of the metrics and information catalogue (focusing on re-usability of information). Since no evaluation version is provided for students, there will not be a more in deep comparison. The Figure 2.4.1 shows a sample screen of the tool.

Figure 2.4.1 Screencapture of “midina analyze"



Source: (IKV++ TECHNOLOGIES AG, [s.d.]b, p. 3)

2.4.2 Safe Automotive soFtware architEcture (SAFE) - WP3 - Deliverable D3.2.2

The Safe Automotive soFtware architEcture (SAFE) project is a model based solution for safe automotive applications to show compliance with the ISO 26262 standard (ITEA3.ORG, [s.d.]). This solution proposes several different models for covering all the

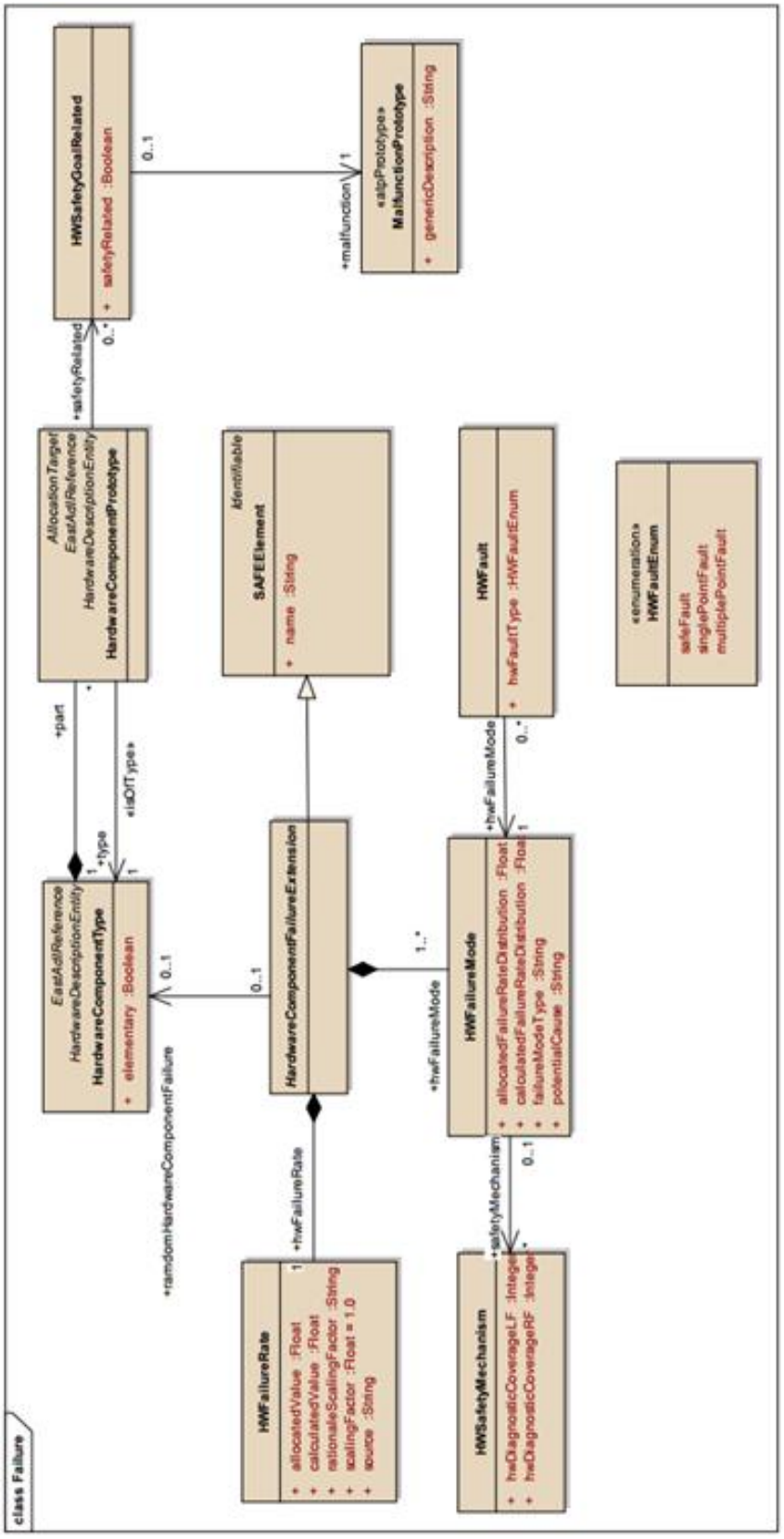
aspects of the safety life cycle. Due to the scope of this thesis, the only part to be analyzed will the the “WP3 - Deliverable D3.2.2 - Proposal for extension of Meta model for hardware”.

The *SAFE - WP3 - Deliverable D3.2.2* solution is of high complexity. The (ITEA3.ORG, 2013) defines meta-models for the modelling of several different aspect of safety-related hardware seeking compliance with the ISO 26262 standard, including the relationship between hardware components and their faults and failures.

The main relation between the work of this thesis and the *SAFE – WP3* is that both offer metal-models capable of representing the information related to the Hardware Architecture Metrics. Another similarity is that the *SAFE – WP3* offers an abstraction layer for the representation of hardware, allowing a more functional view (ITEA3.ORG, 2013).

The Figure 2.4.2 shows the meta-model for the relation between the *Safety Goal, Component, Failure Mode, Failure Rate and Safety Mechanism.*

Figure 2.4.2 Class diagram meta-model for the relation between the *Safety Goal, Component, Failure Mode, Failure Rate and Safety Mechanism.*



Source: (ITEA3.ORG, 2013, p. 63)

3 DESIGN AND IMPLEMENTATION

This chapter describes the design and implementation process of the HWSafetyToolbox. The HWSafetyToolbox is a tool to help with the evaluation of Hardware Architecture Metrics. The tool provides a top-down approach to model your system and its safety goals, offering a layer of abstraction for viewing and analyzing the components of the system and its faults. The tool performs the automatic evaluation of Hardware Architecture Metrics for the modelled safety goals; the evaluation is done and printed in a Microsoft Excel file for the user. The HWSafetyToolbox focus on high reusability of information. The chapter opens with a description of the applied methodology, then discusses details of the project, as the requirements, use case diagrams and system architecture and it is closed by the finals considerations regarding the implementation.

3.1 Project Background

This work was developed as part of an ongoing partnership between the Microelectronic Systems Design Research Group of the Technische Universität Kaiserslautern and the Embedded Systems Quality Assurance department of the Fraunhofer Institute for Experimental Software Engineering (IESE). This partnership occurs through the contribution of their expertise in microelectronics design and functional safety to research projects where these areas overlap each other.

The Microelectronic Systems Design Research Group, headed by Prof. Dr.-Ing. Norbert Wehn, focus on design methodologies and architectures for microelectronic systems. This project was developed under the direct guidance of Dipl.-Ing. Matthias Jung and Dipl.-Ing. Sebastian Wille, members of the PhDs team of the Microelectronic Systems Design Research Group, and the general supervision of Prof. Dr.-Ing. Norbert Wehn.

The Embedded Systems Quality Assurance department is headed by Sören Kemmann and it is part of the Fraunhofer IESE, one of the leading institutions in the functional safety area. From the Fraunhofer IESE team, the contributors to this work were Dipl.-Inf Bastian Zimmer and Sören Kemmann, as direct advisors.

3.2 Methodology

The methodology adopted on the project is summarized as follows:

1. The first step was to study and understand the basic concepts around the ISO 26262 standard. After that, a more deep study was done regarding the Part 5 of the standard.
2. The second step, done during several meetings with all the advisors, was to discuss the proposal and define the main characteristics of the meta-model that would be the basis of the tool developed.
3. It was discussed the best technologies available to be used on the development.
4. It was draw the first version of the meta-model. After, this first version of the meta-model was tested to evaluate if the meta-model correctly represented, at least, all the information related to the calculation of the *Hardware Architecture Metrics* of the modelled *safety goals* of the target system.
5. After that, the meta-model was further specified, as well the requirements of the tool.
6. Weekly meetings were held to review the ongoing development of the tool and to discuss the last requirements of the tool.
7. Finally, when the HWSafetyToolbox was with the main features already implemented, it was used to model and analyze a real safety-related system: the AmICA sensor node.

3.3 Project

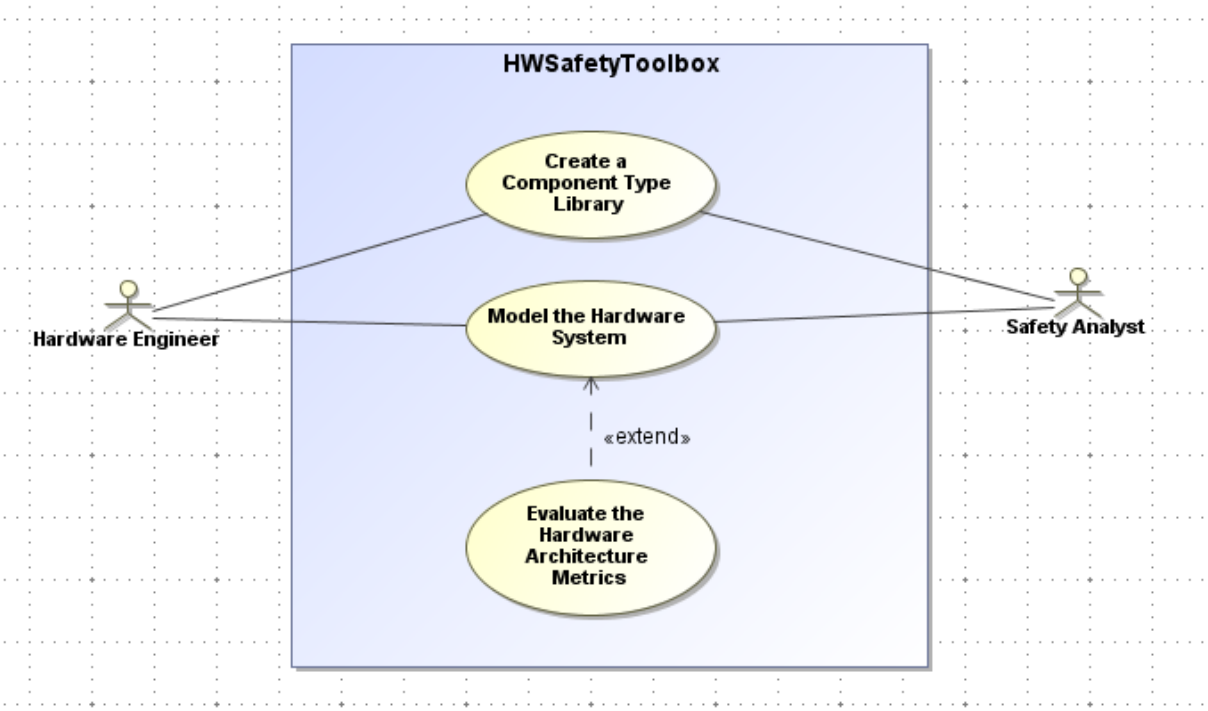
Here it will be described the main aspects of the solution proposed, the HWSafetyToolbox. Several different tools were used in the creation of the diagrams here presented. The Use Case diagram was created using "Magic Draw". The meta-model was created using the Ecore Graphical Editor of the Eclipse Modeling Framework in Eclipse. The diagram representing the system architecture was done on the draw.io website (www.draw.io).

3.3.1 Use Cases

Since Use Cases provide a good overview of the main usage scenarios of the project, it was created a Use Case Diagram to illustrate them. There are two actors, *Hardware Engineer*, that would be the person who designed the hardware under analysis, *and Safety Analyst*, that would be the person which solely objective is to analyse the hardware, they represent the

users of our tool and it is considered that they are able to perform all of the use cases proposed. The use cases performed by them are the creation of component libraries and the modelling of the hardware system, which was extended to the evaluation of the *Hardware Architecture Metrics.* The figure 3.2.1 shows the use case diagram.

Figure 3.3.1 Use Case diagram



Source: (Author)

3.3.1 Requirements

The main requirements were defined at the beginning of the planning phase of the project. The solution proposed should present the following basic features:

- The creation of component type library, with information regarding the reliability of electronic and electric hardware components. The library should have information related to the Failure Rate, Failure Modes and Failure Rate Distribution of a type of component. When creating a model, by defining that a component is of a type present in the library, the system should automatically copy the information of the Failure Rate, Failure Modes and Failure Rate distribution to the component being modeled;
- The modelling of safety-related hardware systems. The basic system modelling approach is known as bottom-up approach. This includes: the modelling of the

E/E hardware components and their *failure modes*; the modelling of the *safety mechanism* used on the system; the modelling of the *safety goals* of the system, by the modelling of what, here in this project, was called the *Failure Relation*. Failure Relation is the *residual or single point fault failure rate* and the *latent multiple point fault failure rate* of the *failure modes* of the components, taking into consideration the coverage of possibly related *safety mechanism(s)*;

- The evaluation of the *Hardware Architecture Metrics* of the modelled system.

During the planning phase of the project and the weekly meetings, it was defined requirements that further define the features of the tool. They are as follows:

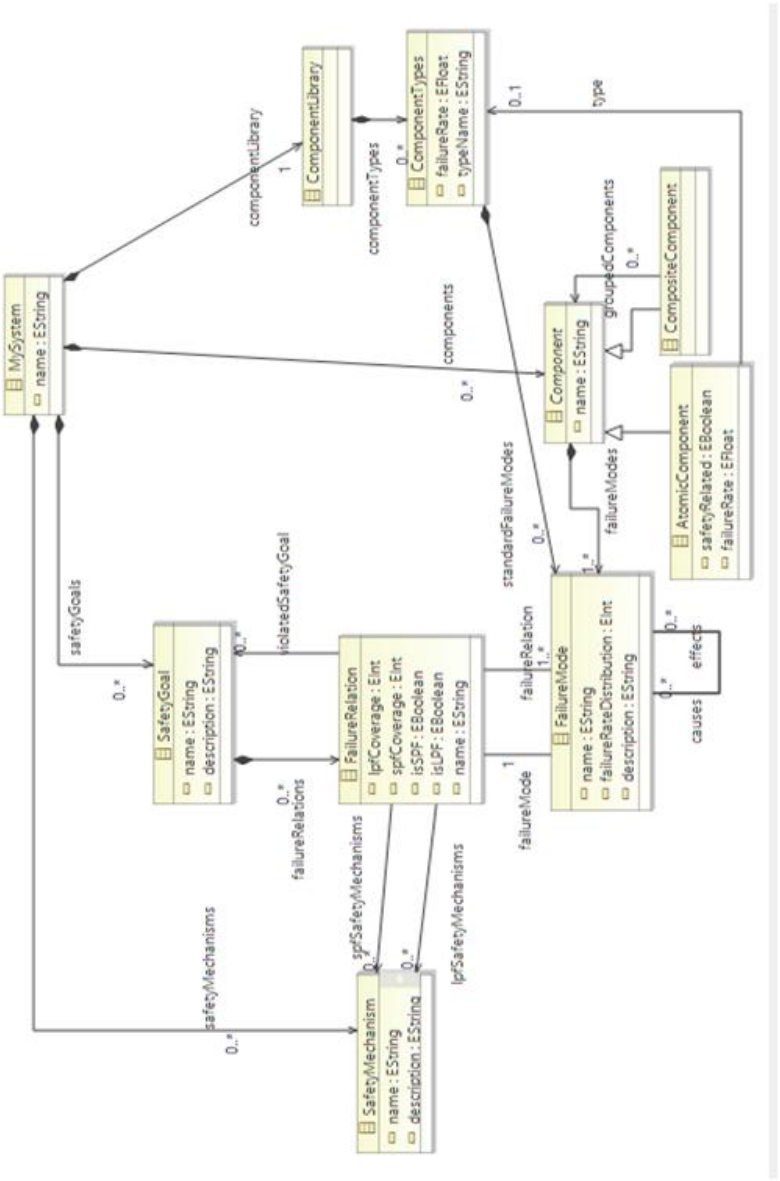
- The modelling of the hardware with a top-down approach. The user shall be able to create a high-level view of the system modelled. The top-down approach will allow users to group atomic components of the system into component blocks. It is also allowed the creation of component blocks formed by component blocks, adding another layer of abstraction.
- The modelling of failure modes of component blocks.
- The modelling of cause-effect relationship between *failure modes* of different layers of abstraction. The user shall be able to connect failure modes of atomic components to the failure modes of component blocks, or vice-versa, creating a cause-effect relation between them.
- The down propagation of *safety mechanisms*. If the user specifies that a safety mechanism handles a failure mode of a component block, the tool automatically adds the safety mechanism to all failure modes that causes the failure mode of that component block, with the same coverage.
- The reusability of information. All the parts of the model shall be reusable. The user shall be able to be “copy and paste” and “drag and drop” every part of the model.
- The evaluation of the Hardware Architecture Metrics shall be printed in the form of a table to an Excel file for the user.

3.3.3 The Meta-Model

One of the most critical steps on this project was the creation of the meta-model for expressing the hardware system and its safety-related information. The meta-model was first

sketched and discussed and then recreated using the Ecore Graphical Editor in Eclipse. The Ecore Graphical Editor allows the creation of diagrams that are a subset of a UML class diagram. The Figure 3.3.2 shows the meta-model.

Figure 3.3.2 The meta-model for represention of the modeled system information



One important aspect of the meta-model is the definition of the atomic components and block of components. To allow the HWSafetyToolbox to see group of components the same way it sees atomic components, it was used the composite pattern. Composite pattern is a design pattern that allows a system to treat atomic elements the same way that it treats composite elements. The usage of the pattern can be seen on the lower right corner of the Figure 3.3.2

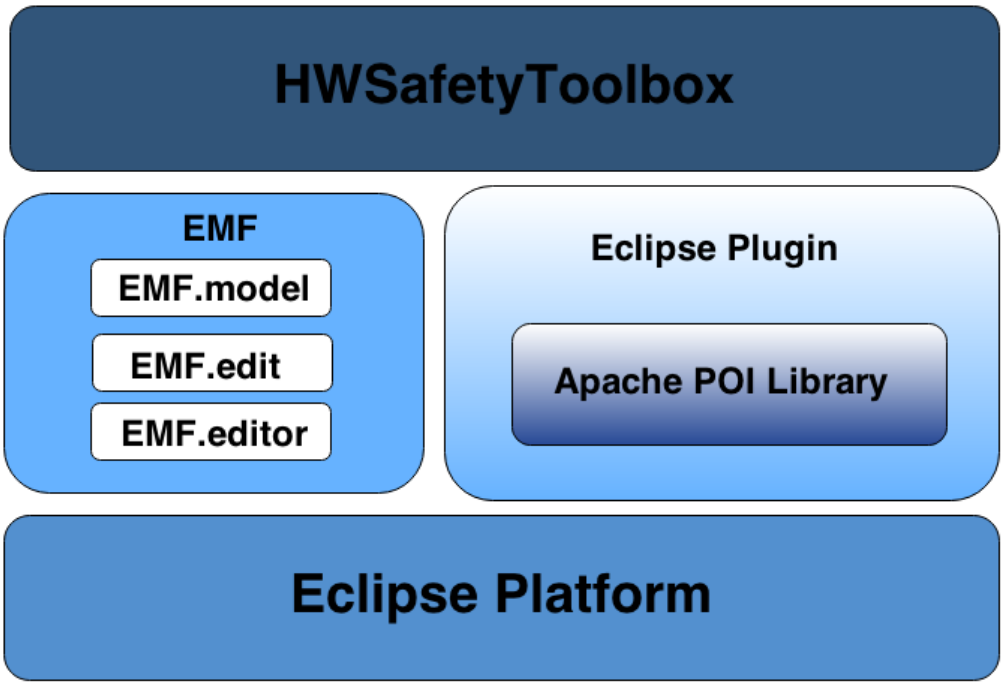
Another important aspect of the meta-model is the *Failure Relation*, a class for representing information regarding the *residual or single point fault failure rate* and the *latent multiple point fault failure rate* of the failure modes of the components, taking into consideration the coverage of possibly related safety mechanism(s).

The concept of *Safety Goal(s)* direct connection to *Failure Relations* is also important. It would be a mistake to consider that a Safety Goal is formed by components, or by failure modes, while the reality is that component and failure mode information can be shared by several safety goals. The information that is related to only one safety goal is exactly the *Failure Relation*, just by understanding this that we are able to easily reuse information and to model the system with top-down approach.

3.3.4 System Architecture

The HWSafetyToolbox was developed using Java and the Eclipse Platform. Here will be described in detail the architecture of the system, giving an overview of the technologies used. The Figure 3.3.3 shows a block diagram view on the system architecture.

Figure 3.3.3 System Architecture



Source: (Author)

Using the diagram of the meta-model as input, the Eclipse Modelling Framework automatically generated the Java code for the classes described on the diagram. This part of the EMF solution is called the EMF.model.

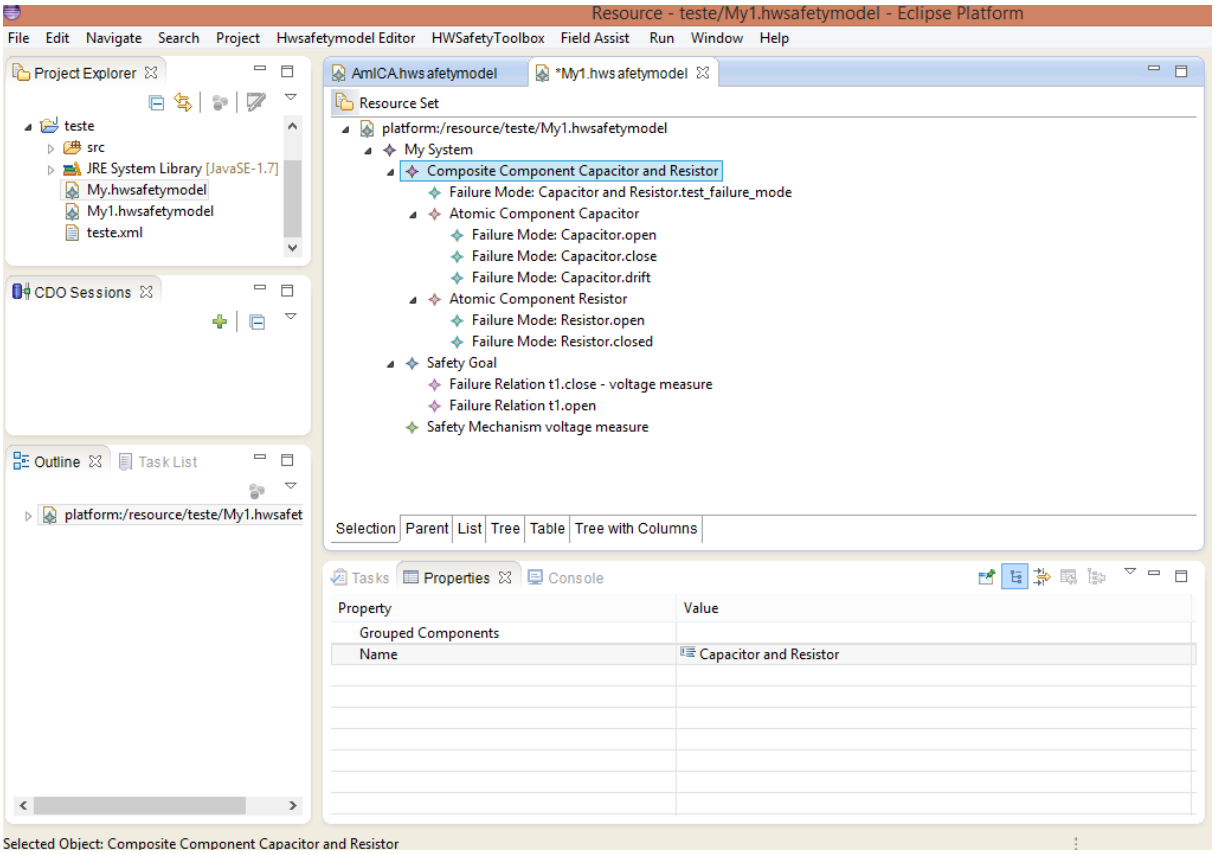
After this, EMF generated the code for doing operations with our model. This part is called the EMF.edit, and it provides code for the creation of instances of the model and for editing these instances, it also provides code for a tree-structured visualization of the model. Since the operations provided are just basic ones, the code here generated had to be heavily edited on this project to be able to complete all the requirements.

Finally, EMF generates what it calls the EMF.editor. EMF.editor is a simple Graphic User Interface for the EMF.edit, running on top of a heavily modified Eclipse IDE. It allows you to create, visualize and edit instances of your model in a tree-view structure. The EMF.editor, even if simplistic, offers most of the basic functionality that is built-in on the Eclipse IDE, resulting in a robust editor for the model created. Examples of this functionality are the support for undo and redo, copy and paste and drag and drop. The instances of the model created using the editor are saved in a XML Metadata Interchange (XMI) file, an important characteristic, since it allows any other program to parse and edit information of your model. The Annex B of this work presents a copy of an XMI file.

To be able to do the evaluation of the Hardware Architecture Metrics and to generate the Excel file, it was developed an Eclipse Plug-in that would be attached to the EMF.editor mentioned above. This plug-in makes use of the Apache POI library, a library for the creation of Microsoft Office related files.

Summarizing the system architecture, the HWSafetyToolbox runs on top of a highly customized version of the Eclipse Platform, provided by EMF. Attached to this Eclipse IDE, it is the plug-in responsible for generating the *Hardware Architecture Metrics* table in an Excel file. The figure 3.3.4 present the HWSafetyToolbox running with a simple system modelled in it, also showing the tree structure to visualize the system modelled. The simple system modelled is composed by one capacitor and one resistor, grouped in a sample composite component called “Capacitor and Resistor”. The tree structure provided has as root the system, with the children being the Components (or Composite Components), Safety Goals and Safety Mechanisms. The components have as children other components and failure modes. The Safety Goal has as children the failure relations.

Figure 3.3.4 : HWSafetyToolbox sample screen



Source: (Author)

3.4 Final Considerations

During the implementation part of this project, there were some obstacles faced. The first one was the shift from Papyrus to Ecore/EMF for the creation of the meta-model. Papyrus lacked the documentation necessary to allow a comfortable usage of the environment. After that, most of the obstacles faced were related to the lack of experience with the EMF and editing EMF generated code and also with the programming of Eclipse Plug-ins.

After these difficulties were surpassed, the development went on without remarkable problems.

4 TEST OF THE PROTOTYPE

This chapter presents the last phase of this project, the test of the prototype. To do a demonstration of the usage of the HWSafetyToolbox, it was modelled an AmICA sensor node using the tool. The chapter starts with a presentation of the AmICA platform, and then provides the board diagrams, the modelling and its results.

4.1 AmICA Platform

AmICA is a flexible, compact, easy-to-program, and low-power Wireless Sensor Node (WSN) platform. An AmICA node is approximately the size of a 0,25 Euro coin and can be used for Ambient Assisted Living (AAL) applications (S. WILLE et al., 2010). The figure 4.1.1 shows an AmICA node.

Figure 4.1.1: Top and bottom view of an AmICA node

Source: (S. WILLE et al., 2010)

AmICA was chosen for this demonstration mainly for three reasons:

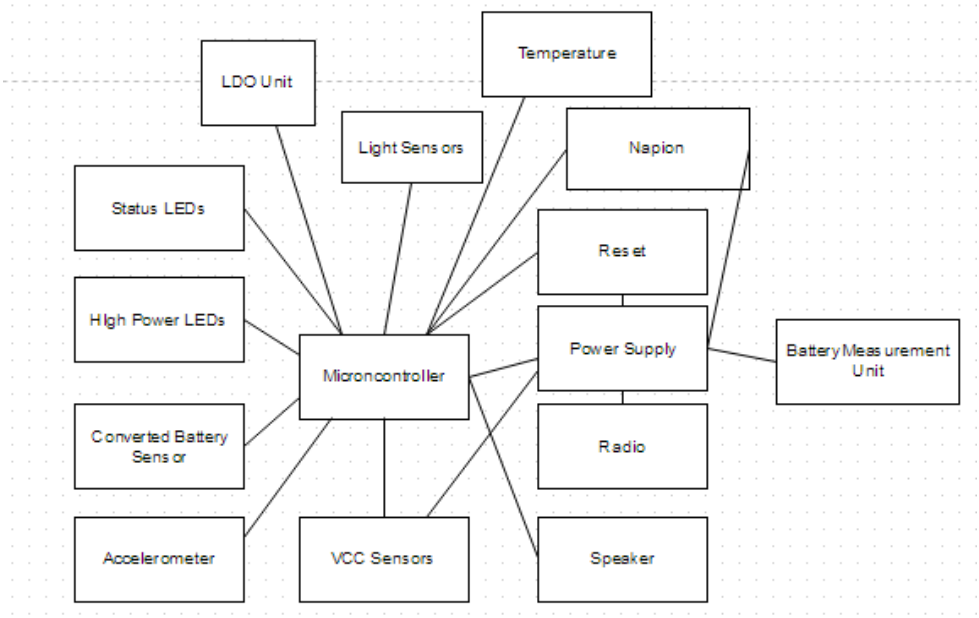
- The person responsible for designing the AmICA hardware was Dipl.-Ing Sebastian Wille, one of the advisors of this project. It is important to have someone with deep knowledge of the hardware during the modelling, making easier the process of evaluating possible faults.
- The AmICA node is formed by a small list of components.
- Even if not developed aiming functional safety or compliance with any safety related standard, AmICA can be considered a safety related system if provided with a safety critical AAL application.

Therefore, even if not a part of road vehicle hardware, it was the best safety related system available for evaluating the tool. For the purpose of this demonstration, it was considered that the AmICA node was being used in an AAL application, where it should detect if a person in a given room is moving or not and send this information to a server. If the person is not moving, it could indicate that the subject had a health problem and needed medical help.

4.1.1 Grouped Components

The block diagram of AmICA provides an overview of main components of the system and the connection between them. The block diagram can be found on the Annex B of this thesis. Using the block diagram as starting point, it was created several groups of components on the AmICA node. This added layer of abstraction provides a more functional view of the hardware elements. Figures 4.1.2 present the result of the creation of composite components. Between these groups it is important to highlight the Napion group, which the main atomic component is the Napion infrared sensor, one of the most important components of the AmICA sensor, since it provides human detection information. The Napion sensor alone has a failure rate (FIT) of 50, having high impact on the safety analysis. Another composite component important to highlight is the microcontroller group, formed by another composite components, containing parts like the ATMEGA324PV-10AU microntroller, the crystal and the ADC parts. The ATMEGA324PV-10AU also has a high impact on the system safety, since it has the failure rate of 100, but at least it contains an internal watchdog, a safety mechanism with 90% coverage of failure modes with respect to single point faults.

Figure 4.1.2 Grouped components of the AmICA node



4.2 AmICA Model

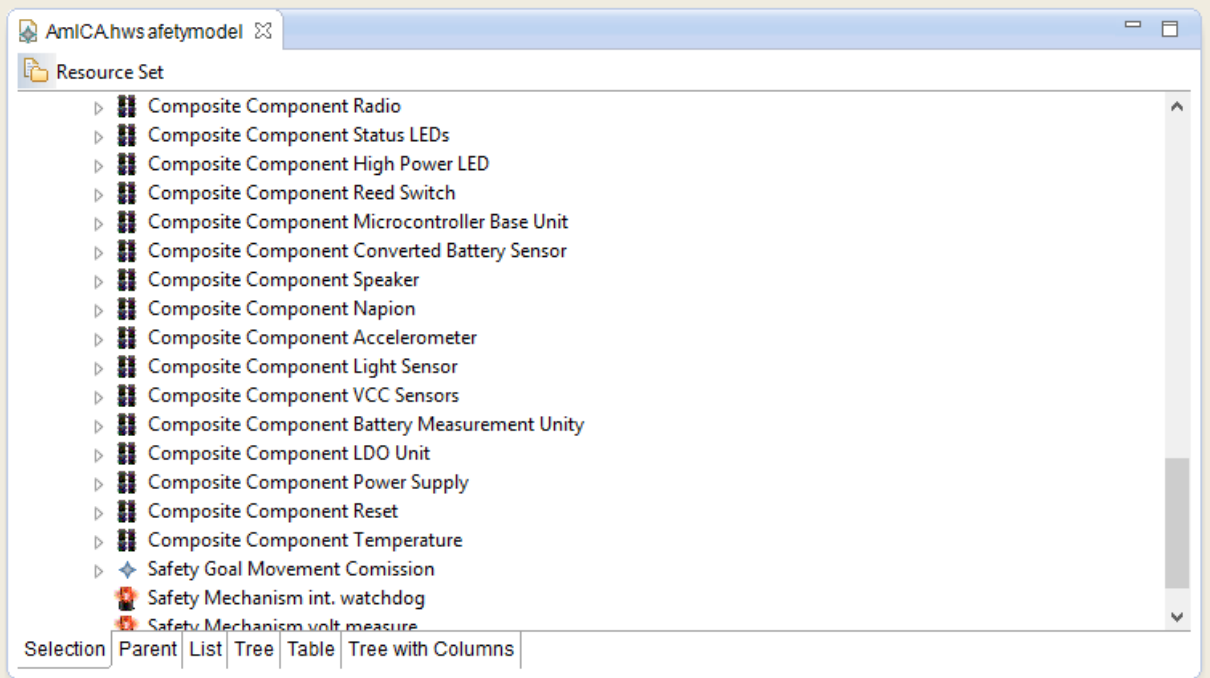
The AmICA node was modelled in the HWSafetyToolbox first with Bottom-Up and finally with Top-Down approach. The top-down modelling was done using the grouped components that have been shown on the previous section of this chapter as the basis. The Figure 4.2.1 shows the created composite components. The failure mode and failure rate information of the hardware components were extracted from the SN 29500 standard.

To create a model for the Hardware Architecture Metrics evaluation, the first step should be the definition of the safety goals of the system. For the AmICA, it was determined the existence of just one safety goal: movement commission. Movement commission is defined as the possibility of the node, after a random hardware fault, to keep sending information indicating the presence of movement of persons in the ambient even if there is none.

After the safety goal was defined, it was modelled the safety mechanisms of the AmICA. Then, it was modelled the Failure Relations of the safety related components. For the components of the Power Supply group, it was defined the another level of composite components and it was modelled the cause-effect relationship between failure modes of

components. Figures 4.2.1 presents a screen capture showing the composite components of the AmICA, modelling the figure 4.1.2 previously presented on this chapter.

Figure 4.2.1 Composite Components of the AmICA



Source: (Author)

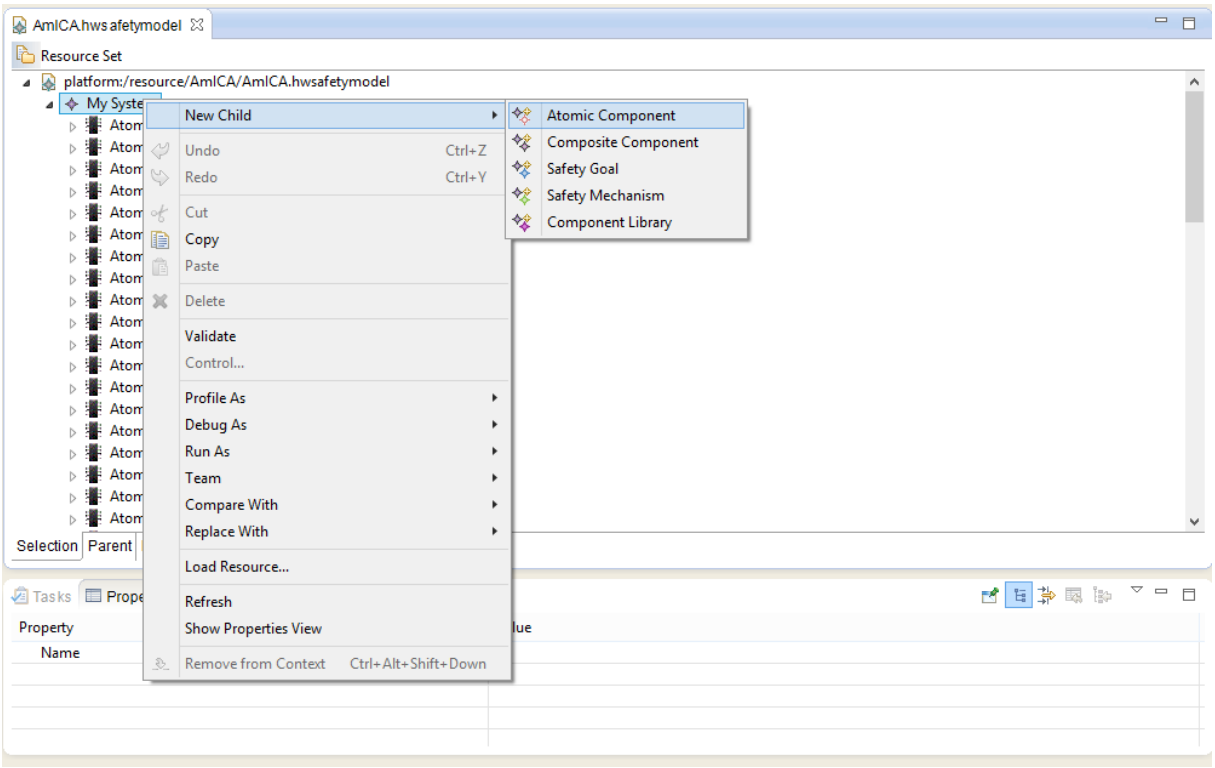
To illustrate how the modelling is done using the tool, it will be shown the step-by-step modelling of the ATMEGA324PV-10AU component. The first step is to create a new Atomic Component, to do this the user should right click the system (the root node of the tree) and select “New Child – Atomic Component”. Figure 4.2.2 illustrates this process.

After this, the user should fill the information related to the component in the form at the bottom of the window. The information required is the component’s name, its failure rate and if it is safety related or not. If the user has a component library, it can also select the component’s type, by doing this the failure rate and the component’s failure mode (children nodes of the component) will be automatically filled for him. Figure 4.2.3 illustrates this paragraph.

When the user finishes the main information about the component, it is time to model its Failure Modes. The user should right click the component and select “New Child – Failure Mode”. When the Failure Mode is created, the user must give the information related to the Failure Mode in bottom of the window. The failure modes of our microcontroller are called All and All, each one of them has 50% of failure rate distribution, this information was retrieved from the Siemens SN29500 norm. It could be created causes and effects with between these failure modes and the failure modes, but that is not the case for the failure

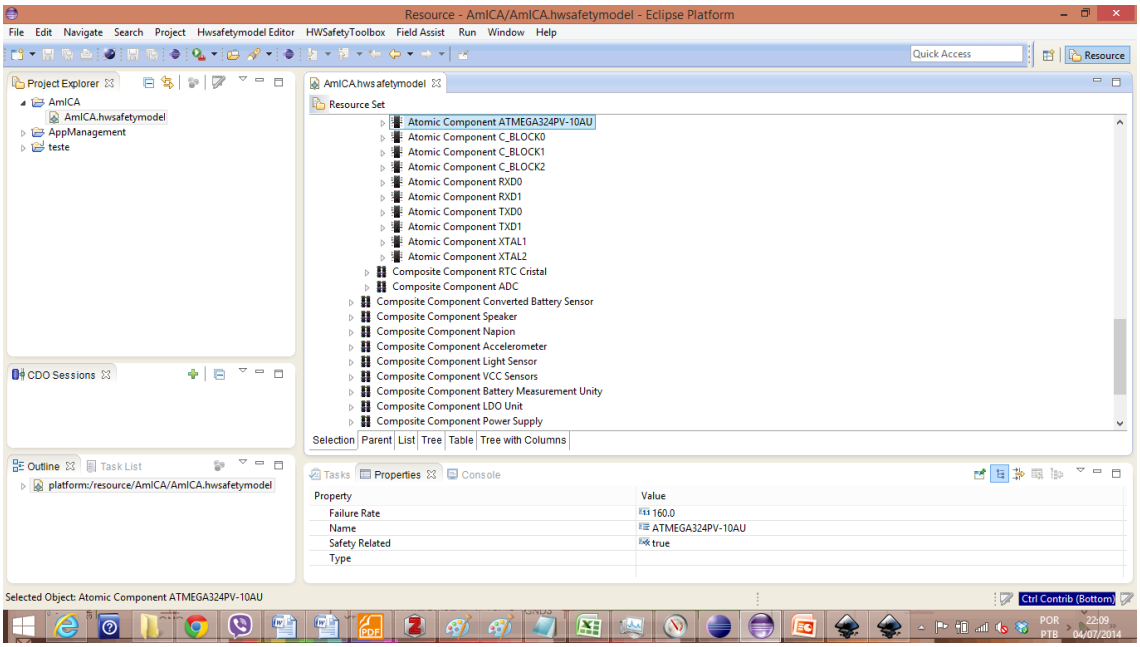
modes of this component. There is also available a field for describing the failure modes. Finally, there is the option for establishing a connection between the failure mode being modeled and a failure relation, but, since we still don't have the failure relation modeled, this connection will be created later.

Figure 4.2.1 The creation of a new component in the system



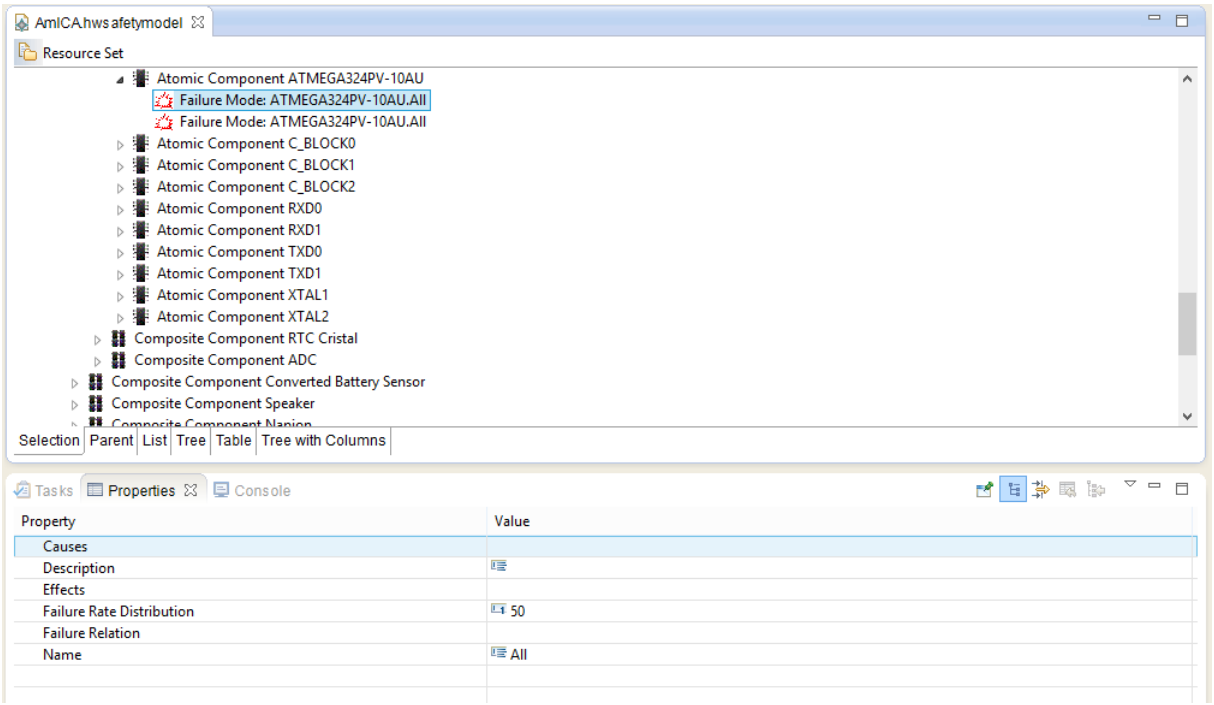
Source: (Author)

Figure 4.2.2 The creation of the microcontroller ATMEGA component



Source: (Author)

Figure 4.2.3 Creating the Failure Modes for the ATMEGA component

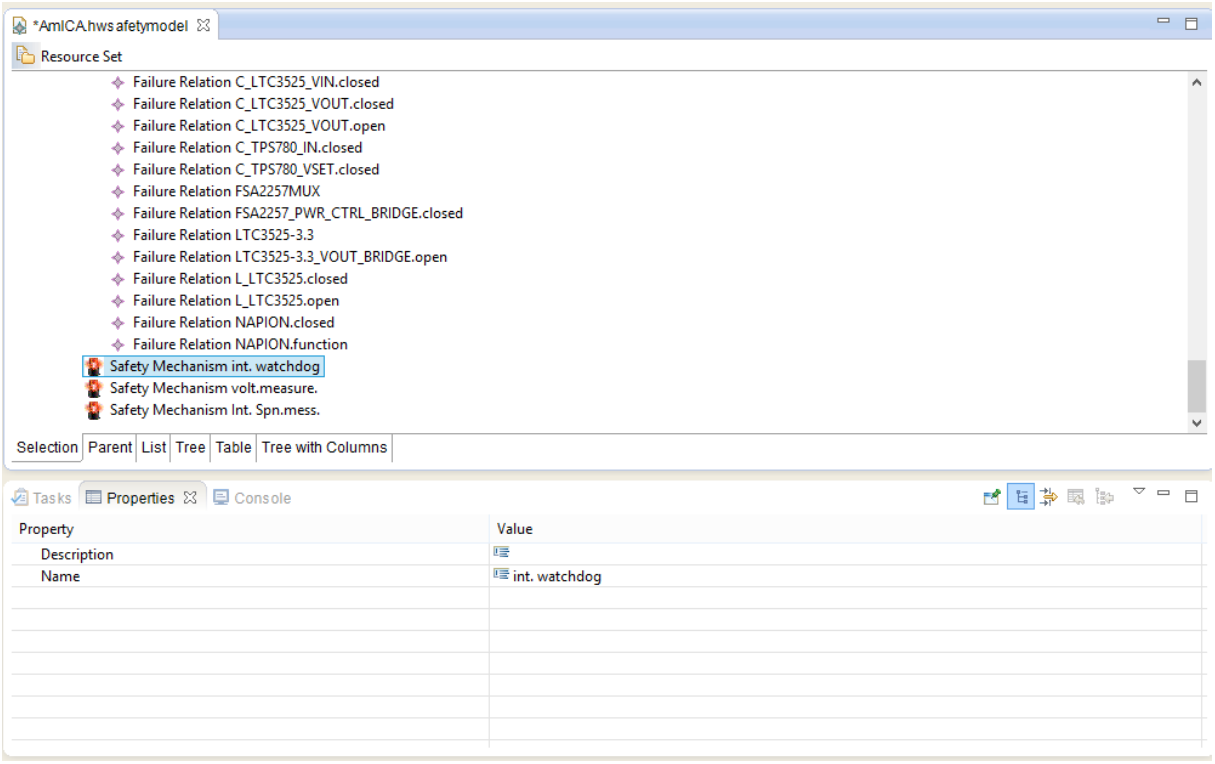


Source: (Author)

Since the ATMEGA has a safety mechanism connected to its failure modes, let's now model the Internal Watchdog safety mechanism. Since Safety Mechanisms can be connected to more than one component's failure modes (the connection is done in the Failure Relation), it is created as a child of the system, but, in this case, the internal watchdog is just related to our ATMEGA microcontroller. To create a new Safety Mechanism, the user should right click the system and select "New Child – Safety Mechanism". There are only two fields of information for safety mechanisms, the name and the description. Figure 4.2.4 shows the internal watchdog.

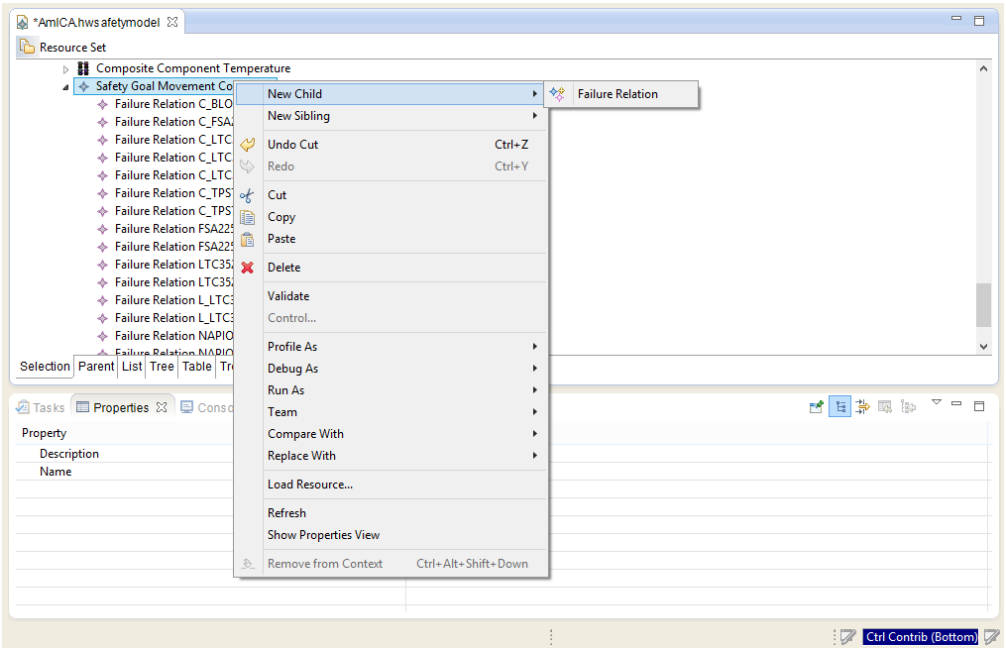
After all these steps, it is time to model the connection between the failure modes and the safety goals, the Failure Relation. For this example, the Safety Goal of Movement Comission is already created, so to model the failure relation of the microcontroller, the user should right click the safety goal and select "New Child – Failure Relation". Figure 4.2.5 shows this step. Next, the user should select the failure mode of this failure relation, to do this, the user clicks on the failure mode field, it will be shown a drop-down list with all the failure modes of the system. Figure 4.2.6 shows the selection of the Failure Mode.

Figure 4.2.4 The modelling of the internal watchdog.



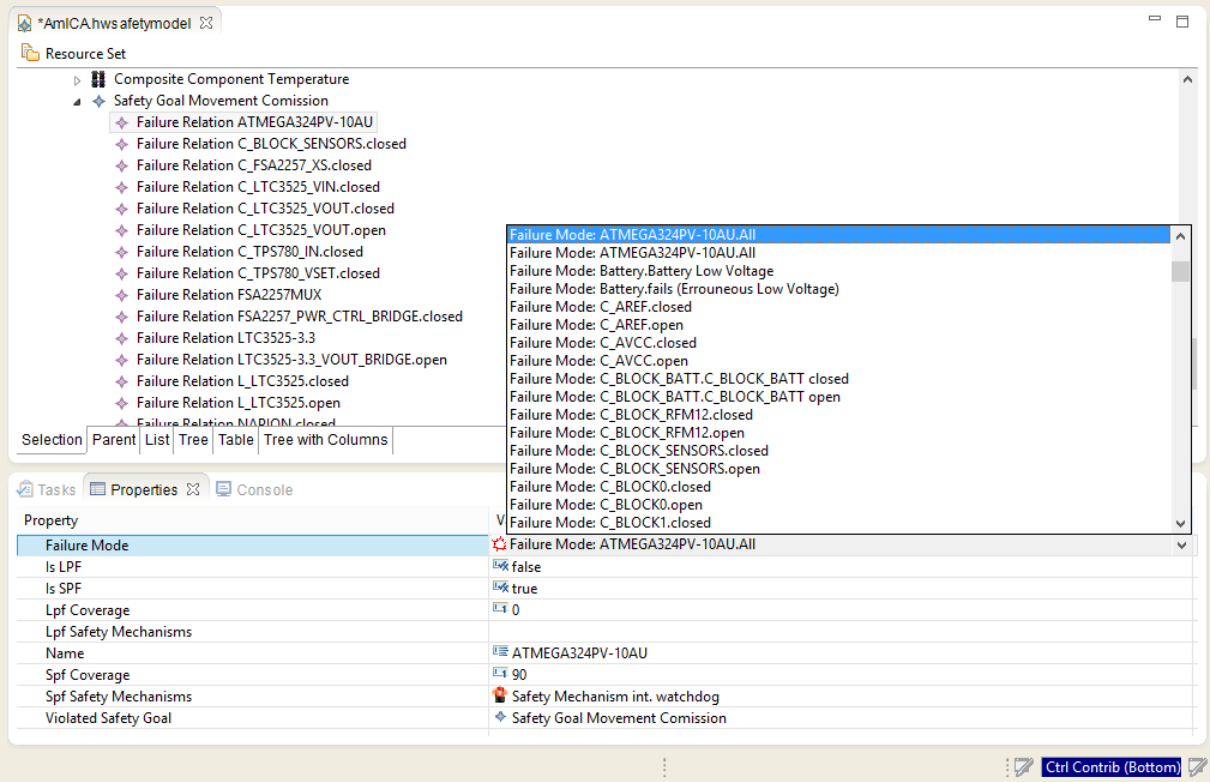
Source: (Author)

Figure 4.2.5 Creation of a new Failure Relation



Source: (Author)

Figure 4.2.6 Selection of the Failure Mode

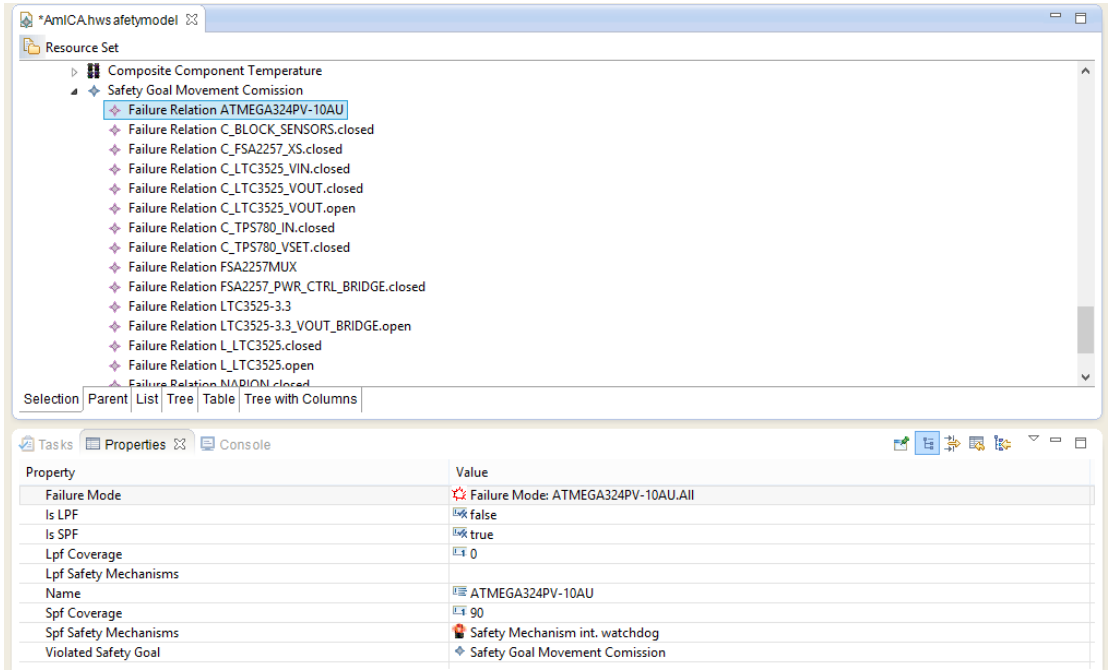


Source: (Author)

After selecting the failure relation (it will be automatically modeled also on the Failure Mode selected the connection with this failure relation). The user shall select the connected Safety Mechanism to single point failure and to latent failures, in this case the internal watchdog is connected to the single point. After that, the user should fill the rest of the information of this failure relation, i.e. mark “Is SPF” if the failure mode has the potential to violate the safety goal in absence of a Safety Mechanisms (in this case, yes); mark “Is LPF” if the failure mode may lead to the violation of safety goal in combination with an independent failure of another component; inform the safety mechanisms coverage for single (in this case, it is 90%) and latent faults. The violated safety goal information is automatically filled by the system and serves to provide direct information to the user, since in large systems, if the users want to recheck the safety goal that is modelling, without this information at hand they should go up in the tree structure. The Figure 2.4.7 shows the modelled failure relation.

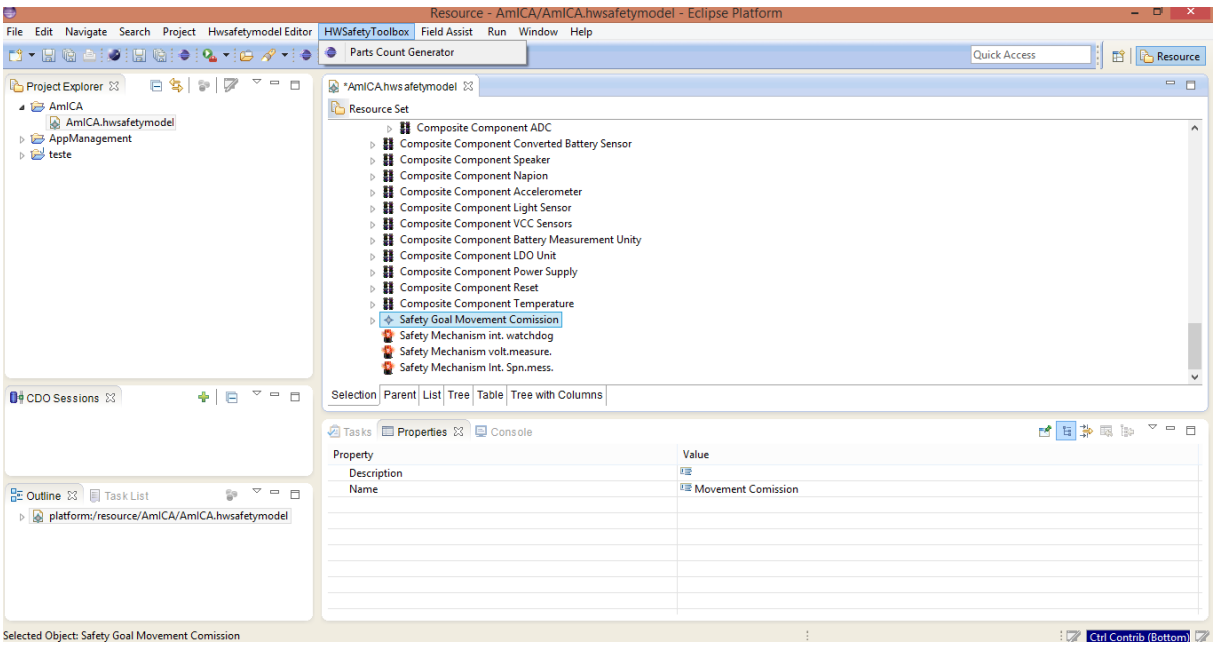
With the explanation provided previously, the user shall be able to model a system on his own. After the system has been all modelled, to create the Excel file with the Hardware Evaluation Metrics, the user must select a Safety Goal with the mouse and then click on the toolbar button HWSafetyToolbox and click on the Parts Count Generator. The Figure 4.2.8 shows this action.

Figure 4.2.7 Selection of the Failure Mode



Source: (Author)

Figure 4.2.8 Generating the Hardware Architecture Metrics Excel File



Source: (Author)

4.3 Test Results

After the modelling was finished, the HWSafetyToolbox was used to generate the Hardware Architecture Metrics table. The results were 80% for the single fault metric and 99.80% for the latent fault metric. The results indicate that the system architecture is not robust enough to handle single point faults with relation to the safety goal analyzed, not achieving any value of the target values. The result for the latent fault metric may look wrong, since it achieved ASIL D, the most rigorous target, but that is because the system has just two components that are classified as possible points of latent faults. Since the AmICA node was not developed with safety as main concern, the results of the metric evaluation are, understandably, not meaningful.

The modelling of a real hardware system brought into view the benefits of using the HWSafetyToolbox. The added layer of abstraction allowed a functional view of the system that is much more meaningful for the person analyzing the hardware. Additionally, the creation of cause-effect relationship between failure modes of components provided the user with good analysis capabilities, by showing which atomic components can lead their functional block to a fault.

5 CONCLUSION

To achieve functional safety on road vehicles is not a simple task. The framework provided by the ISO 26262 standard is of great help on this. However, the standard requires some complex steps for the ones who seek compliance for their products. In this context, the project here developed aimed to be able to help on the phase of hardware development of the ISO 26262, more precisely on the evaluation of the Hardware Architecture Metrics.

This thesis presented the HWSafetyToolbox, a tool for model and analysis of safety related hardware seeking ISO26262 compliance. The HWSafetyToolbox allows the user to model safety related hardware systems, with top-down and bottom-up modelling approaches, and to analyse the modelled system by the automatic evaluation of the *Hardware Architecture Metrics*. The tool also focuses on information reusability, allowing the user to setup a component reliability information library and to copy-and-paste / drag-and-drop parts of a modelled system into another system.

The prototype created fulfils the goals proposed on the chapter 1 and in the beginning of the project. The prototype was validated by the modelling of a real hardware system, the AmICA node, and the demonstration showed how the tool here proposed can be useful to hardware engineers and safety analysts.

5.1 Future Work

Further work on this project is encouraged. The suggested next iterations in this project would be:

- Extraction of failure rate information from the composite components, in order to allow the total abstraction of atomic components.
- Expand the meta-model to represent also hardware-software interaction.
- Expand the tool to help with other processes of the Part 5 of the standard.
- The creation of a safety mechanism library, containing coverage information.

REFERENCES

ADLER, N. et al. **Graphically notated fault modeling and safety analysis in the context of electric and electronic architecture development and functional safety**2012 23rd IEEE International Symposium on Rapid System Prototyping (RSP). **Anais**... In: 2012 23RD IEEE INTERNATIONAL SYMPOSIUM ON RAPID SYSTEM PROTOTYPING (RSP). out. 2012

ADLER, R. et al. Component-based Modeling and Verification of Dynamic Adaptation in Safety-critical Embedded Systems. **ACM Trans. Embed. Comput. Syst.**, v. 10, n. 2, p. 20:1–20:39, jan. 2011.

ADLER, R.; FORSTER, M.; TRAPP, M. **Determining Configuration Probabilities of Safety-Critical Adaptive Systems**21st International Conference on Advanced Information Networking and Applications Workshops, 2007, AINAW '07. **Anais**... In: 21ST INTERNATIONAL CONFERENCE ON ADVANCED INFORMATION NETWORKING AND APPLICATIONS WORKSHOPS, 2007, AINAW '07. maio 2007

AVIZIENIS, A. et al. Basic Concepts and Taxonomy of Dependable and Secure Computing. **IEEE Transactions on Dependable and Secure Computing**, v. 1, p. 11–33, 2004.

BAUMGARTEN, G. Definition and classification of faults of an automotive electrical/electronic system based on the standard ISO 26262. 2012.

BORN, M.; FAVARO, J.; KATH, O. **Application of ISO DIS 26262 in Practice**Proceedings of the 1st Workshop on Critical Automotive Applications: Robustness & Safety. **Anais**...: CARS '10.New York, NY, USA: ACM, 2010Disponível em: <http://doi.acm.org/10.1145/1772643.1772645>. Acesso em: 14 jun. 2014

CUENOT, P. et al. Applying Model Based Techniques for Early Safety Evaluation of an Automotive Architecture in Compliance with the ISO 26262 Standard. [s.d.].

DOMIS, D. et al. **Safety Concept Trees**Reliability and Maintainability Symposium, 2009. RAMS 2009. Annual. **Anais**... In: RELIABILITY AND MAINTAINABILITY SYMPOSIUM, 2009. RAMS 2009. ANNUAL. jan. 2009

DUNN, W. R. Designing safety-critical computer systems. **Computer**, v. 36, n. 11, p. 40–46, nov. 2003.

EDWARDS, S. et al. Design of embedded systems: formal models, validation, and synthesis. **Proceedings of the IEEE**, v. 85, n. 3, p. 366–390, mar. 1997.

GAMMA, E.; BECK, K. **Contributing to Eclipse: Principles, Patterns, and Plug-Ins**. 1 edition ed. Boston: Addison-Wesley Professional, 2003.

HILLENBRAND, M. et al. **Failure mode and effect analysis based on electric and electronic architectures of vehicles to support the safety lifecycle ISO/DIS 26262**2010 21st IEEE International Symposium on Rapid System Prototyping (RSP). **Anais**... In: 2010 21ST IEEE INTERNATIONAL SYMPOSIUM ON RAPID SYSTEM PROTOTYPING (RSP). jun. 2010

IBARRA, I. et al. **ISO 26262 concept phase safety argument for a complex item**7th IET International Conference on System Safety, incorporating the Cyber Security Conference 2012. **Anais**... In: 7TH IET INTERNATIONAL CONFERENCE ON SYSTEM SAFETY, INCORPORATING THE CYBER SECURITY CONFERENCE 2012. out. 2012

IKV++ TECHNOLOGIES AG. **medini analyze**. Disponível em: <http://www.ikv.de/index.php/en/products/functional-safety>. Acesso em: 20 jun. 2014a.

IKV++ TECHNOLOGIES AG. **medini analyze - factsheet**. Disponível em: <http://www.ikv.de/images/stories/ikv/pdf/medinianalyzekeyfacts.pdf>. Acesso em: 20 jun. 2014b.

INTERNATIONAL ELECTROTECHNICAL COMMISSION. IEC 61508 - Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems (E/E/PE, or E/E/PES). 2011.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION. **ISO 26262 - Road vehicles - Functional safety - Part 1**. 2011a.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**. ISO 26262 - Road vehicles - Functional safety - Part 10**. 2011b.

INTERNATIONAL ORGANIZATION FOR STANDARDIZATION**. ISO 26262 - Road vehicles - Functional safety - Part 5**. 2011c.

ITEA3.ORG. **SAFE - WP3 - Deliverable D3.2.2 - Proposal for extension of Meta model for hardware**, 2013.

ITEA3.ORG. **10039 SAFE**. Disponível em: <https://itea3.org/project/safe.html>. Acesso em: 9 jun. 2014.

JEON, S.-H. et al. **Automotive hardware development according to ISO 26262**2011 13th International Conference on Advanced Communication Technology (ICACT). **Anais**... In: 2011 13TH INTERNATIONAL CONFERENCE ON ADVANCED COMMUNICATION TECHNOLOGY (ICACT). fev. 2011

KRAMMER, M. et al. **Improving methods and processes for the development of safety-critical automotive embedded systems**2010 IEEE Conference on Emerging Technologies and Factory Automation (ETFA). **Anais**... In: 2010 IEEE CONFERENCE ON EMERGING TECHNOLOGIES AND FACTORY AUTOMATION (ETFA). set. 2010

MARIANI, R. **The impact of functional safety standards in the design and test of reliable and available integrated circuits**Test Symposium (ETS), 2012 17th IEEE European. **Anais**... In: TEST SYMPOSIUM (ETS), 2012 17TH IEEE EUROPEAN. maio 2012

PALIN, R. et al. **ISO 26262 safety cases: Compliance and assurance**2011 6th IET International Conference on System Safety. **Anais**... In: 2011 6TH IET INTERNATIONAL CONFERENCE ON SYSTEM SAFETY. set. 2011

PLATZNER, M.; TEICH, J.; WEHN, N. (EDS.). **Dynamically Reconfigurable Systems**. Dordrecht: Springer Netherlands, 2010.

RATAN, V. et al. **Safety analysis tools for requirements specifications**Proceedings of the Eleventh Annual Conference on Computer Assurance, 1996. COMPASS '96, Systems Integrity. Software Safety. Process Security. **Anais**... In: PROCEEDINGS OF THE ELEVENTH ANNUAL CONFERENCE ON COMPUTER ASSURANCE, 1996. COMPASS '96, SYSTEMS INTEGRITY. SOFTWARE SAFETY. PROCESS SECURITY. jun. 1996

S. WILLE et al. AmICA - A flexible, compact, easy-to-program and low-power WSN platform. **Mobiquitous 2010**, n. 2010, dez. 2010.

SAFE SAFE AUTOMOTIVE SOFTWARE ARCHITECTURE. **SAFE Safe Automotive soFtware architEcture**. Disponível em: <http://www.safe-project.eu/>. Acesso em: 20 jun. 2014.

SEETHA RAMAIAH, P.; BEN SWARUP, M.; KUMAR, K. R. **Conceptual Modeling for Safety Critical Computer Systems**Ninth ACIS International Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing, 2008. SNPD ’08. **Anais**... In: NINTH ACIS INTERNATIONAL CONFERENCE ON SOFTWARE ENGINEERING, ARTIFICIAL INTELLIGENCE, NETWORKING, AND PARALLEL/DISTRIBUTED COMPUTING, 2008. SNPD ’08. ago. 2008

STEINBERG, D. et al. **EMF: Eclipse Modeling Framework**. 2 edition ed. [s.l.] Addison-Wesley Professional, 2008.

STIRGWOLT, P. **Effective management of functional safety for ISO 26262 standard**Reliability and Maintainability Symposium (RAMS), 2013 Proceedings - Annual. **Anais**... In: RELIABILITY AND MAINTAINABILITY SYMPOSIUM (RAMS), 2013 PROCEEDINGS - ANNUAL. jan. 2013

ZELKOWITZ, M. V.; WALLACE, D. R. Experimental models for validating technology. **Computer**, v. 31, n. 5, p. 23–31, maio 1998.

ANNEX A

The present annex contains bits of the XMI file of the modelled AmICA system, since the full information would be several pages long, here it is presented just some of the atomic and composite components, the failure modes, safety mechanisms and the most important failure relations. This annex also aims to show the XMI file structure adopted by EMF editor and consequently the HWSafetyToolbox.

```
<?xml version="1.0" encoding="UTF-8"?>
<hwsafetymodel:MySystem xmi:version="2.0"
xmlns:xmi="http://www.omg.org/XMI"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:hwsafetymodel="http://hwsafetymodel/1.0">
  <components xsi:type="hwsafetymodel:CompositeComponent"
name="Microcontroller Base Unit">
    <nestedComponents xsi:type="hwsafetymodel:CompositeComponent"
name="uC">
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="ATMEGA324PV-10AU" safetyRelated="true" failureRate="160.0">
        <failureModes name="All" failureRateDistribution="50"/>
        <failureModes name="All" failureRateDistribution="50"/>
      </nestedComponents>
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="C_BLOCK0" failureRate="1.0">
        <failureModes name="open" failureRateDistribution="20"/>
        <failureModes name="closed" failureRateDistribution="80"/>
      </nestedComponents>
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="C_BLOCK1" failureRate="1.0">
        <failureModes name="open" failureRateDistribution="20"/>
        <failureModes name="closed" failureRateDistribution="80"/>
      </nestedComponents>
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="C_BLOCK2" failureRate="1.0">
        <failureModes name="open" failureRateDistribution="20"/>
        <failureModes name="closed" failureRateDistribution="80"/>
      </nestedComponents>
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="RXD0">
        <failureModes name=""/>
      </nestedComponents>
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="RXD1">
        <failureModes name=""/>
      </nestedComponents>
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="TXD0">
        <failureModes name=""/>
      </nestedComponents>
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="TXD1">
        <failureModes name=""/>
      </nestedComponents>
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="XTAL1">
        <failureModes name=""/>
      </nestedComponents>
```

```
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="XTAL2">
        <failureModes name=""/>
      </nestedComponents>
    </nestedComponents>
    <nestedComponents xsi:type="hwsafetymodel:CompositeComponent" name="RTC
Cristal">
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="32768-QUARZ" failureRate="15.0">
        <failureModes name="open" failureRateDistribution="45"/>
        <failureModes name="closed" failureRateDistribution="45"/>
        <failureModes name="drift 5" failureRateDistribution="5"/>
        <failureModes name="drift 50" failureRateDistribution="5"/>
      </nestedComponents>
    </nestedComponents>
    <nestedComponents xsi:type="hwsafetymodel:CompositeComponent"
name="ADC">
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="C_AREF" failureRate="1.0">
        <failureModes name="open" failureRateDistribution="20"/>
        <failureModes name="closed" failureRateDistribution="80"/>
      </nestedComponents>
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="L_AVCC" failureRate="1.5">
        <failureModes name="open" failureRateDistribution="90"/>
        <failureModes name="closed" failureRateDistribution="10"/>
      </nestedComponents>
      <nestedComponents xsi:type="hwsafetymodel:AtomicComponent"
name="C_AVCC" failureRate="1.0">
        <failureModes name="open" failureRateDistribution="20"/>
        <failureModes name="closed" failureRateDistribution="80"/>
      </nestedComponents>
    </nestedComponents>
  </components>
  <safetyGoals name="Movement Comission">
    <failureRelations spfCoverage="90" isSPF="true"
spfSafetyMechanisms="//@safetyMechanisms.0"
violatedSafetyGoal="//@safetyGoals.0" name="ATMEGA324PV-10AU"/>
    <failureRelations isSPF="true" violatedSafetyGoal="//@safetyGoals.0"
name="C_BLOCK_SENSORS.closed"
failureMode="//@components.55/@nestedComponents.0/@failureModes.1"/>
    <failureRelations isSPF="true" violatedSafetyGoal="//@safetyGoals.0"
name="C_FSA2257_XS.closed"
failureMode="//@components.58/@nestedComponents.0/@nestedComponents.15/@fai
lureModes.1"/>
    <failureRelations isSPF="true" violatedSafetyGoal="//@safetyGoals.0"
name="C_LTC3525_VIN.closed"
failureMode="//@components.58/@nestedComponents.1/@nestedComponents.0/@fail
ureModes.1"/>
    <failureRelations isSPF="true" violatedSafetyGoal="//@safetyGoals.0"
name="C_LTC3525_VOUT.closed"
failureMode="//@components.58/@nestedComponents.1/@nestedComponents.1/@fail
ureModes.1"/>
    <failureRelations isSPF="true" violatedSafetyGoal="//@safetyGoals.0"
name="C_LTC3525_VOUT.open"
failureMode="//@components.58/@nestedComponents.1/@nestedComponents.1/@fail
ureModes.0"/>
    <failureRelations isSPF="true" violatedSafetyGoal="//@safetyGoals.0"
name="C_TPS780_IN.closed"
failureMode="//@components.57/@nestedComponents.0/@failureModes.1"/>
```

```
    <failureRelations isSPF="true" violatedSafetyGoal="//@safetyGoals.0"
name="C_TPS780_VSET.closed"
failureMode="//@components.57/@nestedComponents.2/@failureModes.1"/>
    <failureRelations spfCoverage="90" isSPF="true"
spfSafetyMechanisms="//@safetyMechanisms.2"
violatedSafetyGoal="//@safetyGoals.0" name="FSA2257MUX"
failureMode="//@components.58/@nestedComponents.0/@nestedComponents.1/@fail
ureModes.0"/>
    <failureRelations spfCoverage="90" isSPF="true"
spfSafetyMechanisms="//@safetyMechanisms.2"
violatedSafetyGoal="//@safetyGoals.0" name="FSA2257_PWR_CTRL_BRIDGE.closed"
failureMode="//@components.58/@nestedComponents.0/@nestedComponents.4/@fail
ureModes.0"/>
    <failureRelations spfCoverage="90" isSPF="true"
spfSafetyMechanisms="//@safetyMechanisms.1"
violatedSafetyGoal="//@safetyGoals.0" name="LTC3525-3.3"
failureMode="//@components.58/@nestedComponents.1/@nestedComponents.2/@fail
ureModes.0"/>
    <failureRelations spfCoverage="90" isSPF="true"
spfSafetyMechanisms="//@safetyMechanisms.1"
violatedSafetyGoal="//@safetyGoals.0" name="LTC3525-3.3_VOUT_BRIDGE.open"
failureMode="//@components.58/@nestedComponents.5/@nestedComponents.11/@fai
lureModes.0"/>
    <failureRelations spfCoverage="90" isSPF="true"
spfSafetyMechanisms="//@safetyMechanisms.1"
violatedSafetyGoal="//@safetyGoals.0" name="L_LTC3525.closed"
failureMode="//@components.58/@nestedComponents.1/@nestedComponents.3/@fail
ureModes.1"/>
    <failureRelations spfCoverage="90" isSPF="true"
spfSafetyMechanisms="//@safetyMechanisms.1"
violatedSafetyGoal="//@safetyGoals.0" name="L_LTC3525.open"
failureMode="//@components.58/@nestedComponents.1/@nestedComponents.3/@fail
ureModes.0"/>
    <failureRelations isSPF="true" violatedSafetyGoal="//@safetyGoals.0"
name="NAPION.closed"
failureMode="//@components.52/@nestedComponents.0/@failureModes.1"/>
    <failureRelations isSPF="true" violatedSafetyGoal="//@safetyGoals.0"
name="NAPION.function"
failureMode="//@components.52/@nestedComponents.0/@failureModes.2"/>
  </safetyGoals>
  <safetyMechanisms name="int. watchdog"/>
  <safetyMechanisms name="volt.measure." description=""/>
  <safetyMechanisms name="Int. Spn.mess." description=""/>
</hwsafetymodel:MySystem>
```

ANNEX B

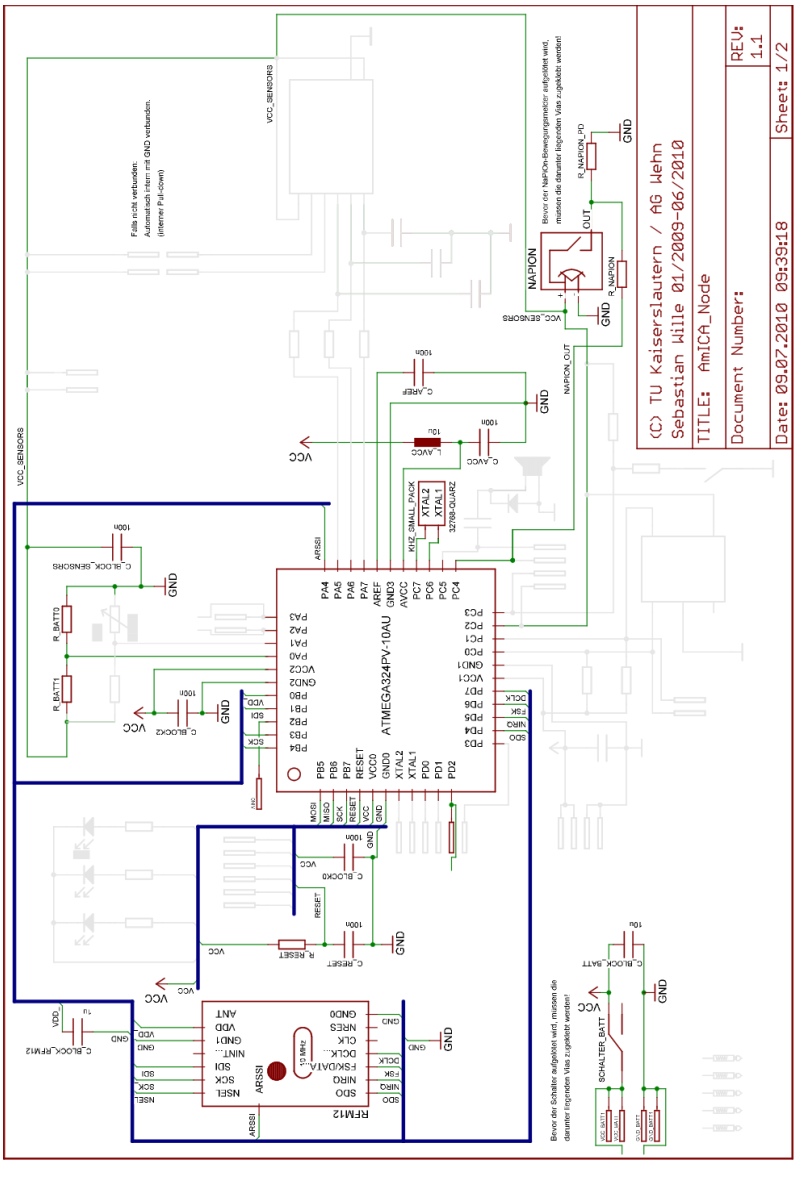
Here is presented a screen capture of the MS Excel file generated using the HWSafetyToolbox for the evaluation Hardware Evaluation Metrics. This file was generated for the AmICA model. The first line of the spreadsheet was fixed to help with the visualization.

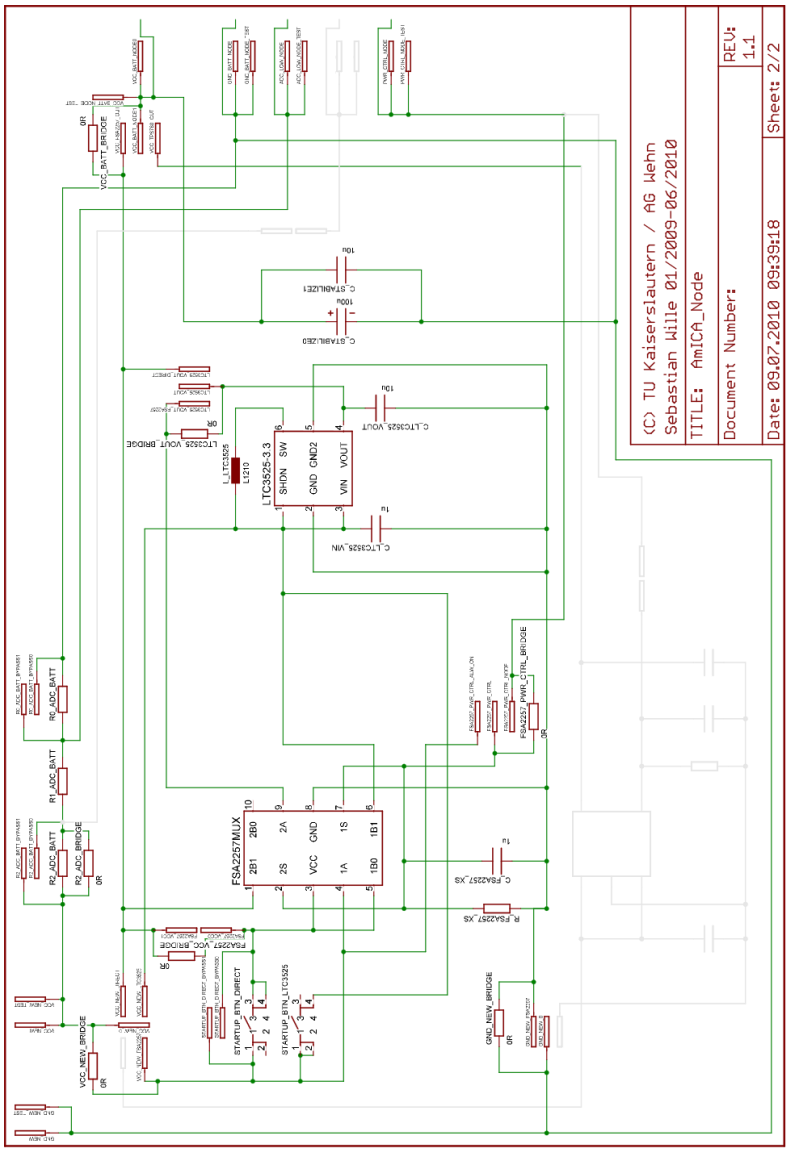
O1

	A	B	C	D	E	F	G	H	I	J	K	L	M
1	Component Name	Failure rate / FIT	(S)afety (R)elated Component? (N)o (S)afety (R)elated Component?	Failure Mode	Failure Rate Distribution	Failure mode that has the potential to violate the safety goal in absence of Safety Mechanisms?	Safety mechanism(s) allowing to prevent the failure mode from violating the safety goal?	Failure mode coverage wrt. violation of safety goal	Residual or Single Point Fault failure rate / FIT	Failure mode that my lead to the violation of safety goal in combination with an independent failure of another component?	Safety mechanism(s) allowing to prevent the failure mode from being latent?	Failure mode coverage wrt. latent failures	Latent Multiple Point Fault failure rate / FIT
190	R2_ADC_BRIDGE	0,2	NSR	open	90								
191				closed	10								
192	VCC_BATT	0	NSR		0								
193	C_TPS780_IN	1	SR	open	20								
194				closed	80	X		0	0,8				
195	C_TPS780_OUT	1	SR	open	20					X		0	0,2
196				closed	80	X		0	0,8				
197	C_TPS780_VSET	1	SR	open	20					X		0	0,2
198				closed	80	X		0	0,8				
199	R_TPS780_VSET	0,2	NSR	open	90								
200				closed	10								
201	TPS780_PWR_CTRL0	0	NSR		0								

ANNEX C

The present annex contains the block diagrams of the AmICA sensor node. These block diagrams were provided by Dipl-Ing. Sebastian Wille. To know more about the AmICA platform, please refer to (S. WILLE et al., 2010)





16.04.2012 16:40:10 C:\Eigene Dateien\Specu\Eagle-Projekte\AmICA_Node\AmICA_Node.sch (Sheet: 2/2)