

# **Behavioural Safety of Technical Systems**

Mario Gleirscher



Fakultät für Informatik  
der Technischen Universität München  
Software & Systems Engineering

**Behavioural Safety of Technical Systems**

**Verhaltenssicherheit technischer Systeme**

**Mario Gleirscher**

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Hans-Arno Jacobsen

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Dr. h.c. Manfred Broy

2. Univ.-Prof. Dr. Jan Peleska,

Universität Bremen

Die Dissertation wurde am 25.06.2014 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 22.10.2014 angenommen.



**Short Biography** Mario Gleirscher ([mario.gleirscher@tum.de](mailto:mario.gleirscher@tum.de)) owns professional education in mechanical engineering (core area: production technology). He received a diploma degree in informatics (core area: theoretical computer science, subsidiary subject: mathematics) from the Technische Universität München (TUM, [www.tum.de](http://www.tum.de)). He collected three years of professional experience in commercial software development, and eight years of experience in doing research and education in systems, requirements and process engineering. His research in quality assurance of software-intensive technical systems was supported by the Software & Systems Engineering chair of Prof. Dr. Manfred Broy at the TUM Department of Informatics. For his doctoral thesis, he worked on a method for safety analysis.

Computer science is no more  
about computers than  
astronomy is about telescopes.

---

*(Edsger W. Dijkstra)*

**Copyright Waiver (Urheberrechtsvermerk gem. §106ff UrhG)**    Wednesday 3<sup>rd</sup> December, 2014

The reproduction, distribution and utilisation of this document as well as the communication of its contents to others have to comply with §19 of the regulations for the award of doctoral degrees at the TUM (status: 2010). Deviations thereof require express authorisation. Offenders will be held liable for the payment of damages. All rights reserved in the event of the grant of a patent, utility model or design.

Weitergabe sowie Vervielfältigung dieses Dokuments, Verwertung und Mitteilung seines Inhalts sind durch §19 der Promotionsordnung der TUM (Stand: 2010) geregelt. Abweichungen davon bedürfen der ausdrücklichen Autorisierung. Zuwiderhandlungen verpflichten zu Schadenersatz. Alle Rechte für den Fall der Patent-, Gebrauchsmuster- oder Geschmacksmustereintragung vorbehalten.

© Mario Gleirscher 2014

All rights reserved / Alle Rechte vorbehalten □

**Abstract** Producers of technical systems are obliged to identify and treat hazards of these systems and their software-intensive control subsystems. *Hazard analysis* is an indispensable task in system *specification* and development. Practices, such as improper modelling, insufficient support and use of analysis methods, bias to reliability goals, and neglect of the system environment promote *specification defects* that can compromise *software and system safety*. Hence, there exists a demand for methods to effectively carry through *safety-oriented requirements validation*.

This thesis presents a *validation method for specifications* extending the *guidance for achieving safety of technical systems*. The method fosters (i) the encoding of safety-related domain knowledge, (ii) the identification and characterisation of hazards and (iii) the specification of safety measures as an input to system and software development. The formal approach applies behavioural modelling of the system and its environment, and provides patterns to disclose and remove hazardous specification defects. The evaluation is achieved by a case study, a literature review and expert interviews.

**Kurzzusammenfassung** Hersteller technischer Systeme sind verpflichtet, das Gefährdungspotential ihrer Anlagen und deren softwarebasierten Steuerungen zu untersuchen. Die *Gefährdungs- und Risikoanalyse* ist eine unerlässliche Aufgabe der *Spezifikation* und Entwicklung solcher Systeme. Praktiken, wie z. B. unzulängliche Modellbildung, wenig Unterstützung durch Analysemethoden, Konzentration auf Zuverlässigkeitsziele und mangelnde Berücksichtigung der Systemumgebung begünstigen *Spezifikationsdefekte* und erschweren die Sicherheitsüberprüfung insbesondere softwarebasierter Systemteile. Es besteht daher ein Bedarf an Hilfsmitteln zur effektiven *Absicherung von Anforderungs- und Systemspezifikationen*.

Die vorliegende Arbeit stellt eine *Absicherungsmethode für Spezifikationen* vor, um das *Gefährdungspotential technischer Systeme* zu verringern. Diese Methode unterstützt (i) die Abbildung von sicherheitsbezogenem Domänenwissen, (ii) die Identifikation und Charakterisierung von Gefährdungen sowie (iii) die Spezifikation von Sicherheitsmaßnahmen als Gegenstand der System- und Softwareentwicklung. Der formale Ansatz basiert auf einer Verhaltensmodellierung des Systems sowie seiner Umgebung und bietet Muster zur Erkennung und Behandlung gefährlicher Spezifikationsdefekte. Die Evaluierung erfolgt durch eine zweiteilige Fallstudie, Literaturrecherche und Expertenbefragungen.



**Acknowledgement** I spend sincere gratitude to my supervisor Prof. Dr. Dr. h.c. Manfred Broy for his mentorship, steady patience, helpful feedback and organisational support throughout these advancing years. I also thank Prof. Dr. Jan Peleska for his critical review, for perceptive hints and his second opinion.

All my colleagues at the chair delighted me with a cordial work climate during my whole affiliation. Special thanks go to: Alarico Campetelli, Georg Hackenberg, Benedikt Hauptmann, Florian Hölzl, Maximilian Irlbeck, Diego Marmsoler, Birgit Penzenstadler, Daniel Rațiu, Andreas Vogelsang, Sebastian Winter and Xiuna Zhu who were tenacious to keep me on track, and for their plenty of time for mind-boggling, stimulating and philosophic discussions during recreational tea, coffee and foosball sessions—Andreas Bauer, Johannes Hölzl, Benjamin Hummel, Leonid Kof and Maria Spichkova who helped me orientate in the jungle of formal methods and who enlightened me with their specialist hints—Katharina Spies for her insight into how to get along with everyday troubles of doctoral students—all technical staffs for their indispensable service keeping me perfectly productive. Further thanks go to several scholars whom I owe valuable knowledge: Bernhard Schätz and Stefan Wagner for their insights into doing scientific work on software modelling, development and quality assurance—David Parnas for a discussion about the commonalities of classical and software engineering—Peter Struss for his expertise in qualitative physics.

I thank all my industrial partners, among them engineers from Airbus Group (formerly EADS), BMW AG, IBM Deutschland GmbH, ITK Engineering AG, Validas AG and six unstated Munich SMEs, for their supportive information on software practice and safety standards, valuable insight into daily engineering tasks, and for financial support of the projects I attended. Without this support, my first publication on hazard analysis (Gleirscher 2013b) and my case on road vehicle safety (Section 6.2) would hardly have been possible. My cordial gratitude goes to the safety practitioners who shared their experience by giving me an interview. For discussions and helpful feedback, I am also thankful to: Oscar Daigeler, Christian Facchi, Hartmut Hencke, Stefan Rink, Frank Sommer and Bernd Spanfelner about software quality assurance in the telecommunication and finance sectors, and formal methods—my student team from the UnternehmerTUM business plan workshop (Andreas Gerö, Clemens Ernstberger and Lukas Stacheder) for a great team performance—master students Sonila Dobi and Carmen Carlan for discussions about physical modelling and safety cases.

Moreover, Maximilian Irlbeck, Ursula Leiter, Klaus Lochmann, Felix von Ranke and Tobias Zimmermann gratified me with reviews, thoughtful questions and hints—Ed Robinson helped me a lot with improving my English writing—Sebastian Kienzl, Clemens Lanthaler, Franz Strasser and Christian Müller had an open ear in many situations of this era of gruelling uncertainty—Nina Jäger proved patience in one of the toughest stages of this lengthy and questionable project, farewell to these wonderful years of heartfelt trial: I am sorry. Not least, I owe my family invaluable gratitude for their support which made possible many things across the years.





# Contents

Abstract and Acknowledgement . . . . .	v
<b>1 Motivation and Overview</b> . . . . .	<b>1</b>
1.1 Safety of Technical Systems . . . . .	1
1.2 Thesis Summary . . . . .	5
1.3 Research Design and Outline . . . . .	8
<b>I Theory and Approach</b> . . . . .	<b>10</b>
<b>2 Technical Systems</b> . . . . .	<b>11</b>
2.1 Systems Engineering . . . . .	11
2.2 System Theory, Modelling and Specification . . . . .	13
2.3 Formal Preliminaries . . . . .	17
2.4 System Specification: A Generic Framework . . . . .	28
2.5 Notes and Further Reading . . . . .	29
<b>3 Safety</b> . . . . .	<b>31</b>
3.1 Safety Viewpoints and Standards . . . . .	31
3.2 Hazard Analysis Techniques . . . . .	34
3.3 Safety Measures . . . . .	37
3.4 More Recent Related Work . . . . .	38
<b>4 Behavioural Safety: Concepts</b> . . . . .	<b>46</b>
4.1 System Specification: A Safety-related Framework . . . . .	46
4.2 Safety-related Defects . . . . .	48
4.3 Mishaps, Hazards and Causal Factors . . . . .	53
4.4 Behavioural Safety . . . . .	56
4.5 Responsibility and Restriction . . . . .	59
4.6 Safety Measures . . . . .	60
4.7 A Stop Criterion for Safety-oriented Validation . . . . .	65
4.8 Notes and Further Reading . . . . .	65
<b>5 Behavioural Safety: Procedure</b> . . . . .	<b>68</b>
5.1 Modelling Stage: Understand System . . . . .	68
5.2 Analysis Stage: Identify Hazards . . . . .	71

5.3 Assurance Stage: Improve System Functionality . . . . .	72
5.4 Notes and Further Reading . . . . .	73
<b>II Application and Evaluation</b>	<b>75</b>
<b>6 Case Study</b>	<b>76</b>
6.1 Pilot Case: Automated Teller Machine . . . . .	76
6.2 Approval Case: Commercial Road Vehicle . . . . .	90
<b>7 Discussion</b>	<b>104</b>
7.1 Evaluation of the Case Study . . . . .	104
7.2 Improvements on Related Work . . . . .	105
7.3 Some Limitations of the Approach . . . . .	108
7.4 Challenges and Hints . . . . .	109
7.5 Conclusions . . . . .	112
7.6 Further Work . . . . .	114
<b>Bibliography</b>	<b>116</b>
<b>A Library, Evidence and Excursions</b>	<b>133</b>
A.1 Transition System Patterns and Guide Words . . . . .	133
A.2 Procedure and Data on Interviews of Safety Practitioners . . . . .	140
A.3 Data on the Systematic Map of Related Work . . . . .	141
A.4 Data on the Comparison with Other Procedures . . . . .	141
A.5 Data on the Automated Teller Machine Case . . . . .	141
A.6 Data on the Commercial Road Vehicle Case . . . . .	141
A.7 Application and Evaluation of the Defect Taxonomy . . . . .	157
A.8 Defects and Model Validity . . . . .	157
<b>B Indices</b>	<b>163</b>
B.1 Glossary of Symbols and Notation . . . . .	163
B.2 List of Figures . . . . .	165
B.3 List of Tables . . . . .	166
B.4 List of Examples . . . . .	166
Index . . . . .	169

## Motivation and Overview

### Contents

1.1	Safety of Technical Systems . . . . .	1
1.2	Thesis Summary . . . . .	5
1.3	Research Design and Outline . . . . .	8

### 1.1. Safety of Technical Systems

*Technical systems* (e.g. plants, machines, vehicles) provide *functionality* which can harmfully impact assets in their environment (e.g. operators, users). *Safety*—the degree of freedom from *hazards*—is a critical property of such systems and their software-intensive *control subsystems* (Leveson 2012, McDermid 2002). Producers face the pressure of high expectations, accident statistics, product liability claims and competition in safety innovations (Rasmussen 1997). This pressure is mirrored by national laws (Luksch 2012), standards such as IEC Std. 61508 (2011) or ISO Std. 26262 (2011), EU directives (Gehlen 2010) or process guidelines. Table 1.1 indicates how *safety requirements* arise from applications as an issue of systems engineering.<sup>1</sup> Systems engineering includes *requirements engineering* to specify the system interface and functionality, and architecture design in order to structure development.<sup>2</sup>

Technical system  
Safety  
control subsystem

**Challenges in Safety Engineering** For system validation and verification (V&V), during requirements engineering a safety engineer gains an understanding of hazards. He or she identifies and assesses hazards (e.g. failure of a road vehicle or a pre-crash state) using various sources of information, such as human error classifications<sup>3</sup>, ac-

<sup>1</sup>See, e.g. Nader (1965), Neumann (1995), Perrow (1984) for further historical accident data.

<sup>2</sup>See, e.g. Ehrlenspiel and Meerkamm 2013, Jackson 1983, Lunze 2010, van Lamsweerde 2009.

<sup>3</sup>See, e.g. Cacciabue 2004, Hall and Silva 2008, Leveson 2012, Shappell and Wiegmann 2000.

## 1. Motivation and Overview

Application or System Category	No. of Incidents/Accidents	Causal Factors
German nuclear power plants, in last 30 years ( <a href="#">Spiegel 2012</a> )	> 4k incidents of INES level $\leq 3$	technical flaws or maloperation
German trucks, 2003 ( <a href="#">Evers 2012</a> )	$\approx$ 38k traffic accidents	6% technical reasons, > 40% inadequate speed, inattention or fatigue
U.S. motor vehicles, 2005-'07 ( <a href="#">NHTSA 2008</a> )	$\approx$ 2.2M traffic accidents	$\approx$ 3% technical reasons incl. 0.75% defective brakes
German production plants, 2007-'10 ( <a href="#">BAUA 2012</a> )	983 fatal work accidents	$\approx$ 15% technical flaws
German road vehicles and traffic, 2011 ( <a href="#">ADAC 2012</a> )	$\approx$ 306k traffic accidents with personal injury	maloperation, $\approx$ 1% technical flaws
Buenos Aires railways, Feb 2012 ( <a href="#">Focus 2012</a> )	one train accident (657 injured, 49 casualties)	defective brakes

Table 1.1.: Data on incidents and accidents of technical systems and their severity

cident databases<sup>4</sup>, reliability statistics, injury severity scorings, the international nuclear event scale (INES; [IAEA 2008](#)), expert discussions, driving situation registers and insurance models. By consulting architecture design documents, he or she relates potential defects to hazards at the system interface. These steps enable the engineer to recommend measures to treat hazards and their *causal factors*. These measures also constrain the software-intensive control subsystem. Furthermore, he or she creates plans for safety-oriented V&V, acting as a requirements engineer to *validate the specification for safety* and as a V&V engineer to prepare for the verification of safety ([Ericson 2005](#), [Lutz 1993, 2000](#)). This procedure involves several questions: Which hazards can and must be considered? Which defects may exist from a safety viewpoint and how can they be handled? How can safety goals and responsibilities be derived and justified? How can safety measures be specified? Example 1.1 objectifies these questions.

- ⌈ **Example 1.1 (Car Airbag)** *What is the intent behind a car airbag? Does it release only and exactly in the right driving situations? Which hazards arise also in these situations? Which causal factors obstruct release as intended? How can they be avoided or kept at minimum risk? Are all intended, extreme, unexpected or defective situations known and handled? How can airbag behaviour be specified in such situations? How can these situations be covered during verification? Which obstacles arise from separate V&V of*
- ⌋ *the airbag?*

<sup>4</sup>For example, from AOPA Air Safety Institute ([www.aopa.org/Education/Accident-Case-Studies.aspx](http://www.aopa.org/Education/Accident-Case-Studies.aspx)), U.S. DoT Federal Motor Carrier Safety Administration ([ai.fmcsa.dot.gov](http://ai.fmcsa.dot.gov), [NHTSA 2008](#)), or "Institut für Unfallanalysen" ([unfallforensik.maindev.de](http://unfallforensik.maindev.de)).

**Problems in Safety Engineering** As discussed in the literature<sup>5</sup>, the presence of *specification defects* (i.e. incompleteness, incorrectness, inconsistency and wrong abstraction) reduces the conclusiveness of V&V results. The following two *problem statements* indicate that current safety engineering practices do not well prevent safety engineers from missing, causing and promoting specification defects:

specification defects

P1 *Lack of modelling guidance, abstraction and precision.*

Hazard analysis can determine the impact of defects in software, electronic and mechanical components on safety. The relationship between system safety goals and component requirements, however, is often imprecisely modelled. The same holds for the relationship of *hazardous failures* at system level and component defects.

functional safety

Standards, such as IEC 61508 or ISO 26262 for *functional safety*, contain extensive process guidelines but have drawbacks: these guidelines focus on providing safety evidence by process prescription and measurement (e.g. Layman et al. 2011), and neglect the use of rigorous methods and behavioural system models. This issue has been stressed by Parnas et al. (1990) and McDermid (2001, 2002). Such methods would improve the detection of specification defects because of their necessity of making safety knowledge explicit. These standards concentrate on reliability and risk analyses of a design, losing sight of validating specifications with respect to hazards. For the assessment of hazardous failures, for example, failure mode and effects (FMEA) or fault tree (FTA) analyses are common in many disciplines: defects, their causal factors and propagation are determined from a structural system model to support design for reliability (see, e.g. Dugan et al. 1992, Ericson 2005, Goddard 2000). Applications of these methods are often prone to, for example, subjective assessments with imprecise measures, and the used models lack expressiveness in capturing subtle hazards: Leveson (2012) emphasises the regard of *multiple modes of system operation* (i.e. non-linear causal chains) instead of *single modes* (i.e. linear causal chains).

Atlee and McDermid (1995) indicate an overlap of requirements validation, hazard analysis and system verification. Nevertheless, rigorous methods (e.g. Haxthausen et al. 2011, Heitmeyer et al. 1998) seem to be only used sparsely in practical safety engineering although helpful for correctness, completeness and consistency checks. Goddard (2000) notices that techniques for applying software FMEA at control system level during specification and architecture design have been largely missing in the literature. Aside from defect and hazard analyses, Schätz et al. (2005) show tool-supported modelling of specifications, whereas Utting et al. (2012) complain that there is a lack of guidance for building such specifications. Safety and reliability tools<sup>6</sup> often miss a behavioural semantics for the modelling of abstractions of technical systems. Moreover, modelling physical systems for simulation<sup>7</sup> and synthesis of realisations can be too expensive for early stage hazard analysis.

<sup>5</sup>See, e.g. Beizer 1995, Hoffmann 2013, Lutz 1993, McDermid 1991, Parnas et al. 1990, Pyle 1991, Smith 1995, van Lamsweerde 2009, Zave and Jackson 1997.

<sup>6</sup>For example, Isograph FaultTree+/Hazop+ ([www.isograph-software.com](http://www.isograph-software.com)), APIS IQ-FMEA ([www.apis.de](http://www.apis.de)), ReliaSoft Xfmea ([www.reliasoft.de](http://www.reliasoft.de)), medini analyze ([www.ikv.de](http://www.ikv.de)).

<sup>7</sup>MathWorks Simulink<sup>®</sup> ([www.mathworks.com](http://www.mathworks.com)) or Modelica<sup>®</sup> ([www.modelica.org](http://www.modelica.org)).

## 1. Motivation and Overview

### P2 Neglect of properties of the system environment.

Behavioural models, which capture interaction between the system and its environment, are rarely used to represent hazard knowledge.<sup>8</sup> Such models can help more precisely investigate observable system behaviour before and after hazardous events, and consider temporally distant or even *external causal factors* aside from system defects. Rasmussen (1997, 1999) pointed out this issue in accident analysis and risk management: he stipulates investigations of long-term combination of environment misbehaviour, maloperation and potentially defective system functionality.

For example, in the automotive domain, *safety measures* are split into *preventive* (e.g. avoidance of crash situations by means of a pre-crash safety function) and *passive* (e.g. mitigation of crash consequences by means of a car airbag) categories (Heißing and Ersoy 2007). Diaconescu (2011) reports that except for physical simulation, the automotive industry desiderates well-founded and standardised guidance for evaluation of preventive measures. Even ISO 26262 neglects to indicate specific techniques.

automation paradox

In addition, safety engineers have to deal with human factors and the *automation paradox* (Bainbridge 1983): an increased degree of automation pushes the role of human operators to perform safety-critical interactions in case of failures. Hence, automation requires qualified operators to react to failures with preventive safety measures. The paradox becomes worse if the *specification assumed by the operator* differs from the one actually realised (Wagner et al. 2010).

Both problem statements are supported by, for example, Braun et al. (2009), Broy (2012), Leveson (2012) and McDermid (2001, 2002). Leveson confirms these problems by questioning presumptions of safety engineering, such as “highly reliable [software] systems are safe,” “accidents occur from single chains or simultaneous occurrence of random events,” or “assigning blame to or punishing operators reduces incidents.”

**Experience of Safety Practitioners** I interviewed several practitioners to gain further insight into safety engineering practice. Four findings<sup>8</sup> resulting from these interviews strengthened my motivation for this work:

(F1) Producers are responsible for the safety of the entire systems they build. For example, in the automotive industry, compliance with product liability law increases the demand for safety of novel vehicle functions and control automation. (F2) Nevertheless, suppliers struggle to maintain control of assuring their subsystems to not compromise safety requirements imposed by the producer. Underspecification, erroneous documentation, intellectual property rights, and other legal and social issues complicate safety assurance among these contractors. (F3) Current practice seems to be shifting this responsibility to the human operator mainly by justifying safety by reliability. Maloperation seems to be treated with insufficient urgency. In this respect, standards for road vehicle safety, such as ISO 26262, have not yet caught up with the usual guidelines and practices of industrial machinery and aeronautics. (F4) Method

<sup>8</sup>The interview transcripts underlying these findings are summarised in Table A.7.

and tool support is an uncritical problem. Specialty models and tools will, nevertheless, gain importance because safety engineers will retain individual procedures according to safety cases. These procedures can be improved by concise and reusable system models.

**Demands of Safety Methods** For example, [Broy \(2012\)](#) outlines artefacts useful for safety-oriented requirements validation, such as a system specification, environment and defect models, assumptions about the environment, and the realised system. From the findings described above, I conclude: Safety analysis models should be (i) comprehensible by control, safety and reliability engineers of various technology domains, (ii) affordable to create, use and reuse, and (iii) scalable through modular hierarchical construction. Safety methods should provide a stop criterion to measure the completeness of analysis steps and the verification coverage for safety measures, as well as a clear procedure and work items for unimpeded adoption in practice.

A literature review (cf. Sections 3.4 and 7.2) shows that only few of the more recent safety approaches particularly combine multiple disciplines, behavioural modelling of the system and its environment, engineering guidance and logical rigour. The approaches from industry and academia, as summarised in Chapter 3, can be improved to more effectively address the questions stated above (cf. Example 1.1).

**Research Goal** The problem statements P1 and P2 motivate my research goal:

*RG Support safety-oriented requirements validation to improve the safety of a technical system.*

This goal is refined by two research questions about guidance for hazard analysis, and improving the safety of specifications and the effectiveness of safety measures:

**RQ1** *How can a technical system be modelled for hazard analysis?* RQ1.1: Which abstraction can be chosen to identify hazards, that is, to describe how the system is able to behave respecting physical, social and computational laws? RQ1.2: Which part of the system boundary has to be considered?

**RQ2** *How can hazards be identified and a safe specification be derived?* RQ2.1: How can extreme, unexpected and defective situations be derived from a specification in order to assess hazards? RQ2.2: How can a valid (i.e. sufficiently complete, correct and consistent) specification be derived which assures safety for each conforming realisation? RQ2.3: How can safety goals be decomposed with respect to the system boundary? RQ2.4: How can the required safety measures be derived?

## 1.2. Thesis Summary

Based on three preliminary studies ([Dobi et al. 2013](#), [Gleirscher 2011](#), [2013b](#)), I describe a *method for safety-oriented validation of specified and defective behaviour* observable for a *system* and its *environment*.

## 1. Motivation and Overview

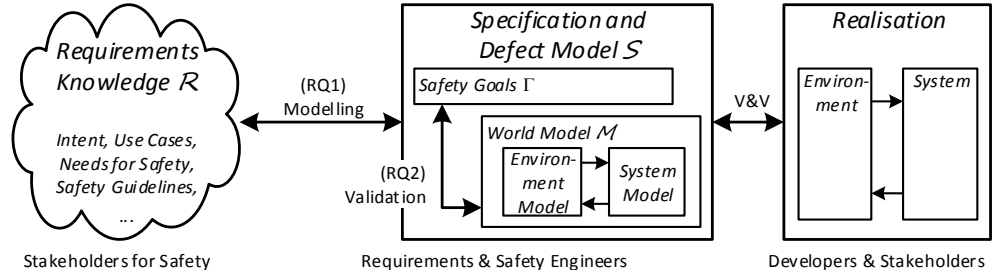


Figure 1.1.: A setting for safety-oriented requirements validation

**Concepts** The proposed method follows a human-centric viewpoint (see below in Section 3.1). This viewpoint leads to the term *behavioural safety* to emphasise

1. the *behavioural view* of a system including its safety measures
2. the *end-to-end view* of the control loop (i.a. physical processes, sensors, control software, actuators, fault tolerance mechanisms) across several disciplines
3. the regard of the *system environment* for the negotiation of *responsibilities*.

Behavioural safety addresses the problems stated in Section 1.1 through *safe conception* of functionality by using safety measures as well as *reliable design, realisation and operation* of these measures. Behavioural safety can enhance functional safety.

Figure 1.1 depicts the setting for RQ1 and RQ2: Let the set  $\mathcal{R}$ , called *requirements knowledge*, contain assertions of intended functions and properties of a technical system. From  $\mathcal{R}$ , we derive a *specification and defect model*  $S$  comprising a set of *safety goals*  $\Gamma$  and a *world model*  $\mathcal{M}$  describing the system and its environment. The world model is a *mode transition system* whose behaviour is described by *actions* producing events at the system boundary, altering states of physical or data entities. A *defect model* included in  $\mathcal{M}$  describes deviations from the specified functionality. *Mishaps* model physical states to be avoided or alleviated. *Hazards* comprise causal factors which are temporally and stochastically related to mishaps. The *system model* is an abstraction of the system to be built and operated including its potential operational defects. The *environment model* contains similar knowledge on the environment including the assets that can be harmed in certain mishaps.

As mentioned in Section 1.1, safety methods are performed on known system designs. Less demanding,  $S$  abstracts from mechanics, electrics, electronics and control software to identify potential mishaps (e.g. car accidents) and hazards (e.g. pre-crash situations), and to perform mishap alleviation (e.g. airbag) and hazard treatment (e.g. fault prevention, failure warning, safe braking).  $S$  constrains behaviour at the safety-related part of the system boundary, that is, the part which can be engaged in causal factors, hazards or mishaps. *Hazard analysis* consists in identifying hazards based on mishaps conceivable in the world model. Mode transition systems allow reasoning about the satisfiability of behavioural properties to especially determine *consistency*



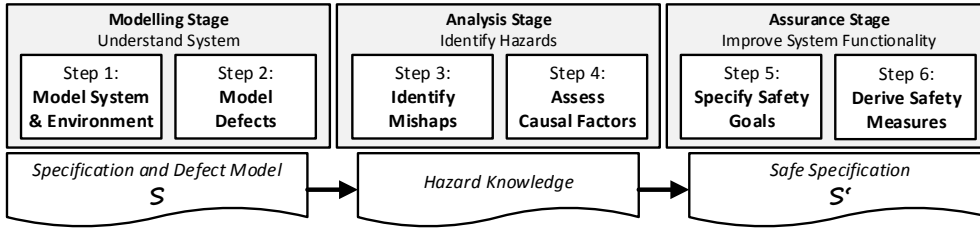


Figure 1.2.: An iterative procedure for safety-oriented requirements validation

of safety goals, probabilistic reachability<sup>9</sup> of hazardous and safe states, and stability of safe modes. See Example 1.2.

**Example 1.2 (Car Airbag)** *The temporal logic formula*

$$P_{\geq 0.9999}[\mathbf{G}(\text{crashed} \leftrightarrow \mathbf{F}^{<350\text{ms}} \text{absorbedBy}_{\text{drv}, \text{airb}})]$$

is one formalisation (see Section 2.3.2) of the behavioural property assertion

“With a chance of at least 99.99% iff a vehicle crash occurs and within at most 350 ms, an airbag has to be released to alleviate driver harm. [With the residual chance that the airbag may release too late or fail totally.]<sup>10</sup>”

which forms a safety goal for a car airbag. The airbag is a running example revived several times in the following chapters.

**Procedure** *Safety-oriented validation* of  $S$  comprises the identification of potential mishaps, hazards and causal factors as well as the planning of safety measures. Figure 1.2 depicts a three-staged procedure proposed to address RQ2. In the following, each stage is outlined by its characteristic steps, questions and tasks. First, the *modelling stage* aims to *understand the system* as operated in its environment:

**Step 1 (Model system and environment)** Which actions describe the specified behaviour?

*Task:* From the set  $\mathcal{R}$ , capture and derive use cases and specify functionality. Build up the world model  $\mathcal{M}$ . Put known *safety goals* into the set  $\Gamma$ .

**Step 2 (Derive defect model)** Which actions could be performed in an unexpected manner? Which operational defects are possible or even unavoidable? How could the system or the environment deviate from the specified behaviour?

*Task:* Make a guess of unexpected behaviour and potentially defective functionality. *Transition system patterns* (Table A.2) help model defect knowledge.

Second, the *analysis stage* aims to *identify hazards* by assessing mishaps and their causal factors:

<sup>9</sup>For example, to answer at which probability a state can be reached from a set of initial states.

<sup>10</sup>The square bracket indicates tacit information.

## 1. Motivation and Overview

**Step 3 (Identify potential mishaps)** Which mishaps are possible? Which actions can be involved in a mishap? How severe is their impact on the environment?

*Task:* Identify *mishaps* based on  $\mathcal{M}$  by using *guide words* (Table A.3).

**Step 4 (Assess causal factors)** Which actions endanger the environment by being performed hazardously? How can causal factors (i.e. operational and specification defects) of mishaps be identified and combined to define hazards?

*Task:* Use *guide words* (Table A.1) to search for hazards, investigate their causal chains and factors, and conduct their assessment in  $\mathcal{M}$ .

Third, the *assurance stage* aims to *improve system functionality* through safety measures based on the identified hazards and their causal factors:

**Step 5 (Specify safety goals)** How can hazards and mishaps be treated descriptively? Where to has responsibility as well as safety integrity to be allocated?

*Task:* Transform hazards and mishaps into new *safety goals* in  $\Gamma$ . Discover inconsistencies among safety goals and maintain realisability. Decompose new and existing *safety goals* into *safety requirements* comprising *assumptions* on the environment and *guarantees* of the system.

**Step 6 (Plan and design safety measures)** How can hazards and mishaps be treated in an operational way? How can effective safety measures be derived?

*Task:* Enhance  $\mathcal{M}$  by safety measures. *Transition system patterns* (Table A.4) help treat operational and specification defects.

The Steps 2, 3 and 4 aim for the *exploration of defects* in  $\mathcal{S}$ ; the Steps 5 and 6 use the results for the treatment of hazardous defects. This procedure supports the *iterative* transformation of  $\mathcal{S}$  into a *safe*  $\mathcal{S}'$  which, after one or more iterations, should contain the necessary safety measures. Each iteration is to be initiated at Step 1 without a model of defects or safety measures. To perform an iteration, we assume that safety measures derived in a previous iteration are converted into specified functionality and that defects treated by safety measures are hidden or removed from  $\mathcal{S}$ . Note that only in the V&V stage, we can assure that a realisation exhibits these safety measures.

## 1.3. Research Design and Outline

The present work takes a *theoretical perspective* (Part I) as well as an *applied and evaluative perspective* (Part II). Motivated by Shaw (2002), I explain the approaches chosen to put the quality of the results gained from these two perspectives into context:

First, Section 1.1 includes the results from *interviews with safety practitioners* to document a state-of-practice snapshot. Insight into the daily challenges of the engineers underpins my motivation for this thesis. Data from the transcripts are classified by content analysis and evaluated in Appendix A.2; I, however, omitted the application of grounded theory (see, e.g. Trochim and Donnelly 2008).

Second, the research questions in Section 1.1 have been derived in a way similar to the *goal-question-metric* approach (Basili et al. 1994): the metrics part is left out because the collection of quantitative data is out of scope of my studies. According to Shaw (2002), the research questions can be classified as “method for analysis,” the research contributions as “procedure or technique” and “analytic model,” and the validation results as “analysis” and “example.”

Third, the Chapters 2 and 3 summarise the *fundamentals* by giving an introduction to the terminology and definitions used throughout the thesis.<sup>11</sup> Based on these fundamentals, the Chapters 4 and 5 use *formalism* (see Sections 2.3 and 2.4) to gain precision for the introduced concepts. In Chapter 4, the reader will find a discussion of concepts for behavioural safety as an answer to RQ1 and RQ2. Chapter 5 presents a procedure as an answer to RQ2. That chapter describes the modelling, analysis and assurance stages. Details on the proposed method can be found in the Appendices A.1 and A.7. The Chapters 2, 4 and 5 close with notes and further reading.

Fourth, a *literature review* in Section 3.4 enhanced with a *systematic map* of related work aims at an appreciation of established and more recent research in the field. This review underpins the competitiveness of the proposed approach by indicating its *strengths* in Section 7.2. Details on the systematic map and a comparison with two other procedures can be found in the Appendices A.3 and A.4. This review can be a basis for an exhaustive literature survey on the field of system safety analysis.

Fifth, a *case study* in Chapter 6 demonstrates and evaluates the approach; it provides evidence to indicate efficacy in practice. The thesis, however, leaves undone a controlled statistical experiment to confidently gain stronger empirical evidence needed for performance questions on the proposed method. The study is set up and conducted along the lines of Yin (2009): The *planning* of the study is done by stating research questions (Section 1.1). The Sections 6.1 and 6.2 form a *two-case holistic design* with a single unit of analysis per case. To *prepare* the study, the procedure is documented in Chapter 5 such that the first case (Section 6.1) acts as a *pilot case* and the second case (Section 6.2) as an *approval case*. The data *collected* during application of the proposed method is directly presented in Chapter 6 and in further detail in the Appendices A.5 and A.6. The way of modelling and graphical representation chosen for the case study is compatible with SysML (Friedenthal et al. 2008) and has been exemplified for a *road vehicle adaptive cruise control* (Lochmann and Gleirscher 2009), an *automated teller machine* (Broy et al. 2012), a *coffee vending machine* (Gleirscher 2011, 2012), and *road vehicle driving dynamics and assistance*.<sup>12</sup> Section 7.1 provides an *analysis* of the study results. These results can be *shared* through the cross-case conclusions drawn in the Sections 7.3 to 7.6.

Sixth, the *discussion* in the Sections 7.2 to 7.4 points at both premises and arguments for the contribution and gives hints on how to eliminate shortcomings of the proposed method. Final conclusions and suggestions for further work are supplied in the Sections 7.5 and 7.6.

<sup>11</sup>Some of the bibliographic references, particularly in the Chapters 2 and 3, are exemplary even if not explicitly indicated. It should, however, be possible for the reader to determine from the context whether a certain reference points to an exemplary or a specific source.

<sup>12</sup>See Dobi et al. 2013, Gleirscher 2013b, Gleirscher and Fuhrmann 2012, Gleirscher et al. 2014.

# Part I.

## Theory and Approach

... the true mark of a humane society must be what it does about *prevention* of accident injuries, not the cleaning up of them afterward.

---

*(Ralph Nader 1965)*

## Technical Systems

This chapter introduces systems engineering, system theory and provides the formal preliminaries for system specification and safety engineering.

### Contents

2.1	Systems Engineering . . . . .	11
2.2	System Theory, Modelling and Specification . . . . .	13
2.2.1	System Concepts and Modelling Techniques . . . . .	13
2.2.2	System Specification and Frameworks therefor . . . . .	15
2.2.3	Defect Modelling and Analysis . . . . .	16
2.3	Formal Preliminaries . . . . .	17
2.3.1	System Modelling . . . . .	17
2.3.2	Behavioural Property Specification . . . . .	27
2.4	System Specification: A Generic Framework . . . . .	28
2.5	Notes and Further Reading . . . . .	29

## 2.1. Systems Engineering

*Technical systems* are engineered for many purposes and can have complex *functions*. Such functions must obey constraints induced by the physical dynamics of the control loop. These systems are built from various technologies such as mechanical subsystems, electrics, analogue and digital electronics, software, sensors and actuators. We speak of *mechatronics* if control is realised by electrical and electronic means (Harashima et al. 1996). Cyber-physical (Lee and Seshia 2011), robotic and software-intensive systems enhance this notion by integrating computation with physical, social and mental processes. A technical system usually has a *control subsystem* whose behaviour is governed by software, depends on and affects states of physical processes and human activity in its environment. Control systems often incorporate *models of the environment* and the system itself to facilitate state observation and to

Technical system

## 2. Technical Systems

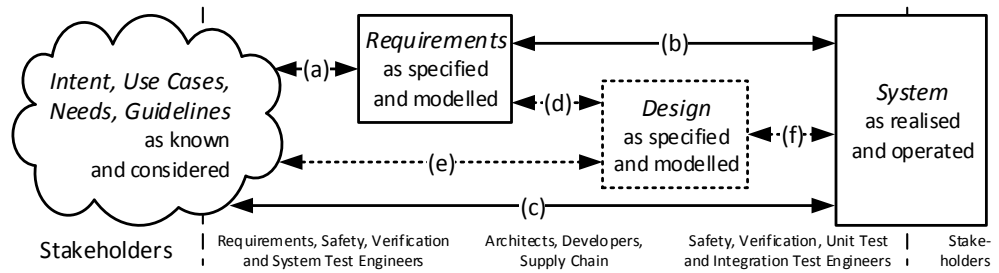


Figure 2.1.: Analytic quality assurance assuming a requirements specification but leaving the separation of requirements and design specifications optional (dashed lines)

achieve properties of control loops such as stability and reliability (Lunze 2010, Struss 2003).

In this work, we use the term *system* for any *technical system* under consideration, *environment* as a synonym for *system environment* or *system context*, and *domain* for an *application domain*. In addition, the term *world* signifies the embedding of a system into its environment.

system life cycle

Specification

A *system life cycle* includes the activities of specification, realisation, V&V and operation, conducted by several disciplines such as mechanical, electrical and software engineering (see, e.g. Ehrlenspiel and Meerkamm 2013, Jackson 1983). Many technical systems have to be long-term operated. *Specifications* representing mechanical, electrical, electronic and software *requirements*, have to be kept consistent to facilitate realisation. Both complexity and extensiveness of a system demand hierarchical decomposition of specifications into manageable pieces to perform development and *quality assurance* in isolation. This demand promotes models, methods and tool support across multiple abstractions. Long-lasting efforts, volatile customer needs and maintenance require *agility* of the involved organisations. The applied engineering methods should be *scalable* (architecture analysis and design without accidental complexity), *hierarchical and modular* (derivation of stable overall conclusions from piecewise evaluations), and *incremental* (realisation in several steps). Specifications should be *well communicable* (professional representation) and *efficiently evolvable* (support of requirements changes via traceability; Gleirscher et al. 2007). In summary, *systems engineering* refers to the planning of a system and the management of its life cycle, prior to its realisation or series production, and operation.

systems engineering

**System Quality Assurance** The *quality* of a system comprises several *properties*, for example, *functionality* and *safety* (Liggesmeyer 2009, Wallmüller 2011). van Lam-sweerde (2009) perceives *specifications* as sets of *goals* which describe such properties. Any deviation of these properties can be seen as a *defect*. Hoffmann (2013) describes constructive and analytic measures to achieve the required level of quality: *Analytic measures* assess the results on the way to realisation (Figure 2.1a–f) and can be continuously applied as a combination of several techniques (e.g. code inspection, design reviews; Gleirscher et al. 2014). Such measures can be performed in reaction to known

goal

## 2.2. System Theory, Modelling and Specification

but previously unexpected defects (e.g. debugging) or as an investigation of presumed defects (e.g. testing). *Constructive measures* such as *systems, requirements and software engineering*, guide the design decisions on the way to realisation. IEEE Std. 830 (1998) and ISO Std. 9126 (2001) guide through the quality assurance of software specifications and implementations.

*Requirements engineering* fosters the achievement of a *valid specification*, its maintenance and communication (van Lamsweerde 2009) through the conduct of:

Requirements engineering

1. *Elicitation*: Exploiting sources of information including stakeholders.
2. *Analysis*: Deriving knowledge of the system environment, the application domain (domain analysis), the system boundary (context analysis) and the problem (usage analysis); solution restriction; structuring; evaluation of use, priorities, cost-efficiency and feasibility; finding consensus; removal of inconsistencies and conflicts; revealing rationale; negotiation and decision.
3. *Specification*: Documenting committed *goals* (van Lamsweerde 2009), requirements and *use cases* (Cockburn 2000) to constrain system behaviour; an environment model to determine the system boundary; functions, conceptual and physical constraints of entities and their states as a basis for realisation.
4. *Assurance*: Requirements validation (Figure 2.1a) and verification of the system (Figure 2.1b,d).

We will use the following terms: *Goal* for any “prescriptive statement of intent,” *guarantee* for a goal to be satisfied under the responsibility of the system and *assumption* for a goal whose satisfaction falls under the responsibility of the environment (van Lamsweerde 2009). The term *requirement* refers to an *assumption/guarantee pair*. The word *specification* (also *requirements specification*) is used for both the requirements engineering task and its artefact.

## 2.2. System Theory, Modelling and Specification

*System, control and automation theories*<sup>1</sup> describe technical systems in many areas of life. *System models* are artefacts which engineers use throughout the life cycle of a technical system to specify and analyse its properties.

### 2.2.1. System Concepts and Modelling Techniques

How can a system model assist in the life cycle? Models help understand, communicate and reason about system *properties*. Models can support *analytic steps* such as abstract interpretation, diagnosis (Reiter 1987), model checking (Baier and Katoen 2008), qualitative spatio-temporal reasoning (Reiter 2001, Struss 2003), runtime

properties

<sup>1</sup>See, e.g. Forrester 1961, Luhmann 2006, Lunze 2010, Paynter 1960, von Bertalanffy 1957, Wang 1964, Wiener 1965.

## 2. Technical Systems

transformation verification (Leucker and Schallhart 2009), simulation<sup>2</sup>, testing or theorem proving.<sup>3</sup> Models can assist with *constructive steps* (also *transformations*) such as decomposition, refinement (Broy and Stølen 2001), composition and abstraction. Both kinds of steps need to be *hierarchically performed* to scale up to large systems and to regard multiple levels of abstraction. Some methods care for *property preservation* across these abstractions and for *modularity* of these steps to reduce dependencies.

**What has to be modelled for the chosen purpose?** According to Beizer (1995), system “models are mental tools and [...] there is no fundamental right or wrong with them.” These models can refer to any concepts of thought or perception (Stachowiak 1973, Wittgenstein 1922). Modelling includes the definition of *variables*, *types* and *entities* to observe or measure (Trochim and Donnelly 2008). The variables can be interpreted for *states of physical or conceptual* entities such as processes (Forrester 1961), business or data objects. Variables can reify *channels* (Broy and Stølen 2001) or *shared phenomena* (Jackson 2001) modelling the interfaces and *interaction* of these entities. Entities can be *passive*, with variables forming their state space, or *active*, monitoring and reactively manipulating variables via their interface and changing their own state (Agha and Hewitt 1985, van Lamsweerde 2009).

behavioural properties Modelling can help express *behavioural properties* of a system, its *functionality*, role or purpose in its environment (Umeda et al. 1990): *system functions* relate *dependent* (also output, written, controlled or manipulated) with *independent* (also input, read, monitored or observed) variables.<sup>4</sup> It has been argued to describe such functions by specifying *behaviour*, either using behavioural properties or executable models (Broy and Stølen 2001). *Transitions* between states can represent system *actions* and structure complex behaviours. While every entity can incorporate states, transitions can only be performed by active entities. The abstraction of a system to its *boundary* is known as *behavioural*, *interface* or *black-box* view (Wiener 1965). Generic behavioural properties are, for example, causality, indeterminacy, liveness, probability, realisability and safety.<sup>5</sup>

behaviour

structural properties Aside from behavioural properties, *structural properties* can be modelled as a *structural* or *glass-box* view to understand distribution and causal chains (Broy and Stølen 2001). According to Broy (2005, 2010), *architecture* denotes the *hierarchical decomposition* of a system into components, the definition of their interfaces and their connection via *channels*. Interaction takes place via transmission of information (e.g. communication by message or signal flow), energy (e.g. force flow or mechanical load, electric voltage, light) and material (e.g. fluid, workpiece or data flow, electric current) at defined connection points or areas (Paynter 1960). Generic structural properties are, for example, causal nets, communication, cooperation, coordination, modularity and resource optimisation.

structural properties

hierarchical decomposition

abstraction We use the term *abstraction* for the act of *reducing variable types and rela-*

<sup>2</sup>See, e.g. Ljung 1998, Sargent 1999, Schneider et al. 2001, Waters and Ponton 1989.

<sup>3</sup>See, e.g. Hoffmann 2013, Liggesmeyer 2009, Wallmüller 2011.

<sup>4</sup>See, e.g. Courtois and Parnas 1993, Parnas and Madey 1995.

<sup>5</sup>See, e.g. Broy and Stølen 2001, Lamport 2002, Manna and Pnueli 1995, Rushby 1994.



*tionships among entities* to a low number of distinguishable values being discretely observed, thus reducing the world under consideration to a *qualitative model*.<sup>6</sup>

**How can a system model be represented?** Many *modelling techniques* are available to represent the above concepts (Davis 1988, Friedenthal et al. 2008). We can distinguish *structural, behavioural* and *mixed* techniques.

Structural techniques specialise in *conceptual or data modelling* of passive entities (Baader et al. 2003, Chen 1976), regard the *architecture* of active entities (DeMarco 1979, Jackson 1983) or consider both aspects (Kifer et al. 1995).

Behavioural techniques focus on active entities using interaction sequences (Harel and Marelly 2003), input/output functions (Parnas and Madey 1995) or stream functions (Broy 2005). *Transition systems* provide a versatile way to model active entities with output capabilities, known as *transducers*, and without output capabilities, known as *acceptors*.<sup>7</sup> *Action languages* provide a formal logic representation of transition systems (Reiter 2001, Thielscher 2011).

Among the mixed techniques, *data flow models* put emphasis on communication within a distributed system (Kahn 1974). *Control flow and process models*<sup>8</sup> focus on sets of actions and preorders of these sets. The combination of data and control flow as well as hierarchical decomposition enables the investigation of concurrency and architecture (Beizer 1995, Broy 2010).

Models can carry formal semantics using mathematical structures such as *traces* (van Glabbeek 2001), *streams* (Broy and Stølen 2001), *situations* (Reiter 2001), KRIPKE *structures* (Baier and Katoen 2008), metric and probability spaces and differential equations (Paynter 1960, Wang 1964). For example, system, control and automation theories apply structural techniques to visualise control loops formalised by differential equations (Lunze 2010) or bond graphs (Secchi et al. 2007).

### 2.2.2. System Specification and Frameworks therefor

Requirements engineering (Section 2.1) can use system models: We can distinguish *operational* (also procedural, executable, explicit, model-based) from *descriptive* (also declarative, constraining, implicit, property-based) *styles of specification* combinable with the above modelling techniques. Transducers are given operational semantics and goal graphs (Kelly 1998, van Lamswerde 2009) carry descriptive semantics. Sequence charts (Harel and Marelly 2003), transition systems and architectures (Broy 2005) can be used with both styles.

Regarding the structure and environment of a system, the *pre-/postcondition* style, a variant of both styles, started as a way to attach meaning to programs (Floyd 1967) and was enhanced for systems as the *assumption/guarantee* style (e.g. Broy 1998, van Lamswerde 2009): a system has to fulfil a guarantee if its environment fulfils the

<sup>6</sup>See, e.g. Forrester 1961, Ostroff 1997, Sampath et al. 1996, Struss 2003, Wonham 1976.

<sup>7</sup>See, e.g. Baier and Katoen 2008, Harel and Politi 1998, Hopcroft et al. 2006, Kaynar et al. 2010, Mealy 1955, Moore 1956.

<sup>8</sup>See, e.g. Agha and Hewitt 1985, Hoare 1985, Milner 1973, Petersen 1981.

## 2. Technical Systems

### Specification framework

assumption associated with this guarantee. This style simplifies modular verification and helps clarify *responsibilities* among contractors.

*Specification frameworks* (Zave and Jackson 1997) aim at correct and concise system description including careful underspecification to leave freedom to developers. Specifications can be represented *informally* (e.g. using natural language or sketches), in a *controlled* way (e.g. abstraction into formal, semi-structured grammar; Dwyer et al. 1999) and *formally* (e.g. using temporal logic and transition systems; Baier and Katoen 2008). *Formal methods* for specification and control system design often contain elements of both styles of specification to allow a comprehensive description of intent behind and usage of a system, for example, 4-Var (Parnas and Madey 1995), B (Abrial 1998), CORE (see, e.g. Jureta et al. 2008), FOCUS (Broy and Stølen 2001), Gist (Feather 1987), KAOS (van Lamswerde 2009), Problem Frames (Jackson 2001), SCR\* (Heitmeyer et al. 1998), SpecTRM (Leveson et al. 1998), TLA<sup>+</sup> (Lamport 2002), TTM/RTTL (Ostroff 1997), Z (Spivey 2008), the method described by Hatley and Pirbhai (1987) and hybrid<sup>9</sup> models for behavioural verification. Many of these methods support hierarchical decomposition.

### 2.2.3. Defect Modelling and Analysis

#### defect fault

A failure can be perceived as a *deviation* of an *actually observed* behaviour from the one *ideally specified*. Faults act as causes of such deviations. Gärtner (1999), Liggesmeyer (2009) and Wallmüller (2011), for software systems, and Struss and Fraracci (2011), for physical systems, *classify defects* according to their position in causal chains based on a structural view. Consequently, *faults* may initiate causal chains containing *erroneous states* and result in *failures*. According to IEEE Std. 610 (1990), the life cycle of a defect essentially starts with a programming *error* or hardware *fault* ending up in either a visible *failure* or being intercepted and handled. Taking a behavioural view, ANSI and IEEE Std. 792 (1983) perceives a *software defect* as a “software-related discrepancy between a computed, observed, or measured value or condition and the true, specified, or theoretically correct value or condition.”

#### failure

#### defect model

For defect models, two *system views* can be considered: *actual* and *ideal behaviour*. Both may use system modelling. Hence, a *defect model* is a system model which describes known or imaginable defects in the chosen modelling technique. Such a model allows the estimation of failure impacts, defect treatment at runtime (Gärtner 1999) and safety analysis of many defective realisations of a system at once. Defect models are used in reliability (Dugan et al. 1992) or fault tolerance (Gärtner 1999) analyses, testing, diagnosis (Reiter 1987, Sampath et al. 1996), runtime verification and fault localisation, using techniques such as fault injection or mutation.

Sampath et al. approach *diagnosis* by modelling a technical system as a transition system and building an abstraction of this model based on the available sensors. Using this abstraction, the authors distinguish observable from unobservable events (i.e. action effects). Unobservable events can include defects. A system is *diagnosable* if it is possible to detect, with a finite delay, occurrences of defects using the record of

<sup>9</sup>See, e.g. Henzinger 2000, Lee and Seshia 2011, Platzer 2010, Schneider et al. 2001.

observed events. The task to identify such occurrences, given partial observations, is performed by a transition system called the *diagnoser*.

Similarly, defect models can stem from the abstraction of system designs. For example, Reese and Leveson (1997) apply *qualitative modelling* in safety analysis to calculate the propagation of input *deviations* of a program to deviations in its outputs. Struss and Fraracci (2011) apply a deviation model as a defect model of a physical system to perform diagnosis. *Negative testing* (Das et al. 2012) and *fault injection* (Peikenkamp et al. 2006) are controlled ways of diagnosis enforcing defective system states to perform fault localisation and assess fault tolerance. Moreover, *fault model-based testing* uses modified transition systems.<sup>10</sup> Gärtner (1999) introduces *fault actions* into his model for reasoning about fault detection and tolerance at runtime. Diagnosis has been applied to software debugging: failed *test runs* are analysed to aid in localising faults (Mayer and Stumptner 2007). Obtaining a defect model by *mutation analysis* means applying erroneous modifications to a system model. In software testing, mutations are modifications of source code which aid in finding faults and defining test stop criteria (Liggesmeyer 2009).

Botaschanjan and Hummel (2009) propose extension patterns for *modular data flow specifications* to derive a defect model for control systems (Breitling 2000, Das et al. 2012, Pister 2008). These extensions capture sensor and actuator faults, and failure modes, claimed to be applicable to FMEA. Das et al. (2012) use *fault variables* to model the indication of faults, for example, in FTA.

## 2.3. Formal Preliminaries

This section explains the formal modelling and specification of technical systems applied in the remainder of this work.

### 2.3.1. System Modelling

Let  $\mathcal{V} = \{v_1, v_2, \dots, v_n\}$  be a set of *variables* and  $\mathbb{T}$  be a set of countable types including the type  $\mathbb{B} \triangleq \{\perp, \top\}$  as the set of truth values. Given a map  $\mathbf{type} : \mathcal{V} \rightarrow \mathbb{T}$  for type declaration, we write  $v : T$  to declare  $\mathbf{type}(v) = T$  for  $v \in \mathcal{V}$  and  $T \in \mathbb{T}$ .

**Definition 2.1 (State)** *Given the set  $\mathbb{U} \triangleq \bigcup_{T \in \mathbb{T}} T$ , a state is a map  $\sigma : \mathcal{V} \rightarrow \mathbb{U}$  which fulfils  $\forall v \in \mathcal{V} : \sigma(v) \in \mathbf{type}(v)$ .* ⌋

The expression  $[v_1 \mapsto x_1, \dots, v_n \mapsto x_n]$  stands for a state associated with the values  $x_1, \dots, x_n$ . The set of all states over  $\mathcal{V}$  is called the (observable) *state space* of  $\mathcal{V}$ . Let  $\Sigma$  be a set of states in the remainder of this section.

**Definition 2.2 (Run)** *Let  $n \in \mathbb{N}_0$ . A run (of length  $n$ ) is a map  $\rho : \{0, 1, \dots, n\} \rightarrow \Sigma$ . For any set of runs (with equal state  $\rho(0)$ ), we also use the term behaviour (computation tree<sup>11</sup>).* ⌋

<sup>10</sup>See, e.g. Esser and Struss 2007, Godskesen 1999, Petrenko 2000.

<sup>11</sup>See, e.g. Baier and Katoen (2008) or Pnueli and Kesten (2002) on branching temporal logic.

## 2. Technical Systems

⌈ **Definition 2.3 (Behavioural Spectrum)** Let  $[\mathcal{V}]_i$  denote the set of all runs over  $\mathcal{V}$  of length  $i \in \mathbb{N}_0$ . We write  $[\mathcal{V}] \triangleq \bigcup_{i \in \mathbb{N}_0} [\mathcal{V}]_i$  to denote the (infinite, discrete and timed) behavioural spectrum over  $\mathcal{V}$ .

⌈ **Definition 2.4 (State Constraint)** A state constraint is a quantifier-free formula  $c$  of first-order logic only using variables in  $\mathcal{V}$ .

**Interpretation of State Constraints** Let  $\mathcal{C}$  be the set of all state constraints. Suppose that the map  $\mathcal{I} : \mathcal{C} \times \Sigma \rightarrow \mathbb{B}$  interprets and evaluates any state constraint based on substituting its variables with values determined by a given state. For convenience, we use a state constraint  $c$  as a predicate  $c : \Sigma \rightarrow \mathbb{B}$  on states: the biconditional  $c(\sigma) \leftrightarrow \mathcal{I}(c, \sigma)$  then explains when  $c$  holds for a state  $\sigma$ .

**Projection** Given a state  $\sigma$  and a set  $V \subset \mathcal{V}$ , we denote by  $\sigma|_V : V \rightarrow \mathbb{U}$  the projection of  $\sigma$  onto the state space of  $V$  with  $\forall v \in V : \sigma|_V(v) = \sigma(v)$ . Accordingly, we obtain the projection of  $\Sigma$  by  $\Sigma|_V \triangleq \{\sigma|_V \mid \sigma \in \Sigma\}$  and the projection of a run  $\rho$  by

$$\rho|_V : \{0, 1, \dots, n\} \rightarrow \Sigma|_V \quad \text{with} \quad \forall i \in \{0, 1, \dots, n\} : \rho|_V(i) = \rho(i)|_V$$

⌈ **Definition 2.5 (Action)** Given  $V \subset \mathcal{V}$  and abbreviating  $\Sigma \rightarrow \Sigma|_V$  with  $\mathcal{E}_V$ , an action

$$(\text{pre}, \text{delay}, \text{trig}, \text{prio}, \pi, \text{post}) \in \mathcal{C} \times \mathbb{N}_0 \times \mathcal{C} \times [0, \infty) \times (0, 1] \times \mathcal{E}_V$$

consists<sup>12</sup> of

- a state constraint *pre* specifying when its execution is possible
- a parameter *delay* defining the least number of time steps being inactive
- a state constraint *trig* specifying when its execution is necessary
- a parameter *prio* defining the priority of its choice among other actions
- a parameter  $\pi$  defining the probability of its choice among other actions
- a total function *post* specifying the effects of its execution.

⌋ The semantics of action specifications is described below on page 19.

**Properties and Use of Variables** Let  $\mathcal{V}_m \subset \mathcal{V}$  and  $\mathcal{V}_e \subset \mathcal{V}$  be sets of variables with  $\mathcal{V}_m \cap \mathcal{V}_e = \emptyset$ . We call  $\mathcal{V}_m$  the set of *mode channels* and  $\mathcal{V}_e$  the set of (*event channels*). We divide  $\mathcal{V}_e$  into a set  $\mathcal{V}_{ef}$  of *functional channels* and a set  $\mathcal{V}_{ec}$  of *control channels*. Let  $t : \mathbb{N}_0 \in \mathcal{V}$  with  $\rho(n)(t) = n$  for each run  $\rho$  to track *global time*.

⌈ **Definition 2.6 (Mode Transition System, MTS)** An MTS  $(\mu, \mathbb{A}, \Delta, \mu_0, \Sigma_0)$  is defined by

- a set  $\mu$  of labels denoting modes (of operation)

<sup>12</sup>We can use short cuts for the tuples, see Example 2.1.

- a set  $\mathbb{A}$  of actions according to Definition 2.5
- a relation  $\Delta \subseteq \mu \times \mathbb{A} \times \mu$  describing mode transitions
- a set  $\mu_0 \subseteq \mu$  of initial modes and
- a set  $\Sigma_0 \subseteq \Sigma$  of initial states.

Well-formedness of a mode-specific action set is described below on page 20.  $\lrcorner$

Let  $\mathbb{M}$  be the set of all MTSs and  $\mathcal{M} = (\mu, \mathbb{A}, \Delta, \mu_0, \Sigma_0)$  be an MTS in the remainder of this section. For  $\mathcal{M}$ , we have two variables  $m_{\mathcal{M}} : \mu \in \mathcal{V}_m$  and  $t_{\mathcal{M}} : \mathbb{N}_0 \in \mathcal{V}$  to track modes and local time. For  $\mu_0$  and  $\Sigma_0$ , we assume that

$$\forall m \in \mu_0 \exists \sigma \in \Sigma_0 : \sigma(m_{\mathcal{M}}) = m \quad (2.1)$$

Furthermore, suppose that we have two maps  $\text{mon} : \mathbb{M} \rightarrow \mathcal{P}(\mathcal{V})$  and  $\text{ctr} : \mathbb{M} \rightarrow \mathcal{P}(\mathcal{V} \setminus \mathcal{V}_m)$  which associate sets of monitored and controlled variables with each MTS.

**Definition 2.7 (Mode, Event)** Given a run  $\rho$ ,  $m \in \mu$  and a variable  $m_{\mathcal{M}} \in \mathcal{V}_m$ , we say that the mode  $m$  is active at time  $t$  iff  $\rho(t)(m_{\mathcal{M}}) = m$ . Consistent with Formula (2.1), we require that  $\rho(0)(m_{\mathcal{M}}) \in \mu_0$ . Furthermore, we denote the set of all actions available in mode  $m$  by  $\mathbb{A}_m$   $\lrcorner$

$$\mathbb{A}_m \triangleq \{a \in \mathbb{A} \mid \exists m' \in \mu : (m, a, m') \in \Delta\}$$

Then, for any action  $a$ ,

$$a.\text{pre} = \begin{cases} \bigwedge_{b \in \mathbb{A}_m \setminus \{a\}} \neg b.\text{pre}, & \text{if } \mathbb{A}_m \neq \emptyset, \\ \top, & \text{otherwise,} \end{cases} \quad \text{abbreviated by } a.\text{pre} = *,$$

which stands for the “default case” or “else”. The same pattern applies to  $a.\text{trig}$ . Given an event  $e : \mathcal{V}_e \rightarrow \mathbb{U}$ , we say that we observe  $e$  at time  $t$  iff  $\rho(t)|_{\mathcal{V}_e} = e$ .  $\lrcorner$

**Semantics of Action Execution** Given an action  $a \in \mathbb{A}_m$ ,  $a.\text{delay}$  defines the shortest period of activity of mode  $m$  after which  $a$  gets available. The state constraint  $a.\text{pre}$  defines when it is *possible* (also permissive) for  $\mathcal{M}$  to perform  $a$  whereas  $a.\text{trig}$  defines when it is *necessary* (also coercive) to perform  $a$ . For a run  $\rho$ , the evaluation of

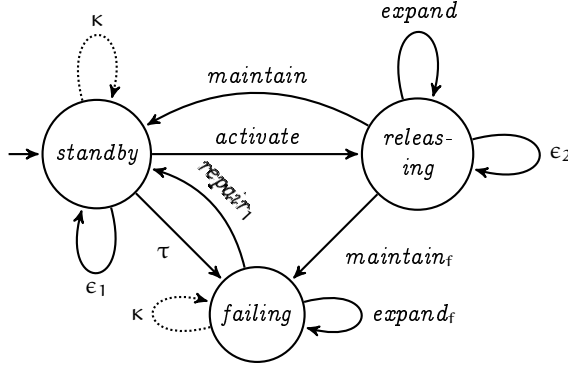
$$\mathcal{I}(a.\text{pre} \wedge a.\text{delay} \leq t_{\mathcal{M}} \wedge a.\text{trig}, \rho(t))$$

determines the necessity of  $a$  in the state at time  $t$  (cf. Table 2.1 below). The parameter  $a.\text{prio}$  determines whether  $a$  is among the possible actions with the highest (i.e. smallest) priority. We call a set  $A \subseteq \mathbb{A}_m$  a *bunch* iff all actions in  $A$  are equal in their pre, delay, trig and prio elements. Let  $a \in A$ . Given that  $A$  is associated with a probability distribution, the parameter  $a.\pi$  defines the probability of  $a$  to be chosen from  $A$  and of  $a.\text{post}$  to be performed. The function  $a.\text{post}$  maps fractions of states to *successor states* and can be written as a formula coding this map via assignment or temporal operators and equality (cf. Section 2.3.2). This procedure is formalised in Corollary 2.1 at page 22. Finally, the runs in  $\llbracket \mathcal{V} \rrbracket$  result from consecutively executed actions of a modelled or realised system observable through  $\mathcal{V}$ .

## 2. Technical Systems

**Classes of Actions** For each action  $a \in \mathbb{A}$ , we assume that  $a.post : \mathcal{E}_{ctr(\mathcal{M})}$ . In particular,  $NOP : \mathcal{E}_{ctr(\mathcal{M})}$  is a map which retains the value of any variable  $v \in ctr(\mathcal{M}) \cap (\mathcal{V} \setminus \mathcal{V}_{ec})$ , and retains or possibly quiesces<sup>13</sup> any control channel  $v \in ctr(\mathcal{M}) \cap \mathcal{V}_{ec}$ . Table 2.1 classifies *basic actions* executable by  $\mathcal{M}$ : We consider the set  $\mathbb{A}_f$  of functional actions which affect *functional channels* and the set  $\mathbb{A}_c$  of control actions which produce events on *control channels*. The effect post of a functional or idle action always carries an implicit conjunct ( $t_{\mathcal{M}} := t_{\mathcal{M}} + 1$ ). For a control, spontaneous or time-triggered action, post is implicitly extended by the conjunct ( $m_{\mathcal{M}} := m' \wedge t_{\mathcal{M}} := 0$ ). Example 2.1 applies Definition 2.6 to the modelling of a car airbag system.

□ **Example 2.1 (MTS of a Car Airbag)**



**Remarks:**

We set

$ctr(\text{Airbag}_S) \cup \text{mon}(\text{Airbag}_S) = \{$   
 $\text{energy} : \mathbb{B}, \text{crashed} : \mathbb{B},$   
 $\text{gas} : \{-1 \equiv \text{defective}, 0 \equiv$   
 $\text{empty}, 1, \dots, 5 \equiv \text{full}\},$   
 $\text{released} : \{0 \equiv \text{no}, 1, \dots, 5\}$   
 $\}.$

The transition diagram shows that  $\mu_0 = \{\text{standby}\}.$

The function  $NOP$  stands for “no effect” and is described above.

**Action Specifications of the MTS  $\text{Airbag}_S$  (only actions in solid lines)**

Label	pre	delay	trig	prio	$\pi$	post
activate	$\text{energy} \wedge \text{crashed}$	0	crashed	2	1	$NOP$
$\epsilon_1$	$\neg \text{crashed}$	0	$\top$	2	1	$NOP$
$\tau$	$\top$	0	$\perp$	2	1	$NOP$
expand	$\text{gas} \neq \text{empty}$	0	$\top$	2	1	$\text{released} += 1, \text{gas} -= 1$
$\epsilon_2$	$\text{gas} = \text{empty}$	0	$\top$	2	1	$NOP$
maintain	$\text{gas} \neq \text{full}$	10	$\perp$	2	.95	$\text{gas} := \text{full}$
maintain <sub>f</sub>	$\text{gas} \neq \text{full}$	10	$\perp$	2	.05	$\text{gas} := \text{defective}$
expand <sub>f</sub>	$\top$	0	$\perp$	1.5	.01	$\text{released} += 1, \text{gas} -= 1$
repair <sub>1</sub>	$\top$	0	$\perp$	1.5	1	$\text{released} := \text{no}, \text{gas} := \text{full}$

**Well-formedness and Construction of Modes** We call  $\mathcal{M}$  *basic* if  $\mathbb{A}$  only consists of  $\mathbb{A}_f$  and  $\mathbb{A}_c$ , with  $\mathbb{A}_f \cap \mathbb{A}_c = \emptyset$ . Given that  $\mathcal{M}$  is a basic MTS and for a mode  $m \in \mu$ , we only consider definitions of  $\mathbb{A}_m$  which fulfil the following conditions:

1. For an action  $a \in \mathbb{A}_m$ , the formula

$$(a.trig \rightarrow (a.pre \wedge t_{\mathcal{M}} \geq a.delay)) \vee (a.pre \wedge a.trig \wedge t_{\mathcal{M}} \geq a.delay)$$

<sup>13</sup>For example, assigning to  $v$  a value not in  $\text{type}(v)$ .

Action Class	Action Specification ( $m, a, m'$ ) $\in \Delta$ with $a =$	Enabling Conditions (i.e. guards, triggers and other conditions) for execution in state $\sigma$
functional ( $\mathbb{A}_f \subseteq \mathbb{A}$ )	(pre, 0, $\top$ , prio, $\pi$ , post) and $m = m'$	Execution is <i>necessary</i> iff $\mathcal{I}(\text{pre}, \sigma)$ . <i>Coerciveness</i> is mandatory by $\text{trig} = \top$ . post only uses variables in $\mathcal{V} \setminus (\mathcal{V}_{ec} \cup \{t, m_{\mathcal{M}}, t_{\mathcal{M}}\})$ .
control ( $\mathbb{A}_c \subseteq \mathbb{A}$ )	(pre, delay, trig, prio, $\pi$ , post) and $m \neq m'$	Execution is <i>possible</i> iff $\mathcal{I}(\text{delay} \leq t_{\mathcal{M}} \wedge \text{pre}, \sigma)$ and <i>necessary</i> iff $\mathcal{I}(\text{delay} \leq t_{\mathcal{M}} \wedge \text{trig}, \sigma)$ . <i>Coerciveness</i> can be modelled through $\text{pre} = \text{trig}$ . post only uses variables in $\mathcal{V} \setminus (\mathcal{V}_{ef} \cup \{t, m_{\mathcal{M}}, t_{\mathcal{M}}\})$ .
idle ( $\epsilon \in \mathbb{A}_f$ )	(pre, 0, $\top$ , prio, $\pi$ , NOP)	$\epsilon$ -completion ( $\text{pre} = *$ ), universally ( $\text{pre} = \top$ ) or individually enabled, without effects ( $\text{post} = \text{NOP}$ ).
spontaneous ( $\tau \in \mathbb{A}_c$ )	( $\top$ , 0, $\perp$ , prio, 1, NOP)	Execution is <i>always possible</i> , quiescing or retaining values of variables in $\mathcal{V}_{ec}$ , that is, $\text{pre} = \top$ , $\text{trig} = \perp$ , $\text{post} = \text{NOP}$ , $\text{delay} = 0$ and $\pi = 1$ .
time-triggered	(pre, delay, $\top$ , prio, $\pi$ , post)	Only applicable to control actions ( $\in \mathbb{A}_c$ ), with $\text{delay} \geq 1$ and $\text{trig} = \top$ .
chaos ( $\kappa$ )	( $\top$ , 0, *, *, *, *)	$\text{trig} = *$ denotes $\kappa$ -completion, $\text{prio} = *$ and $\text{post} = *$ denote any priority and arbitrary effects, $\pi = *$ completes any probability distribution.

Table 2.1.: Classes of basic actions to be found in  $\mathcal{M}$ 

is *satisfiable*. If pre, trig and  $t_{\mathcal{M}}$  are evaluated at time  $t$ , the results of post in time  $t + 1$  only depend on valuations at time  $t$ . Moreover, we require

$$\forall a_f \in \mathbb{A}_m \cap \mathbb{A}_f, a_c \in \mathbb{A}_m \cap \mathbb{A}_c : a_f.\text{prio} \geq a_c.\text{prio}$$

- $\mathbb{A}_m$  can be *totalised* unless pre, trig and delay cover the state space. For subsets of the state space not covered (*residual*), the *chaos action*  $\kappa \notin \mathbb{A}_m$  is implicitly enabled ( $\kappa$ -completion).  $\mathbb{A}_m$  can then be
  - *strongly underspecified*: pre and delay only achieve partial coverage; *strong*  $\kappa$ -completion applies to the residual.
  - *weakly underspecified*: complete state space coverage by pre and delay, but trig is incongruent with pre and delay; *weak*  $\kappa$ -completion applies.
  - *fully specified*: complete state space coverage by trig and delay which removes weak underspecification;  $\kappa$ -completion ceases to apply.

Independent of totality, we assume variables not referenced by post or  $\text{ctr}(\mathcal{M})$  to be unchanged or quiesced, equivalent to using NOP ( $\epsilon$ -completion).

- $\mathbb{A}_m$  can be *non-deterministic*<sup>14</sup>, that is, multiple actions can become enabled if their elements pre, delay, trig and prio overlap instead of partitioning the state space, and their  $\pi$  parameters equal 1.

<sup>14</sup>An MTS can generate two runs which contain the same monitored events (i.a. stimuli) but exhibit different controlled events (i.a. reactions).

## 2. Technical Systems

4. To a set  $\mathbb{A}_\pi \subseteq \mathbb{A}_m$  of non-deterministic actions congruent<sup>15</sup> in their pre, delay, trig and prio can be assigned a *probability*<sup>16</sup> distribution on their effects and target modes.  $\kappa$ -completion extends to the probabilistic case such that

$$\sum_{a \in \mathbb{A}_\pi \cup \{\kappa\}} a.\pi = 1$$

if there are no effects specified for a portion of  $\mathbb{A}_\pi$ 's probability distribution.

In analogy to Definition 2.3, we characterise the runs of  $\mathcal{M}$  as follows:

**Corollary 2.1 (Runs of an MTS)** *Given the set  $\mathbb{A}_m$  by Definition 2.7, suppose that*

$$\begin{aligned} \text{available}(\sigma) &= \{a \in \mathbb{A}_{\sigma(m, \mathcal{M})} \mid (t_{\mathcal{M}} \geq a.\text{delay})(\sigma)\} \\ \text{necessary}(\sigma) &= \{a \in \text{available}(\sigma) \mid (a.\text{pre} \wedge a.\text{trig})(\sigma)\} \\ \text{possible}(\sigma) &= \begin{cases} \text{necessary}(\sigma), & \text{if necessary}(\sigma) \neq \emptyset \\ \{a \in \text{available}(\sigma) \cup \{\kappa\} \mid a.\text{pre}(\sigma)\}, & \text{otherwise} \end{cases} \end{aligned}$$

Let  $\equiv_B \subseteq \mathbb{A} \times \mathbb{A}$  and  $\leq_{\text{prio}} \subseteq \mathcal{P}(\mathbb{A}) \times \mathcal{P}(\mathbb{A})$  be relations such that

$$\begin{aligned} a_1 \equiv_B a_2 &\Leftrightarrow a_1.(\text{pre}, \text{delay}, \text{trig}, \text{prio}) = a_2.(\text{pre}, \text{delay}, \text{trig}, \text{prio}) \\ A_1 \leq_{\text{prio}} A_2 &\Leftrightarrow \forall a_1 \in A_1, a_2 \in A_2 : a_1.\text{prio} \leq a_2.\text{prio} \end{aligned}$$

Suppose that  $\inf_R$  and  $\sup_R$  return all least upper and greatest lower bounds with respect to a partial order  $R$ . Given the equivalence  $\equiv_B$  and the partial order  $\leq_{\text{prio}}$ , we define

$$\begin{aligned} \text{bunches}(\sigma) &= \sup_{\subseteq} \{A \subseteq \text{possible}(\sigma) \mid \forall a_1, a_2 \in A : a_1 \equiv_B a_2\} \\ \text{cbunches}(\sigma) &= \{A \mid (A \in \text{bunches}(\sigma) \wedge \sum_{a \in A} a.\pi = 1) \vee \\ &\quad (\exists A' \in \text{bunches}(\sigma) : A' \cup \{\kappa\} = A \wedge \sum_{a \in A'} a.\pi < 1)\} \\ \text{enabled}(\sigma) &= \bigcup_{\leq_{\text{prio}}} \inf_{\leq_{\text{prio}}} \text{cbunches}(\sigma) \end{aligned}$$

to return the set of actions enabled in a state  $\sigma \in \Sigma$ . Let  $i \in \mathbb{N}_0$  and  $\frown: \llbracket \mathcal{V} \rrbracket_i \times \Sigma \rightarrow \llbracket \mathcal{V} \rrbracket_{i+1}$ . Then, we obtain the set of all runs of  $\mathcal{M}$  by the inductive definition

$$\begin{aligned} \llbracket \mathcal{M} \rrbracket_0 &= \{\rho \in \llbracket \mathcal{V} \rrbracket_0 \mid \rho(0) \in \Sigma_0\} \\ \llbracket \mathcal{M} \rrbracket_{i+1} &= \{\rho \frown \sigma \mid \rho \in \llbracket \mathcal{M} \rrbracket_i \wedge \sigma \in \Sigma \wedge \\ &\quad \exists a \in \text{enabled}(\rho(i)) : \sigma|_{\text{dr}(\mathcal{M})} = a.\text{post}(\rho(i))\} \end{aligned}$$

We define  $\llbracket \mathcal{M} \rrbracket = \bigcup_{i \in \mathbb{N}_0} \llbracket \mathcal{M} \rrbracket_i$  as the set of runs of  $\mathcal{M}$ . It follows that  $\llbracket \mathcal{M} \rrbracket \subseteq \llbracket \mathcal{V} \rrbracket$ .



### 2.3. Formal Preliminaries

**Example 2.2 (MTS of a Car Airbag)** We continue with Example 2.1: Each mode contains indeterminacy, for example, the choice between  $\tau$  and the other actions in mode standby, the choice between  $\text{expand}_f$  and  $\text{repair}_1$ . Moreover, weak  $\kappa$ -completion applies to the standby mode, particularly for the case  $\neg\text{energy} \wedge \text{crashed}$ , and to the mode failing. The defective functional action  $\text{expand}_f$  leaves a probabilistic  $\kappa$ -completion of 99%.  $\lrcorner$

Let  $\mathcal{M}_i = (\mu_i, \mathbb{A}_i, \Delta_i, \mu_{0_i}, \Sigma_{0_i}) \in \mathbb{M}$  for  $i \in \{1, 2\}$ . The operator  $\oplus : \mathbb{M} \times \mathbb{M} \rightarrow \mathbb{M}$  denotes the *superimposition* of two MTSs; the operator  $\otimes : \mathbb{M} \times \mathbb{M} \rightarrow \mathbb{M}$  denotes the *composition* of two MTSs into a concurrent MTS. The following definitions describe how  $\mathcal{M}_1 \oplus \mathcal{M}_2$  and  $\mathcal{M}_1 \otimes \mathcal{M}_2$  are defined and how  $\mathcal{M}$  can be abstracted from:

**Definition 2.8 (Superimposition)** Suppose that  $\exists n \in \mathbb{N} : \mu_1, \mu_2 \subset \mathbb{U}^n$ ,  $\mu_1 \cap \mu_2 \neq \emptyset$  and  $\mu_{0_1} \cup \mu_{0_2} \neq \emptyset$ .  $\mathcal{M}_1 \oplus \mathcal{M}_2 = (\mu, \mathbb{A}, \Delta, \mu_0, \Sigma_0)$  is defined by the equations  $\lrcorner$

$$\begin{aligned} \mu &= \mu_1 \cup \mu_2 & \mathbb{A} &= \mathbb{A}_1 \cup \mathbb{A}_2 \\ \Delta &= \Delta_1 \cup \Delta_2 & \mu_0 &= \mu_{0_1} \cup \mu_{0_2} \\ & & \Sigma_0 &= \Sigma_{0_1} \cup \Sigma_{0_2} \end{aligned}$$

Then, we call  $\mathcal{M}_1$  and  $\mathcal{M}_2$  fragments of the superimposition  $\mathcal{M}_1 \oplus \mathcal{M}_2$ .  $\lrcorner$

The map  $\text{merge} : \mathcal{E}_{\text{ctr}(\mathcal{M}_1)} \times \mathcal{E}_{\text{ctr}(\mathcal{M}_2)} \rightarrow \mathcal{E}_{\text{ctr}(\mathcal{M}_1 \otimes \mathcal{M}_2)}$  defines mergers of pairs of action effects (Definition 2.5). For each action effect,  $\mathbb{P} : \mathcal{E}_{\text{ctr}(\mathcal{M}_1 \otimes \mathcal{M}_2)} \rightarrow [0, 1]$  yields the probability to observe it. Given two actions  $a_1 \in \mathbb{A}_1$  and  $a_2 \in \mathbb{A}_2$ , we denote by  $a_1 \otimes a_2$  a *composite action*.

**Definition 2.9 (Parallel Composition)**  $\mathcal{M}_1 \otimes \mathcal{M}_2 = (\mu, \mathbb{A}, \Delta, \mu_0, \Sigma_0)$  is defined by the equations  $\lrcorner$

$$\begin{aligned} \mu &= \mu_1 \times \mu_2 \\ \mathbb{A} &= \{a_1 \otimes a_2 \mid \\ &\quad \forall i \in \{1, 2\} [a_i = (\text{pre}_i, \text{delay}_i, \text{trig}_i, \text{prio}_i, \pi_i, \text{post}_i) \wedge a_i \in \mathbb{A}_i] \wedge \\ &\quad a_1 \otimes a_2 = ( \text{pre}_1 \wedge \text{pre}_2, \max\{\text{delay}_1, \text{delay}_2\}, \text{trig}_1 \wedge \text{trig}_2, \\ &\quad \text{prio}_1 + \text{prio}_2, \pi_1 * \pi_2, \text{merge}(\text{post}_1, \text{post}_2) ) \wedge \\ &\quad (\text{pre}_1 \wedge \text{pre}_2 \not\perp) \wedge \\ &\quad \mathbb{P}(\text{merge}(\text{post}_1, \text{post}_2)) = \pi_1 * \pi_2 \} \\ \Delta &= \{((m_1, m_2), a_1 \otimes a_2, (m'_1, m'_2)) \mid \\ &\quad a_1 \otimes a_2 \in \mathbb{A} \wedge (m_1, a_1, m'_1) \in \Delta_1 \wedge (m_2, a_2, m'_2) \in \Delta_2 \} \\ \mu_0 &= \mu_{0_1} \times \mu_{0_2} \\ \Sigma_0 &= \{ \sigma \in \Sigma \mid \bigwedge_{i \in \{1, 2\}} \exists \sigma_i \in \Sigma_{0_i} |_{\text{mon}(\mathcal{M}_i)} : \sigma_i = \sigma |_{\text{mon}(\mathcal{M}_i)} \} \end{aligned}$$

<sup>15</sup>Incongruent pre and trig constraints can be avoided by splitting state constraints accordingly.

<sup>16</sup>An MTS can generate an infinite set of runs which contain the same monitored events but exhibit controlled events according to the probability distribution assigned to the action effects.

## 2. Technical Systems

Well-formedness for superimposition and parallel composition is discussed on page 25.

- ▮ **Definition 2.10 (Complex Action)** Given an MTS  $(\mu, \mathbb{A}, \Delta, \mu_0, \Sigma_0)$ , a complex action  $A$  is an expression defined by

$$A ::= a \mid A;A \mid A|A \mid (A) \mid A^+ \mid A^* \mid A^n \mid .$$

with each  $a \in \mathbb{A}$ , and the non-terminals  $A;A$  for concatenation,  $A|A$  for alternative choice,  $A^+$  for optional repetition,  $A^*$  for omission or optional repetition,  $A^n$  for repetition of  $n$  times, and  $.$  as the wildcard. For a set  $\{a_1, \dots, a_n\} \subseteq \mathbb{A}$ , we say that the expression  $a_1; \dots; a_n$  denotes an action trace if

$$\exists (m_0, \dots, m_n) \in \mu^{n+1} : m_0 \in \mu_0 \wedge \forall a_i \in \{a_1, \dots, a_n\} : (m_{i-1}, a_i, m_i) \in \Delta$$

and that the expression  $a_1 | \dots | a_n$  stands for a bundle of actions if

$$\exists m, m' \in \mu \forall a_i \in \{a_1, \dots, a_n\} : (m, a_i, m') \in \Delta$$

⊥

**Abstract Transition Systems** Let  $\alpha : \mathbb{M} \rightarrow \mathbb{M}$  assign an *abstraction* to each MTS.

- ▮ **Definition 2.11 (State Abstraction)** We define an abstract state  $\sigma_\alpha$  to be a state constraint (Definition 2.4). An abstract mode (event) is an abstract state only using variables in  $\mathcal{V}_m$  ( $\mathcal{V}_e$ ) in its state constraint. Let  $n \in \mathbb{N}_0$  and  $\Sigma_\alpha \subset \mathcal{C}$  be a set of abstract states which partition  $\Sigma$ ; we call  $\Sigma_\alpha$  an abstract state space. Then, an abstract run is a map  $\rho_\alpha : \{0, 1, \dots, n\} \rightarrow \Sigma_\alpha$ .

- ▮ **Definition 2.12 (Action Abstraction)** An abstract action  $a_\alpha$  is defined by

1. a composite action according to Definition 2.9, or
2. a complex action (Definition 2.10), or
3. a combination of 1., 2. and state abstraction.

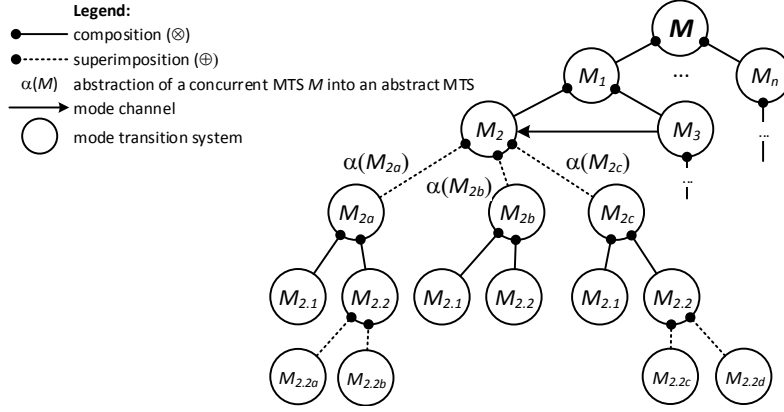
⊥

Based on  $\Sigma$ , we can define an abstract state space  $\Sigma_\alpha$  according to Definition 2.11. For  $\mathcal{M}$ ,  $\alpha$  yields an *abstract transition system*  $\mathcal{M}_\alpha = \alpha(\mathcal{M}) = (\mu_\alpha, \mathbb{A}_\alpha, \Delta_\alpha, \mu_{0_\alpha}, \Sigma_{0_\alpha})$  with  $\Sigma_{0_\alpha} \subseteq \Sigma_\alpha$  according to Definition 2.12. A map  $L : \Sigma \rightarrow \mathcal{P}(\mathcal{C})$  can show for any run  $\rho \in \llbracket \mathcal{M} \rrbracket$  which constraints and abstract state hold in some state. Whenever a transition in  $\Delta_\alpha$  is taken with an action  $a_\alpha \in \mathbb{A}_\alpha$ ,  $\mathcal{M}$  executes the actions of  $\mathbb{A}$  being part of  $a_\alpha$ . The results are abstract runs modelling *behavioural properties*.

- ▮ **Definition 2.13 (Mode Transition System, algebraic Representation)** Based on the Definitions 2.8 to 2.12, a mode transition system  $\mathcal{M}$  is an expression defined by

$$\begin{aligned} \mathcal{M} &::= \mathcal{M}_a \mid \mathcal{M}_c \mid \mathcal{M}_f & \mathcal{M}_c &::= \mathcal{M}_f \mid \mathcal{M}_c \otimes \mathcal{M}_c \\ \mathcal{M}_a &::= f \mid \alpha(\mathcal{M}_c) & \mathcal{M}_f &::= \mathcal{M}_a \mid \mathcal{M}_f \oplus \mathcal{M}_f \end{aligned}$$

- ⊥ with each  $f \in \mathbb{M}$ . This definition complements Definition 2.6.

Figure 2.2.: Exemplary visualisation of hierarchical decomposition of  $\mathcal{M}$ 

**Definition 2.14 (Aspect)** An aspect is a set  $\mathbb{M}_{\text{asp}} \subset \mathbb{M}$ .  $\mathcal{M}|_{\text{asp}}$  denotes the removal of all MTSs from  $\mathcal{M}$  which are not in  $\mathbb{M}_{\text{asp}}$ . ⌋

**Well-formedness and Construction of MTSs** Hierarchical decomposition of  $\mathcal{M}$  can be visualised, see Figure 2.2. Aside from the Definitions 2.8, 2.9 and 2.13, we require that the operator  $\oplus$  precedes  $\otimes$ , that  $\oplus$  preserves graph connectedness in  $\Delta$  and, thus, reachability of modes from  $\mu_0$ . Moreover, we only consider a composition  $\mathcal{M}_1 \oplus \mathcal{M}_2$  or  $\mathcal{M}_1 \otimes \mathcal{M}_2$  if this composition *preserves* well-formedness of modes as described on page 20.

We call  $\mathcal{M}$  *closed* iff  $\text{mon}(\mathcal{M}) = \text{ctr}(\mathcal{M})$ , *open* otherwise. We can close an open  $\mathcal{M}$  *finitely* by providing a set of runs  $\{\rho|_{\text{mon}(\mathcal{M}) \setminus \text{ctr}(\mathcal{M})}, \dots\}$  of sufficient length starting in  $\Sigma_0|_{\text{mon}(\mathcal{M}) \setminus \text{ctr}(\mathcal{M})}$ , or *infinitely* by composing  $\mathcal{M}$  with all required MTSs. For the remainder of this work, we suppose  $\mu_0 \neq \emptyset$  and  $\Sigma_0 \neq \emptyset$  for any closed and well-formed MTS. The discussion of how to maintain well-formedness is out of scope of this work.

**Corollary 2.2** From Definition 2.8 follows commutativity of  $\oplus$ .

**Properties and Use of Modes and Actions** Based on Definition 2.7, we call a mode  $m$

- *stable* iff  $\forall a \in \mathbb{A}_m : (a \in \mathbb{A}_f \wedge a = \epsilon) \vee (a \in \mathbb{A}_c \wedge a \neq \tau)$ , *unstable* otherwise
- *unknown* iff  $m$  is only reachable by  $\tau$  and  $\mathbb{A}_m = \emptyset$
- *final* iff  $\forall a \in \mathbb{A}_m \nexists m' : m \neq m' \wedge (m, a, m') \in \Delta$
- *dead* iff  $m$  is stable and final, that is,  $\epsilon$  models deadlock behaviour.

Modes can be used to model enabled, continued, paused or disabled phases of a system function (Section 2.2.1). The modelling of functional actions can be guided by

## 2. Technical Systems

modelling modes in advance. If indeterminacy in  $\mathbb{A}_f$  is unresolvable because of a lack of observability then prio (0 is highest) can be equal among all functional actions of a mode. A reduction of this lack can make the mode deterministic.  $\kappa$  allows any realisation to have arbitrary<sup>17</sup> effects in post, for example, none if  $\kappa = \epsilon$ . The control action  $\tau$  gives opportunity for the modelling of unknown (internal) activity. The well-formedness conditions on page 20 permit the use of incompleteness and indeterminacy for underspecification and, moreover, for refinement.

Each control action should be defined such that its pre constraint is disjoint from every functional action in the same mode, or its trig constraint totally covers the cutting set of its pre constraint and the pre constraints of the functional actions in this mode. Otherwise, it might happen that a possible control action is ignored although its priority is higher than that of any other functional action in that mode (cf. well-formedness condition 1). This way, control is assumed to take priority over function. In contrast to hybrid models (e.g. Henzinger 2000), MTSs neglect the concurrency of control and functional actions during mode transitions.

**Properties and Use of Composition in MTSs** *Concurrency.* Two MTSs composed in parallel (Definition 2.9) can not contain superimposable fragments (Definition 2.8) in their sub-hierarchies. Modes of a concurrent MTS are composed of the modes of its comprising MTSs. Based on strong  $\kappa$ -completion, a composite action models the *concurrent*<sup>18</sup> execution of all involved MTSs. *Totality of a mode* (see page 20) is needed for *realisable*<sup>19</sup> parallel composition.  $\epsilon$ -completion is one way to achieve *fully specified* modes.

*Effect Interference.* Such interference can arise from the execution of composite (i) functional or (ii) control actions. We only consider cases where two isolated functions  $\text{post}_1$  and  $\text{post}_2$  are merged into a *composite effect* by  $\text{merge}(\text{post}_1, \text{post}_2)$ , regarding, for example, domain properties of variables for (i), the parameter prio or exclusive resource assignments for (ii), and idempotencies of two interfering effects.  $\mathcal{V}_{\text{ef}} \cap \mathcal{V}_{\text{ec}} = \emptyset$  rules out interference of functional and control effects. For two MTSs  $\mathcal{M}_1$  and  $\mathcal{M}_2$ ,  $\text{ctr}(\mathcal{M}_1) \cap \text{ctr}(\mathcal{M}_2) = \emptyset$  can be a solution<sup>19</sup> to avoid effect interference.

*Dependency and Interaction.* A *mode dependency* between two parallel MTSs can be modelled by referring to a mode channel in the pre or trig constraints of an action specification. Mode channels are only affected by control actions of an MTS and can be monitored by other MTSs:  $\mathcal{M}_1$  and  $\mathcal{M}_2$  are *mode dependent* iff  $\{\text{m}_{\mathcal{M}_1}, \text{m}_{\mathcal{M}_2}\} \cap (\text{mon}(\mathcal{M}_1) \cup \text{mon}(\mathcal{M}_2)) \neq \emptyset$  and independent otherwise. Suppose that  $n$  composed MTSs  $\mathcal{M}_1, \dots, \mathcal{M}_n$  are applied to state  $\sigma$ : all possible effects of  $\mathcal{M}_j$  to the successor state  $\sigma'$  are determined by the modes of  $\mathcal{M}_l$  ( $l \neq j$ ), the actions of  $\mathcal{M}_j$  enabled in mode  $\sigma(\text{m}_{\mathcal{M}_j})$  and effect interference.

*Superimposition* can be used for pending modifications, for example, union, update, addition and removal of actions, modes and indeterminacy. *Priorities* help resolve indeterminacy arising from superimposition and effect interference of concurrent MTSs.

<sup>17</sup>An arbitrary choice can be hazardous. Anyway, this situation is different from the *implicit denial of side effects*, that is, variables not referenced by post shall stay unchanged.

<sup>18</sup>A run makes action effects of all comprising MTSs observable for each time interval  $[t, t + 1)$ .

<sup>19</sup>Notions of *realisability* and *interference freedom* are discussed in Broy and Stølen (2001).

Prioritisation can be applied to aspects (Definition 2.14). Part of the overlaps which model indeterminacy can be resolved by trig constraints or priorities.

### 2.3.2. Behavioural Property Specification

The following chapters adopt full probabilistic computation tree logic (PCTL\*); e.g. Baier and Katoen 2008) for asserting temporal and behavioural properties.

**Definition 2.15 (Behavioural Property)** *Let  $\mathcal{V}$  be a set of variables,  $\mathcal{V}_e \subset \mathcal{V}$  be a set of channels and  $\mathcal{M}$  be an MTS (Definition 2.13). We can assert a behavioural property  $\phi$  (of  $\mathcal{M}$ ) using a PCTL\* state formula defined by* ⌈

$$\phi ::= \top \mid c \mid \phi \wedge \phi \mid \neg\phi \mid P_{\sim p}[\psi] \mid \mathbf{E}\psi \mid \mathbf{A}\psi$$

containing path formulae defined by

$$\psi ::= \phi \mid \psi \wedge \psi \mid \neg\psi \mid \mathbf{X}\psi \mid \psi \mathbf{U}^{[\sim k]}\psi \mid \mathbf{F}^{[\leq k]}\psi \mid \mathbf{G}^{[\leq k]}\psi \mid \chi$$

and past formulae defined by

$$\chi ::= \top \mid c \mid \chi \wedge \chi \mid \neg\chi \mid \bar{\mathbf{X}}\chi \mid \chi \bar{\mathbf{U}}^{[\sim k]}\chi \mid \bar{\mathbf{F}}^{[\leq k]}\chi \mid \bar{\mathbf{G}}^{[\leq k]}\chi \mid a \mid [\phi \text{ P } p]$$

and action effect formulae defined by

$$a ::= \exists x_1, \dots, x_n : \bar{\mathbf{X}}c \wedge c'$$

with  $c, c' \in \mathcal{C}$  (Definition 2.4),  $p \in [0, 1]$ ,  $\sim \in \{<, >, \leq, \geq\}$ ,  $\{x_1, \dots, x_n\} \cap \mathcal{V} = \emptyset$  and  $k, n \in \mathbb{N}$ .  $P_{\sim p}[\psi]$  yields the probability of  $\psi$  (in  $\mathcal{M}$ ).  $\phi$  is called interface property if each state constraint in  $\phi$  only uses variables in  $\mathcal{V}_e$ . ⌋

For a brief introduction, the operators of PCTL\* are explained in the following. Let  $\text{CT} \subseteq \llbracket \mathcal{M} \rrbracket$  be a computation tree according to Definition 2.2,  $\rho$  be a run of CT,  $\sigma$  be a state of  $\rho$  and  $\sigma_0 = \rho(0)$ :

- The atomic statements  $\top$ ,  $c$  and  $c'$ , the connectives  $\wedge$  and  $\neg$ , and the quantifier  $\exists$  inherit their meanings from first-order logic.
- By annotating CT with the  $\pi$  parameters of the action set  $\mathbb{A}$  of  $\mathcal{M}$ , CT can be associated with a probability space. This space contains for any path formula  $\psi$  the probability of its satisfying subset of CT: The expression  $\text{CT} \models P_{\sim p}[\psi]$  denotes that the probability, that  $\psi$  is true for a run of CT, satisfies  $\sim p$ . The formula  $\mathbf{E}\psi$  denotes that  $\psi$  is true for at least one run in CT;  $\mathbf{A}\psi$  denotes the truth of  $\psi$  for all such runs. We have the equivalences  $\mathbf{E}\psi \equiv P_{>0}[\psi]$  and, for finite computation trees,  $\mathbf{A}\psi \equiv P_{\geq 1}[\psi]$ .
- The formula  $\mathbf{X}\psi$  denotes that  $\psi$  is true for the next state of a run  $\rho$ , that is,  $\rho(1)$ . The formula  $\psi_1 \mathbf{U} \psi_2$  asserts that  $\psi_1$  is true from the start of a run  $\rho$  until  $\psi_2$  gets true in  $\rho$ . The formula  $\mathbf{F}\psi$  describes that  $\psi$  is true in some state of  $\rho$ ;  $\mathbf{G}\psi$  describes truth in all of  $\rho$ 's states.

## 2. Technical Systems

- The formula  $\psi_1 \mathbf{U}^{\sim k} \psi_2$  restricts  $\psi_1 \mathbf{U} \psi_2$  to get satisfied in  $\rho$  within ( $<, \leq$ ) or only after ( $>, \geq$ )  $k$  time steps. Analogously, the bounded operators  $\mathbf{F}^{\leq k}$  and  $\mathbf{G}^{\leq k}$  restrict  $\mathbf{F}$  and  $\mathbf{G}$  to get satisfied in  $\rho$  within  $k$  time steps. For any association of time units, such as ms, s, min or h, with  $k$ , we assume the conversion of  $k$  into the time unit into which runs of  $\mathcal{M}$  are interpreted. The same holds for the parameter delay in an action specification.
- Suppose that the map  $\leftarrow : \llbracket \mathcal{V} \rrbracket \rightarrow \llbracket \mathcal{V} \rrbracket$  inverts any finite run  $\rho$  such that  $\rho$ 's last state gets  $\sigma_0$ : the past fragment, consisting of the operators  $\overline{\mathbf{X}}, \overline{\mathbf{U}}, \overline{\mathbf{F}}$  and  $\overline{\mathbf{G}}$ , then corresponds to the future operators  $\mathbf{X}, \mathbf{U}, \mathbf{F}$  and  $\mathbf{G}$  evaluated for  $\bar{\rho}$ .
- The past formula  $[\phi \mathbf{P} p]$  extends PCTL\* by taking the frequency view of probability and ignoring the  $\pi$  parameter (Johnson 1993).  $[\phi \mathbf{P} p]$  is true for a state  $\sigma = \rho(n)$ , with  $n, p \in \mathbb{N}$ , if the number of  $\sigma$ 's previous states in which  $\phi$  was true divided by all its previous states in  $\rho$  equals  $p$ .  $p$  can also be an interval.

The operators  $\mathbf{A}$  and  $\mathbf{E}$ , the bounded operators  $\mathbf{U}^{\sim k}, \mathbf{F}^{\leq k}$  and  $\mathbf{G}^{\leq k}$ , past formulae and action effect formulae serve convenience and are inexpressive for finite computation trees: we assume that appropriate expansion or rewriting rules can be applied.

The *satisfaction relation* on states ( $\sigma \models \phi$ ), runs ( $\rho \models \phi$ ), and computation trees or mode transition systems ( $\mathcal{M} \models \phi$ ) is inductively defined over the structure of  $\phi$ .<sup>20</sup>  $\mathcal{M} \models \phi$  can be translated into an element of the subset relation  $\subseteq$  on sets of runs. The notation  $\mathcal{M} \models \phi$  (spoken “ $\mathcal{M}$  satisfies  $\phi$ ”) states that each run of  $\mathcal{M}$  is allowed by  $\phi$ ;  $\mathcal{M} \not\models \phi$  (spoken “ $\mathcal{M}$  violates  $\phi$ ”) means that some runs are prohibited. For a behavioural property  $\phi$ ,  $\llbracket \phi \rrbracket$  stands for the set of its logically valid evaluations, with  $\llbracket \phi \rrbracket \subseteq \llbracket \mathcal{V} \rrbracket$ . We call a transition system  $\mathcal{M}$  *valid* iff  $\mathcal{M} \models \phi$  or  $\llbracket \mathcal{M} \rrbracket \subseteq \llbracket \phi \rrbracket$ . We say that a path formula  $\psi$  is *finitely satisfiable* by a transition system  $\mathcal{M}$  iff there exists a *finite run*  $\rho \in \llbracket \mathcal{M} \rrbracket$  with  $\rho \models \psi$ . For a set of formulae  $\Phi$ ,  $\hat{\Phi}$  denotes  $\bigwedge_{\phi \in \Phi} \phi$  and  $\check{\Phi}$  denotes  $\bigvee_{\phi \in \Phi} \phi$ . Consequently, we have that  $\llbracket \hat{\Phi} \rrbracket = \bigcap_{\phi \in \Phi} \llbracket \phi \rrbracket$ .

### 2.4. System Specification: A Generic Framework

Consider a *specification*  $\mathcal{S} = (\mathcal{V}, \mathcal{M}, \Gamma)$  and a *realisation*  $\mathcal{W}$  (Figure 2.1) with

- a set  $\mathcal{V}$  of variables through which to observe the physical dynamics of  $\mathcal{W}$
- a *mode transition system*  $\mathcal{M}$  (Definition 2.13) which models  $\mathcal{W}$
- a set  $\Gamma$  of assertions of *behavioural properties* (Definition 2.15) of  $\mathcal{M}$ .

According to Section 2.2.2,  $\mathcal{M}$  and  $\Gamma$  specify possible realisations by *operationally* and *descriptively* constraining  $\llbracket \mathcal{V} \rrbracket$ . Given  $\llbracket \mathcal{M} \rrbracket \subseteq \llbracket \mathcal{V} \rrbracket$  and  $\llbracket \hat{\Gamma} \rrbracket \subseteq \llbracket \mathcal{V} \rrbracket$ , the behaviour allowed by  $\mathcal{S}$  is defined by  $\llbracket \mathcal{S} \rrbracket = \llbracket \hat{\Gamma} \rrbracket \cap \llbracket \mathcal{M} \rrbracket$ , and  $\mathcal{W}$  conforms to  $\mathcal{S}$  iff  $\llbracket \mathcal{W} \rrbracket \subseteq \llbracket \mathcal{S} \rrbracket$ .<sup>21</sup>

system view

Adapted from Section 2.2.3,  $\mathcal{S}$  and  $\mathcal{W}$  unfold two pairs of *system views*:

<sup>20</sup>See, e.g. Baier and Katoen 2008, Manna and Pnueli 1995, Pnueli and Kesten 2002.

<sup>21</sup>*Consistency of  $\mathcal{S}$*  can be expressed by  $\llbracket \mathcal{M} \rrbracket \subseteq \llbracket \hat{\Gamma} \rrbracket$  and  $\llbracket \hat{\Gamma} \rrbracket \neq \emptyset$ . Being less restrictive,  $\llbracket \mathcal{M} \rrbracket \cap \llbracket \hat{\Gamma} \rrbracket \neq \emptyset$  makes it sufficient to require  $\llbracket \mathcal{W} \rrbracket \subseteq \llbracket \mathcal{M} \rrbracket \cap \llbracket \hat{\Gamma} \rrbracket$ .

**Actually/ideally specified and modelled** The *actual/ideal* description and representation of a specification will be denoted by  $\mathcal{S}/\mathcal{S}'$ .<sup>22</sup>

**Actually/ideally realised and operated** The *actual/ideal* interpretation of  $\mathcal{S}$  or  $\mathcal{S}'$  and the corresponding operation of a realisation will be denoted by  $\mathcal{W}/\mathcal{W}'$ .

These views enable the use of three sets of variables  $\mathcal{V}_{\text{specified}}$ ,  $\mathcal{V}_{\text{realised}}$  and  $\mathcal{V}_{\text{ideal}} \triangleq \mathcal{V}'$  which yield three spectra (Definition 2.3). Behaviour to be specified by  $\mathcal{S}$  belongs to  $\llbracket \mathcal{V}_{\text{specified}} \rrbracket$ ; behaviour to be actually realised in  $\mathcal{W}$  is a member of  $\llbracket \mathcal{V}_{\text{realised}} \rrbracket$ . Behaviour to be ideally observable in  $\mathcal{S}'$  or  $\mathcal{W}'$  belongs to  $\llbracket \mathcal{V}_{\text{ideal}} \rrbracket$  which uses a complete set of correctly typed variables. We do not presume the case  $\mathcal{V}_{\text{specified}} = \mathcal{V}_{\text{realised}} = \mathcal{V}_{\text{ideal}}$ .

## 2.5. Notes and Further Reading

Like *fluents* (Reiter 2001), *variables* can be used to model, for example, physical or data states, control communication, physical interaction, mode dependencies, timing or fault indication. The term *run* coheres with the notions of *behaviour* (Lampert 2002) and *stream* (Broy and Stølen 2001). We assume that the sets or languages of runs  $\llbracket \mathcal{V} \rrbracket$ ,  $\llbracket \mathcal{M} \rrbracket$  and  $\llbracket \hat{\Gamma} \rrbracket$  are prefix-closed (see, e.g. Sampath et al. 1996).

Courtois and Parnas (1993) and Parnas and Madey (1995) characterised *modes* and mode transitions by predicates over states. By way of the pre and trig elements of control actions, an MTS can encode *may-* and *must-transitions* for deontic reasoning (see, e.g. van Lamswerde 2009). The notion of *stable mode* is motivated by Pugliese and Tronci (1996); the extension of PCTL\* to action effect formulae (Definition 2.15) stems from the works of Lampert (2002) and Reiter (2001).

Parallel composition allows the construction of *composite actions*, superimposition enables the construction of *complex actions* (Lampert 2002, Reiter 2001) and action abstraction (Definition 2.12) leverages the construction of *abstract actions*. *Action priorities* help determine the superimposition of two MTSs. Hence, these priorities are different from process or task priorities as, for example, used by Hoare (1985) or Magee and Kramer (2006). Apel and Lengauer (2008) show a study on superimposition for software composition. Both composition operators attach meaning to transition diagrams such as discussed by Harel and Politi (1998). State and action abstraction have been investigated from various perspectives<sup>23</sup>; such abstractions help simplify the representation of a system as a KRIPKE structure (see, e.g. Baier and Katoen 2008). As defined by Broy (2005), composition (Definition 2.13) without action abstraction induces an *interface subtype* relation on  $\mathbb{M}$  whose elements have to maintain *faithful projection*. Broy (2010) describes the concept of *interface abstraction* of transition systems: the obtained *interface behaviour* helps analyse interface properties (Definition 2.15) and compose system functions.

<sup>22</sup> $\mathcal{S}'$  denotes that the degree of *quality* (e.g. *safety*) has been increased for  $\mathcal{S}$ .

<sup>23</sup>See, e.g. Börger and Stärk 2003, Broy 2010, Clarke et al. 2000, Fantechi et al. 1999, Pasareanu et al. 2007.

## 2. *Technical Systems*

The concurrency model built on Corollary 2.1 allows investigating properties, such as freedom of dead modes and fairness among actions, in a detailed way. Concurrent mode transition systems perform synchronously in time and interaction as opposed to asynchronous and synchronised composition as used in, for example, mode automata (see, e.g. [Rauzy 2002](#)). These decisions reduce technological assumptions on interaction and concurrency mechanisms as discussed by [Hoare \(1985\)](#) and [Broy and Stølen \(2001\)](#). On the contrary, such assumptions (e.g. interleaving semantics, synchronisation on shared events or messages) allow specific action abstractions representing such mechanisms to obtain simplified transition systems with smaller sets of actions.



In this chapter, I provide an overview of safety viewpoints, standards, analysis and design techniques, and measures, together with a discussion of recent related work.

## Contents

3.1	Safety Viewpoints and Standards	31
3.2	Hazard Analysis Techniques	34
3.3	Safety Measures	37
3.4	More Recent Related Work	38
3.4.1	System Modelling for Safety	38
3.4.2	Causative Reasoning for Safety	40
3.4.3	Safety Engineering Guidance	44

## 3.1. Safety Viewpoints and Standards

In the history of the construction and operation of technical systems, safety engineering originated from older engineering disciplines, such as mechanical (Ericson 2005, Luksch 2012), chemical (Sinell and Meyer 1996) and electrical (Börcsök 2011) engineering, and found its way into computer technology and software engineering (Leveson 1986, 1995). The present work aims to stimulate readers from all these disciplines. In the life cycle (Section 2.1), *safety engineering* encompasses the understanding of a system, its potential defects and *hazards*, and the specification, design, realisation and verification of *safety measures* (Ericson 2005, Leveson 2012).

safety engineering

**System-centric, technological Viewpoint** Parnas et al. (1990) discussed *software safety* in relationship with trustworthiness, availability and reliability of computing devices as a whole. Software can be seen as the initialisation of such devices with the entire code running on them being safety-critical. Hence, safety is not an isolated property

### 3. Safety

or role of a separate code part, but a more general implication of reliability and availability (i.e. the probability of absence of operational defects), and trustworthiness (i.e. the probability of absence of specification defects). Parnas et al. raised demand for educational programs for software engineers in safety-critical application domains.

McDermid (1991) alluded to the impossibility of absolute system safety and assessed techniques of how to increase trustworthiness in spite of this limitation (see also Barroca and McDermid 1992). He suggested that software should be treated the same as any other component of a technical system. In accordance with further authors<sup>1</sup>, he stressed the use of formal methods for control system specification and safety analysis in combination with requirements documentation and validation<sup>2</sup>, and independent testing (McDermid 1986). It is generally accepted that formalisation helps automating tedious reasoning tasks. Based on several examples and using his own approach, McDermid argued when and to what extent formal methods can be practiced in a safety context.<sup>3</sup> Parnas et al. and McDermid and Pumfrey (2001) also gave reasons why software analysts could concentrate on systematic defects. Nevertheless, for distributed software-intensive systems, one can consider random behaviour not (solely) to be explained by physical deterioration of the underlying technology (see, e.g. Leveson and Stolzy 1987, Schulz and Peleska 2010).

Under the assumption that no single analysis viewpoint and technique suit comprehensive safety analysis, Pyle (1991) and Wilson and McDermid (1995) proposed to apply several viewpoints and techniques, and to keep consistent their results. Based on a clear definition of the boundary of a control system, Pyle (1991) provided examples of how to restrict programming languages to reduce fault possibilities and software safety risks. Whereas Pyle considered the separation of safety-critical program parts and built-in program test equipment recommendable, Parnas et al. (1990) discouraged from such practice to simplify programs and reduce causal factors. Finally, McDermid (2001, 2002) criticises standards to be process-centric and motivates safety engineers to rely more heavily on product-based safety evidence.

**Human-centric, holistic Viewpoint** Technical systems carry *behavioural properties* with *severe impact* on their environments. Leveson worked on an extension of safety analysis to include causal factors in software programs (Leveson and Harvey 1983a,b). Being influenced by *accident research* and in accordance with other authors (see, e.g. Neumann 1995, Pyle 1991), she stressed software safety to be viewed as a *property* not only of the software running on its computing device but also of, for example, human operators, the environment, the control system, the entire technical system, the organisation operating this system, and regulative laws (Leveson 1995). In fact, hazards and accidents are often caused by multiple, complex and interwoven factors.<sup>4</sup> Summing up, Leveson (2012) points to the identification of such factors beyond linear causal chains or singular system faults.

<sup>1</sup>See, e.g. Bloomfield et al. 1991, Parnas et al. 1990.

<sup>2</sup>See, e.g. Atlee and McDermid 1995, Courtois and Parnas 1993, Fenelon et al. 1994.

<sup>3</sup>See, e.g. Barroca and McDermid 1992, McDermid 1986, McDermid and Pumfrey 1994.

<sup>4</sup>This has been studied by, e.g. Perrow (1984) and Rasmussen (1997).

### 3.1. Safety Viewpoints and Standards

**Formal, logic Viewpoint** According to [Lamport](#), safety as a behavioural property expresses that “nothing *bad* happens”: any violation can be observed over finite runs (see, e.g. [Lamport 2002](#)). This more general idea was formalised, for example, by [Manna and Pnueli \(1995\)](#) who have been working on a taxonomy of behavioural properties. [Pyle \(1991\)](#) criticised that, in verification of safety-critical software, [Lamport](#)’s ‘bad’ is often only associated with unacceptable states in the context of mutual exclusion, critical section synchronisation, absence of deadlock, and partial correctness. Having in mind social system environments and life cycles, [Pyle](#) used the terms *logical* or *software safety* for [Lamport](#)’s motivation, and *physical* or *system safety* to accommodate to the expectations of [Leveson](#), [McDermid](#) and [Parnas et al.](#)

The first two viewpoints more constructively investigate *hazardous causalities* among physical events and expect safety to pertain the entire system life cycle. The third viewpoint is more concerned with possibly unwanted system behaviour underlying any discussion of hazardous causalities, independent of such a life cycle.

**Terminology in brief** [Watson and Leadbetter \(1964\)](#) modelled hazards as conditional failure rates of systems. Being less abstract, according to [Leveson \(1995\)](#),

“a *hazard* is a state or set of conditions of a system (or an object) that, together with other conditions in the environment of the system (or object), will lead inevitably to an accident (loss event).”

[Ericson \(2005\)](#) has a constructive notion of hazard, combining a hazardous element with an initiating mechanism to threaten an asset. The *risk* consists in a potentially negative outcome from this mechanism’s specified or defective performance, that is, a *mishap* for the asset such as human injury, environmental or system damage, or loss. According to [van Lamsweerde \(2009\)](#), a “safety hazard is a risk for the safety of an object in the environment” and a “human health hazard is a risk for the health of a human using the system or affected by the system in his behaviour.” We use the term

mishap

*hazard* (also hazardous state, threat to safety, safety risk) to denote a *minimum condition* on behaviour which entails a *mishap* at certain risk.

hazard

Then, the term *causal factor* will be used to denote any conjunct or whole of such a minimum condition. An occurrence of a hazard can be seen as an *incident*.

causal factor

**Safety-related Standards** Technical risk management<sup>5</sup> comprises *standards* covering life cycles in many domains: ISO 12100 and DIN EN 414 provide safety considerations for machinery. IEC Std. 61508 (2011) applies to non-mechanical safety-related parts of mechatronic systems. ISO 14121 provides risk assessment guidelines for standards such as DIN 19250/1 or DoD MIL-STD-882D. IEC 61508 is applied in both IEC 62061 for industrial machinery and ISO Std. 13849 (2006), to address the EU guideline 2006/42/EG for machinery ([Gehlen 2010](#)). IEC 61508 has been adapted

<sup>5</sup>See, e.g. [Boehm 1991](#), [Layman et al. 2011](#), [Lund et al. 2011](#), [ISO Std. 31000 \(2009\)](#).

### 3. Safety

in ISO Std. 26262 (2011) to support the deployment of control systems in series-production vehicles. IEC 61511 applies to process industry, IEC 61513 to atomic power plants, IEC 62304 and ISO 14971 to medical devices, EN 50128 to railway control, and SAE aircraft recommended practices (ARP) 4754 and 4761 to aerospace industry. The standards DO-178B, DO-248B and DO-254 (RTCA 2001) constrain the development of airborne control software and electronic hardware; the MISRA software guidelines play a similar role in the automotive industry. In summary, many standards are intentionally generic<sup>6</sup>, provide a consensual frame and refer to applicable methods and techniques.

## 3.2. Hazard Analysis Techniques

Hazard analysis is a prerequisite for the derivation of *safety requirements for functions* or parts of a system. Such analysis is required by most safety standards: IEC 65108 speaks of PHA, ISO Std. 26262 (2011) of *hazard analysis and risk assessment (HARA)*. ISO 26262 adheres to hierarchical structural modelling of the system and its hardware/software design for conducting safety management plans. The SAE aircraft recommended practices (ARP) 4754 and 4761 advise the step of functional hazard assessment (FHA) based on a function list regarding system failures and crew actions. This step is followed by preliminary and final system safety/reliability assessments (SSA) which require a mixture of techniques. In summary, safety standards cover the system life cycle including hazard assessment for the regarded applications and technologies. The use of these standards shall reduce unwanted relationships between safety goals of the system, subsystem requirements and component properties.

The mishaps and hazards to identify and assess can stem from defective physical processes or architecture designs (task T1) and defective specifications (task T2). Safety engineers apply their domain knowledge in various ways: We can distinguish *deductive* (top-down, backward), *inductive* (bottom-up, forward), *bidirectional* and *non-linear* techniques (see, e.g. Börcsök 2011, Ericson 2005). Known from Section 2.2.1, these techniques can use *structural*, *behavioural*, *mixed* or *implicit* models.

**Process and Design Analysis (T1)** Many approaches aim at defect detection of a previously known design (e.g. Ericson 2005, McDermid and Pumfrey 1994), requiring a structural model and a corresponding way to model defects (Section 2.2.3).

Fault tree analysis (FTA; Dugan et al. 1992, DIN 25424, IEC 61025) is performed *deductively* such that it considers known or conceivable failures and tries to localise causal factors such as component faults. This approach aims at *minimal cut sets or sequences*<sup>7</sup> by assessing design models prior to or after system operation (Börcsök 2011, Liggesmeyer 2009). Failure probabilities can be determined by quantitative FTA

<sup>6</sup>ISO Std. 26262 (2011) speaks of *safety validation* (cf. Table A.10 column 3) when referring to the V&V of the planned safety concepts. This standard, however, exhibits terminological redundancy and inconsistencies. For example, the terms *transient fault* and *single point fault* seem to be used in part 5 different from their definition in part 1.

<sup>7</sup>Minimal combinations of faults or shortest sequences of erroneous states leading to a system failure.

if design decisions took place. FTA originated from mechanical engineering and can be applied to software designs and implementations (Leveson and Harvey 1983a).

As opposed to FTA, failure mode and effects analysis (FMEA<sup>8</sup>) is conducted *inductively* for it considers the *causal chain of erroneous states* starting from component faults up to their unwanted effects (Ericson 2005, VDI 2222, VDA 2006, DIN 25448). FMEA can be applied to a design or an operated system, capturing the entire causal chain to handle common cause failures. This technique has drawbacks in considering multiple causes at once. Originating from mechanical engineering, like FTA, FMEA became available for software-intensive control systems (see, e.g. Goddard 2000, McDermid 2002). It can be a qualitative worst-case impact analysis, such as hazard identification (HAZID) and preliminary hazard lists (Ericson 2005). In contrast, probabilistic risk assessment (PRA) and MARKOV chain techniques can produce quantified results (see, e.g. Börçsök 2011, Kumamoto 2007). Event tree analysis (ETA; DIN 25419) and layer of protection analysis (LOPA) forwardly follow the causal chain towards mishaps, like FMEA, and are particularly applied to large systems. Hazard and operability studies (HAZOP; IEC 61882) use *guide words* for hazard identification and take account of controllability by humans (Börçsök 2011). This idea has been adopted in accident analysis by Leveson (2012) and, for chemical plants, by Stursberg et al. (1998). “Hazard analysis and critical control points” (HACCP; Sinell and Meyer 1996) is a variant of FMEA for processes in the food industry.

guide words

To gain an *a-posteriori* understanding of mishaps (Hopkins 2000), methods such as accident cause analysis (Perchonok 1972), AcciMaps (Svedung and Rasmussen 2002), events and causal factors (ECF; Buys and Clark 1995), human engineering (Nader 1965), human error risk management in engineering systems (HERMES; Cacciabue 2004), and root cause analysis (RCA; Cacciabue 2004) regard causal factors in the interaction of environment, operator and system. Such analyses take use cases (e.g. driving missions and situations), mishap scenarios (e.g. car accidents) and physical system interfaces to investigate system operation (Luksch 2012). Causal chains between mishaps and hazards are traced backward and forward. For example, “crisis intervention in offshore production” (CRIOP; Johnsen et al. 2011) assesses the interface between operators and systems in offshore control rooms to uncover obstacles for accident response; the international classification for patient safety (ICPS; WHO 2012) helps assess clinical incidents and establish safety measures in health care processes.

In summary, for any distinctive event  $e$  at the system boundary (Figure 3.1),

- *deductive techniques* such as FTA aid *possibility* questions: What are potential causes leading to  $e$ ? Which are the minimal cut sequences for  $e$ ?
- *inductive techniques* such as FMEA aid *impact and severity* questions: Is  $e$  among the consequences of a specific fault? What are the potential mishaps resulting from  $e$ ?

**Requirements Analysis (T2)** Requirements engineering (Section 2.1) aims at valid specifications with *safety goals* based on hazard reports: Hazards can motivate safety

<sup>8</sup>Including extensions for severity or detectability analyses such as FMECA or FMEDA.

### 3. Safety

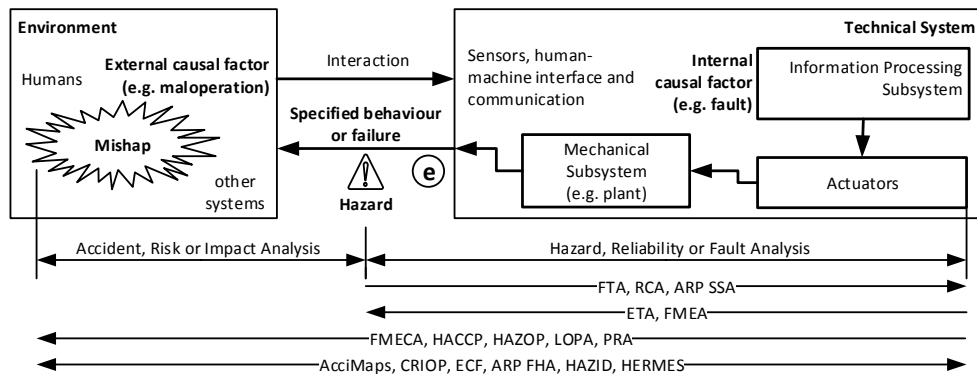


Figure 3.1.: Approaches to hazard analysis of technical systems, mapped according to their focused area and direction of causative reasoning; e . . . distinctive event

goals. With known goals, additional hazards can be elicited by goal negation, tautological refinement and identifying necessary conditions for goals to be obstructed (van Lamsweerde 2009). Fenelon et al. (1994), Kelly (1998) and van Lamsweerde discuss how to specify safety goals and how to build up argumentation for goal achievement.

**Safety versus Reliability** Reliability engineering<sup>9</sup> aims to reduce the fraction of failures by increasing the fraction of treated (i.e. detected, tolerated, avoided) causal defects (T1). Safety engineering aims to reduce the fraction of hazards by increasing the fraction of treated causal factors (T1'), and to reduce the number of unknown or disregarded hazards (T2). Leveson (2012) argues for a careful separation of safety and reliability viewpoints. Again, achieving safety encompasses two tasks:

- The reduction of hazardous system behaviour (i.a. system defects, T1).
- The anticipation of hazardous environment behaviour (i.a. maloperation, T2).

**Hazard Characteristics** Coherent with the quantification of risks (see, e.g. Boehm 1991, Kumamoto 2007), standards such as DIN 19250 and ISO 26262 require the *characterisation* of hazards by the *severity* of impact on the environment, the *probabilities* of *occurrence* and of detection of causes (*detectability*), and the environments' own ability to avoid mishaps (*controllability*). IEC 61508 requires estimates of *exposure* to causal factors of hazards and mishaps (Börcsök 2011).

These characteristics lead to safety requirements, life cycle guidance and design patterns bundled as *integrity classes* (IC) to be assigned to system functions and subsystems: “Anforderungsklassen” (AK; DIN 19250), design assurance levels (DAL; DO-178B), categories (Cat; EN 954, F/JAR 25.1309), safety integrity levels (SIL; IEC 61508), performance levels (PL; ISO 13849), automotive SILs (ASIL; ISO 26262),

<sup>9</sup>See, e.g. Börcsök 2011, Gaede 1977, Kumamoto 2007, Liggesmeyer 2009.

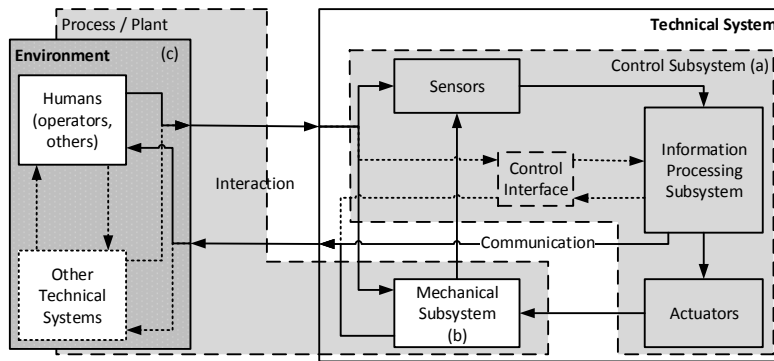


Figure 3.2.: *Safety measures* of a technical system allocated to subsystems and environment: functional (a), direct/indirect (b), organisational (c); adapted from VDI (2004)

“Besondere Merkmale bzgl. Sicherheitsanforderungen” (BM S; VDA 05/2011), etc. Finally, a system’s safety measures and life cycle need fulfil the imposed ICs.

### 3.3. Safety Measures

Safety measures should keep the risk of leaving a *safe state* towards a mishap, for example, *below a quantified level* or *as low as reasonably practicable* (ALARP).<sup>10</sup> According to Martinus (2004) and Luksch (2012), *safety* can be improved by *constructive* (Figure 3.2a,b) and *organisational* (also indicative and personal; Figure 3.2c) measures. The former can be split into

- a. functional measures (e.g. compensate hazardous functionality) which make up the safety-related part of the control subsystem (Figure 3.2a)
- b. direct (e.g. avoid hazardous materials) and indirect (e.g. permanently cover hazardous machine areas) measures which make up the safety-related part of the mechanical subsystem (e.g. part of a plant; Figure 3.2b).

**Functional Safety** To provide what we call *safety functionality*, functional measures usually comprise *electric, electronic and programmable electronic* technologies, sensors and actuators. *Functional safety* is the part of safety of a system realised by functional measures (IEC 2011). *Safety integrity* characterises the reliable operation of functional measures for safe system operation with respect to T1 and, thus, the degree of functional safety (Börcsök 2011). For safety integrity, IEC 61508 and ISO 26262 recommend hardware metrics: the *diagnostic coverage metric* can reflect the fraction of a defect treated by a functional measure whereas *single point* and *latent fault metrics* can capture the robustness of a functional measure.

Safety integrity

<sup>10</sup>McDermid (2001) weighed up the legal idea of ALARP for control software without committing himself to a position.

### 3. Safety

**Design Patterns** Leveson and Stolzy (1987) distinguished *fault tolerance*, *fail-soft* and *fail-safe* properties of a technical system. By unifying the first two classes of properties, Martinus (2004) explains two *fail-safe patterns* for T1 through reliable transition from a *hazardous defective state* to a *safe state*:

**Fail-Operational (FO)** After having detected a defective state, a safe state is reached through redundancy by *recovering* from the defective function to one of its operational but often *degraded* alternatives.

**Fail-Silent (FS)** After having detected a defective state, a safe state is reached by *disabling* the defective function. Such a mechanism can usually be realised without redundancy or fault tolerance.

Both patterns aim to avoid or interrupt the escalation of defects to hazards with *severe impact* (i.e. a mishap). For the FO pattern, fault tolerance improves reliability or availability, and treats hazardous failures; the hand-over is controlled by a safety-related subsystem. For both patterns, safety-related subsystems require diagnosis equipment and can themselves be realised redundantly. See Example 3.1.

▮ **Example 3.1 (Fail-Safe Patterns)** FO: *steer-by-wire exchanged by mechanical fallback steering if a software fault is detected; primary flight-control exchanged by a secondary system if a random fault is observed.* FS: *indicated switch-off of an airbag, anti-blocking system or electronic stability control if a sensor fault is diagnosed.*

▮ **Definition 3.1 (Safety-oriented Requirements Validation)** *By the safety-oriented validation of a specification  $S$ , as defined in Section 2.4, we mean the hazard analysis of  $S$  and the enhancement of  $S$  with safety measures for the tasks T1 and T2.*

This definition suggests specifications of a technical system which combine necessarily hazardous functionality and corresponding safety functionality.

## 3.4. More Recent Related Work

This section lists more recent research related to this thesis with respect to system modelling, causative reasoning and engineering guidance in safety-oriented validation.

### 3.4.1. System Modelling for Safety

**Hazard Analysis Models** Aside from approaches and standards neglecting<sup>11</sup> the discussion of system models, methods using such models can be classified based on the Sections 2.2.1 and 3.2: The first class of approaches<sup>12</sup> uses *structural techniques*. The

<sup>11</sup>See, e.g. Johnsen et al. 2011, Kelly 1998, Lindholm et al. 2012, Stålhane et al. 2012, WHO 2012.

<sup>12</sup>See, e.g. Biehl et al. 2010, Bowles and Wan 2001, Catino and Ungar 1995, Chen et al. 2008, Kath and Temple 2012, McDermid and Pumfrey 1994, Mehrpouyan 2011, Papadopoulos et al. 2001, Pock 2012, Svedung and Rasmussen 2002, Venkatasubramanian et al. 2000, Waters and Ponton 1989, Zhang et al. 2010 and ISO Std. 26262 (2011).



second class uses *behavioural techniques*. This class applies *transition systems*<sup>13</sup> and MARKOV models (Feather 2004, Sayre et al. 2001). The third class<sup>14</sup> uses *mixed techniques*: for example, Rauzy (2002) proposes *mode automata* to be used as a modular reliability formalism. Mode automata and mode transition systems share common foundations in the direction of Harel and Politi (1998), Mealy (1955) and Moore (1956).

The reviewed approaches consider the modelling of electronic hardware (Yenigün et al. 1999), mechanical and other physical views<sup>15</sup>, software<sup>16</sup>, and multiple domains at once (Leveson 2012, Leveson and Stolzy 1987). Abrial (2006) verifies safety goals of a train control system by modelling the rail environment to typify the system interface and to understand the control problem. D’Ippolito et al. (2011) investigates the dynamics of the environment, the technical system and its control subsystem. In summary, the Tables A.8 and A.9 classify these approaches according to their underlying formalism.

**Safety-related Defects and their Representation** Generic *defect taxonomies* (Chillarege et al. 1992) are rare, vague or difficult to use in practice. The variety of perception of defects motivates the use of specific taxonomies: defect classification has been investigated to evaluate the effectiveness of testing techniques (Illes and Paech 2007, Mariani 2003) and for distributed software systems (Hummer et al. 2012). Known from system testing and diagnosis (Section 2.2.3), fault or hazard finding scenarios do not have to result from a defect report or a specification. In the same way, *defect models* for hazard analysis can contain knowledge not derivable from a design or realisation.

Defect models can describe defective component functioning, interfaces and data flow by wrong, missing or untimely values<sup>17</sup>, material wear out and damage<sup>18</sup>, and side effects or defective physical interaction of system parts (Mehrpooyan 2011). Struss and Fraracci (2011) and Dobi et al. (2013) model defects of physical components as behavioural deviations propagable through a structural system model. For such components, Catino and Ungar (1995) proposed a defect model library. Breitling (2000) and Pister (2008) discuss modular defect modelling by modifying data flow specifications. Enhancing this idea, Botaschanjan and Hummel (2009) introduce extension patterns for such specifications based on transition systems. Damm and Peikenkamp (2004) and Gärtner (1999) extend structural models with fault indicators for each system component.

Leveson (2012) captures defective and hazardous states as constraints on the system

<sup>13</sup>See, e.g. D’Ippolito et al. 2011, Gleirscher 2011, Heitmeyer et al. 1998, Neogi 2002, Probst 1996, Stursberg et al. 1998, Voge and Bunimov 2012.

<sup>14</sup>See, e.g. Esser and Struss 2007, Hall and Silva 2008, Herrmann and Krumm 1999, Leveson 2012, Leveson and Stolzy 1987, Nissanke and Dammag 2002, Peikenkamp et al. 2006, Rauzy 2002, Roth and Liggesmeyer 2013.

<sup>15</sup>See, e.g. Catino and Ungar 1995, Herrmann and Krumm 1999, Mehrpooyan 2011, Struss and Fraracci 2011, Waters and Ponton 1989.

<sup>16</sup>See, e.g. Biehl et al. 2010, Chen et al. 2008, D’Ippolito et al. 2011, Heitmeyer et al. 1998, McDermid 2002, Pister 2008.

<sup>17</sup>See, e.g. Biehl et al. 2010, Bowles and Wan 2001, Chen et al. 2008, McDermid and Pumfrey 1994.

<sup>18</sup>See, e.g. Botaschanjan and Hummel 2009, Catino and Ungar 1995, Struss and Fraracci 2011.

### 3. Safety

state space. Being less scalable, [Voge and Bunimov \(2012\)](#) directly model defective states in a transition diagram. According to [Lampport's](#) viewpoint (cf. Section 3.1), [Broy \(2012\)](#) perceives a hazard as an unwanted behavioural property and an incident as a run which models such a property. More specifically, [Broy](#) classifies hazardous defects according to their relationship to the system boundary and to modes. He speaks of *extrinsic* and *intrinsic* hazards guiding the analysis of causal factors: the former are indirectly caused by the system and occur outside the system boundary, the latter are directly caused by the system and occur at the boundary.

#### 3.4.2. Causative Reasoning for Safety

**From Mishap or Hazard to Internal Causal Factor** *Implicit model:* This direction can be carried out using deductive techniques as mentioned in Section 3.2.

*Structural:* [Papadopoulos et al. \(2001\)](#) and [Chen et al. \(2008\)](#) focus on reliability analysis of electronic safety measures. The tool they propose allows FTA via synthesis of fault trees based on fault-to-failure propagation through a design model. [Biehl et al. \(2010\)](#) show early stage analysis of control software for safety-oriented redesign.

*Behavioural:* [Neogi \(2002\)](#) describes a transition system-based technique ([Leveson et al. 1998](#)) to search backwards for paths leading to a previously specified hazardous state. For an autonomous braking system, [Voge and Bunimov \(2012\)](#) exemplify hazard analysis and derivation of safety goals using transition systems according to [Mealy \(1955\)](#). The modelling of hazards as state labels, however, may hinder the scalable description of hazardous behaviour.

*Mixed:* Preceding the work of [Neogi \(2002\)](#), [Leveson and Stolzy \(1987\)](#) experimented with timed PETRI nets: based on a model of a technical system, its environment and defects, they derive fault tolerance and safety requirements for the software-intensive control subsystem. The authors, however, refrained from probabilistic reasoning. [Rauzy \(2002\)](#) shows the transformation of mode automata into fault trees by compilation into BOOLEAN formulae. This step should improve efficiency in reliability assessment and simplify the design and maintenance of BOOLEAN models. Nevertheless, the sequencing among events, which can aid in causal factor search, is lost through this transformation. In summary, the author neglects safety-related topics aside from reliability. [Damm and Peikenkamp \(2004\)](#) sketch verification based on a defective transition system<sup>19</sup> which undergoes reachability checks for hazards to generate fault trees. The approach of these authors avoids too pessimistic fault trees but it is constrained to software and electronic hardware. They omit any discussion of guidance. [Roth and Liggesmeyer \(2013\)](#) apply a qualitative, dynamic variant of FTA to the analysis of software safety. This extension of static FTA can encode event sequences similar to the approach of [Dugan et al. \(1992\)](#). In contrast, the extension is based on stochastic PETRI nets and further develops ideas as discussed in [Leveson and Stolzy \(1987\)](#).

---

<sup>19</sup>The control system is modelled in IBM Rational Statemate ([www.ibm.com](http://www.ibm.com)).

**From Internal Causal Factor to Hazard or Mishap** *Implicit model:* This direction can be addressed by inductive techniques as discussed in Section 3.2. Based on textual specification patterns and generic equipment failure modes, Stålhane et al. (2012) sketch the derivation of HAZID and FMEA tables.

*Structural:* Bowles and Wan (2001) perform FMEA for a control system comprising connected modules of electronic hardware and software. Snooke and Price (2011) discuss abstraction of program code to perform data flow-based safety analysis. Pock (2012) formalises fault propagation through hierarchical data flow models. Waters and Ponton (1989) and Catino and Ungar (1995) presented propagation of component faults (i.e. value and time deviations) through a structural model of a chemical plant given as a set of qualitative equations. Reese and Leveson (1997) use qualitative deviation analysis to investigate software safety. Struss and Fraracci (2011) apply a similar method based on constraint solving to predict such propagations for the diagnosis of a mechanical braking subsystem of a road vehicle, Dobi et al. (2013) for a retarder subsystem. Mehrpouyan (2011) identifies hazards by classifying ways of physical component interaction and applies SysML (Friedenthal et al. 2008) to HAZOP.

*Mixed:* Based on a transition system, Stursberg et al. (1998) applied reachability analysis with high abstraction for early stage hazard analysis of a physical process in a chemical plant and with low abstraction for verification of its control subsystem. Venkatasubramanian et al. (2000) propose PETRI net based HAZOP for chemical plants.

Moik (1999) and Bitsch et al. (1999) refine FMEA by modelling system structure using UML class diagrams and state charts. Their aim is to derive safety requirements (via OCL constraints) as state invariants, and reactivity or interaction requirements (via UML sequence diagrams) as temporal logic assertions, both to be checked for a revised system model. Adopting ideas from Dwyer et al. (1999), Bitsch (2001) provides property patterns for a taxonomy of safety requirements. The authors exemplify their approach for a vehicle braking subsystem and a train subsystem for assistance in passing crossings. Their approach lacks precise notions of hazard and mishap, and details on hazard treatment.

Esser and Struss (2007) apply constraint solving to fault model-based testing of control software by refuting transition system defects. Such defects represent negative hypotheses about system behaviour. They follow an approach to behavioural modelling in accordance with Stursberg et al. and the present work. Pister (2008) combines the formalism of Broy and Stølen (2001) with software FMEA.

David et al. (2010) perform FMEA based on SysML. The authors derive a defect model from a system design using a defect pattern library. For reliability analysis, the formalisation of SysML uses transition system and MARKOV chain semantics. Failure modes of components are obtained from the defect model and the FMEA results. From the defect model and the failure modes, which are added to the nominal SysML model, failure probabilities are calculated. The authors apply their approach to a system controlling the fill level of a tank. Mhenni et al. (2012) apply the approach of David et al. to aileron actuation for an aircraft. Mhenni et al. show how a safety measure can be derived from the analysed SysML model.

### 3. Safety

**Between Internal Causal Factor and Hazard (bidirectional)** *Structural model:* [McDermid and Pumfrey \(1994\)](#) and [Fenelon et al. \(1994\)](#) investigated a method for safe software design where they apply HAZOP and defect classification to data flow models. After having modelled the functionality, guide words are applied to derive component and flow (i.e. value and time) defects which are deductively (i.e. via FTA) traced back to causative faults. Then, the defects and faults are used to inductively (i.e. via FMEA) determine further defects. Selected defects are treated by design modifications. [Wilson and McDermid \(1995\)](#) went further in the integration of several safety analysis techniques by way of consistency rules.

In addition to [Bowles and Wan \(2001\)](#), who only use FMEA, [Zhang et al. \(2010\)](#) apply FTA to a data flow model of automotive control software. Similarly, [Kaiser et al. \(2003\)](#) combine FMEA with FTA based on a hierarchical system model. For the simultaneous analysis of software requirements and hazards, [Feather \(2004\)](#) uses a PRA-based FMEA approach extended by an FTA: faults and their treatment are assigned with impacts on safety goals for cost-benefit calculations of design alternatives. [Yenigün et al. \(1999\)](#) formalise hazards as defective signals causing circuit failure. The authors apply model checking to electronic circuit designs to assure hazard freedom.

*Mixed:* [Johnson \(1993\)](#) proposed a probabilistic extension of computation tree logic to characterise maloperation of human operators when using technical systems, particular, in the event of system failures. He aimed to overcome weaknesses of FTA, FMEA and MARKOV models which are also addressed by, for example, dynamic FTA ([Dugan et al. 1992](#)) and [Rauzy \(2002\)](#). [Johnson's](#) logic combines the contingent and frequency views of probability, that is, historic frequencies of states are taken as best approximations for the probabilities of next states. He presents PROLOG-based MONTE CARLO simulation support for his method. More recently, [Peikenkamp et al. \(2006\)](#) discuss tool support for a combination of FTA, fault injection and FMEA.

[Heitmeyer et al. \(1998\)](#) apply a method to detect defects in software requirements by model checking safety properties of transition systems. Like [Snooke and Price \(2011\)](#), they use program abstraction to improve efficiency. [Probst \(1996\)](#) investigated a similar approach based on checking temporal logic assertions on a transition system model of a chemical plant. [Abrial \(2006\)](#) derives model refinements to fulfil safety requirements. He, however, refrains from defect modelling and hazard analysis. [Herrmann and Krumm \(1999, 2000\)](#) exemplify hybrid, modular property verification of control systems based on TLA ([Lamport 2002](#)). These authors show a proof pattern for a control subsystem of a chemical plant; this subsystem realises a fail-safe pattern (see page 38) that ensures hazard-avoiding state invariants. [Herrmann and Krumm](#) express hazards as constraints on variables defining the physical state space.

[Nissanke and Dammag \(2002\)](#) show for a nuclear reactor control system how safety analysis can be done using the formalism of [Harel and Politi \(1998\)](#). The authors distinguish functional from safety requirements and use a defect model. The defect model captures equipment failures by a non-operational state reached after a failure event and left again via a maintenance/repair action. Transitions can be safe, unsafe and neutral, and can be annotated with *timed safety clauses*. [Nissanke and Dammag](#) propose a risk-based ordering and classification of states to identify both analysis gaps and unpredictable non-deterministic behaviour. The concept of *risk distance* resolves

such behaviour by preferring transitions which reduce hazards.

Haxthausen et al. (2011, 2014) approach safety analysis by using a verifiable transition system technique for the abstraction of low-level program text into a behavioural model of the control system. Based on this model, temporal safety properties are verified using bounded model checking. The authors apply their approach to the software parts of a train control system.

**Between Hazard and Mishap (bidirectional)** *Implicit model, mainly linear causal chain:* Approaches such as AcciMaps, CRIOP, ECF, HERMES and ICPS (Section 3.2) drop into this category although they selectively use physical simulation. As in CRIOP, Sayre et al. (2001) extend MARKOV chain based usage profiles of medical device control systems to identify, quantify and mitigate hazardous maloperation. Sayre et al., however, refuse to explain whether they apply a system model in addition to their model of usage processes. In this class of approaches, the functional resonance accident model (FRAM; Hollnagel 2004) is the only non-linear approach known to me.

*Behavioural, non-linear:* Mode transition systems and A/G style (Section 2.2.2) capture interaction relationships between the system and its environment, and make subtle defects explicit. Dasgupta (2012) verifies safety properties of a combined model of the electronic controller and its environment. He applies A/G style to restrict the behaviour of the combined model. Defective and hazardous states are specified as behavioural properties. Dasgupta (2006) reports on fallacies (e.g. hidden vacuity) when specifying such properties.

D’Ippolito et al. (2011) apply controller synthesis from temporal logic formulae: under certain assumptions, such a controller allows its actuations being erroneous (e.g. wrongly affecting the environment and temporarily violating these assumptions) and, though, guarantees achieving its prescribed safety properties.

*Mixed, non-linear:* Inspired by Rasmussen (1997) and FRAM, the “system-theoretic accident model and processes” approach (STAMP; Leveson 2012) perceives safety as a control problem in a collaboration of humans and technical systems (cf. Section 3.1). Mishaps and hazards are explained by a *non-linear model of causation* where interactions within this collaboration violate safety constraints and lead to hazardous states. Applying the SpecTRM method (Leveson et al. 1998), STAMP classifies human errors, identifies inadequate control aside from system failures and derives required constraints. In addition to preventing failures and technical root causes, these constraints shall be enforced by the collaboration. Dulac (2007) applies qualitative modelling to STAMP; Stringfellow (2010) extends STAMP using HAZOP-like *guide words* to identify maloperation.

Inspired by Zave and Jackson (1997), Hall and Silva (2008) show a model for mishap analysis of technical systems for guiding safety analysis meetings. The authors consider environment modelling, design- and operation-time views, and a *deviation model* of states and actions for defect classification (i.e. human errors, environmental defects). They also distinguish operator and actual system views.

### 3. Safety

#### 3.4.3. Safety Engineering Guidance

Pyle (1991) proposed a safety analysis comprising the viewpoints *victim*, *plant* and *control subsystem*, regarding situations where the entire technical system operates as intended and where it ceases doing so. Based on these viewpoints, he described form-guided steps to analyse *safety* for victims, *dangers* and *protection mechanisms* of the plant, and *detection equipment*, *actuators* and *guards* of the control subsystem. This approach is complementary to the ideas of Wilson and McDermid (1995), who put emphasis on the integration of several safety analysis techniques to reduce the risk of oversights.

Kelly (1998) refines the concept of *safety cases*: A safety case forms a hierarchy of arguments<sup>20</sup> built on evidence using various measures to achieve safety goals on top of this hierarchy. The author discusses argumentation patterns (e.g. for hazard avoidance) and fits FMEA into his method. Rushby (2010) and Hall et al. (2007) formalise such argumentations. Safety cases provide a generic way to build up individual lines of argumentation whereas safety standards (see below and page 33) often carry a specific line of argumentation for an application domain. Being applied in a safety case, standards and modelling techniques can contribute to “the argument that a system is acceptably safe to operate in a particular context” (Kelly 1998). Leveson (2011) identifies the possibility of confirmation biases of such argumentations. Hauge and Stølen (2014) discuss procedural patterns for planning specific safety engineering activities in control systems engineering. The authors discuss the use of safety cases in their approach. In connection with safety cases, van Lamsweerde (2009) uses goal graphs to perform hazard analysis. Similar to the use of guide words in HAZOP, he provides a taxonomy of obstacles which aids in hazard identification.

Lund et al. (2011) describe a stepwise approach to the analysis, documentation, probabilistic assessment and treatment of risks. The authors describe a graphical language for the modelling of assets, threats and risks. *Risk graphs* are formalised using a trace-based probability space similar to probabilistic linear temporal logic. Likelihoods can be given by probabilities and frequencies. In addition to the work of Johnson (1993), the authors use frequency intervals to approximate the probabilities of events. *Frequencies* include times of exposure and relate to probabilities of occurrence; *consequences* of risks concern severity. Lund et al. apply A/G style by splitting risk graphs into assumption and guarantee parts, meaning: if any vulnerability of an asset is exploited then some asset will be harmed, both exploitation and harm within specified probability intervals.

Thramboulidis and Scholz (2010) present hazard analysis based on SysML and a mechatronics engineering process, exemplified for a self-propelled train. Methods such as preliminary hazard analysis (PHA), FMEA and FTA are mapped to the stages of requirements analysis, architecture design, and component specification and design. Among the concepts for hazard analysis, they use the term *system misbehaviour* to hint at specification defects.

---

<sup>20</sup>Kelly applies the *goal structuring notation (GSN)* to depict such a hierarchy.

**Experience on Application of Standards and Methods** Feather and Markosian (2011) describe the elaboration of a safety case for a space vehicle failure warning system. They report on problems when initialising the argument structure. Wagner et al. (2010) discuss an application of GSN-based safety cases (Kelly 1998) and propose assurance patterns similar to the goal refinement patterns of van Lamsweerde (2009). Wagner et al. propose several patterns to be applied in the construction of safety cases: For example, a *fail-safe* pattern includes the transition to a safe state after detection of a hazardous defect. A *failed expectations* pattern prescribes documented specifications for hazards to be identified. Operators mostly or unconsciously expect that the *specification, they assume*, is actually realised by the system. This pattern corresponds to the operator and actual system views considered by Hall and Silva (2008). Wagner et al. perform an FMEA and match the results with the safety case structure, leaving it unclear whether this step was done inductively.

Kath and Temple (2012) report on a tool-based application of ISO 26262 to a vehicle steering support system. Their procedure, shown in Table A.10, is traceable but lacks the use of behavioural models as opposed to, for example, the work of Abdulkhaleq and Wagner (2013) or the present work. Abdulkhaleq and Wagner use goal graphs to decompose safety goals into safety requirements, after item definition and HARA. Then, they assign integrity classes to these requirements and the underlying system parts. An FTA of a preliminary architecture design is followed by the design of safety measures together with quantitative reliability assessments using FTA and FMEA to verify the given integrity classes.

For an avionic subsystem, Mannering et al. (2007) use problem frames (Jackson 2001) to make ARP 4761 traceable during requirements validation. Hence, the authors apply lightweight FMEA and FTA. Lindholm et al. (2012) studies the application of ISO 14971 for analysing risks of patient monitoring systems. The authors identified software risks according to Boehm (1991) by the help of textual use cases, and together with an observed and interviewed safety analysis team. Their analysis neglects the use of models and the classification of operational situations. Lindholm et al. report on the problem of an unclear system boundary, difficulties in slicing of causal chains and imprecise assessment due to missing knowledge of the environment.

## Behavioural Safety: Concepts

This chapter introduces safety-related system modelling, discusses kinds and representation of defects and provides instruments for hazard analysis and treatment.

### Contents

4.1	System Specification: A Safety-related Framework . . . . .	46
4.2	Safety-related Defects . . . . .	48
4.2.1	Taxonomy . . . . .	49
4.2.2	Representation . . . . .	51
4.3	Mishaps, Hazards and Causal Factors . . . . .	53
4.4	Behavioural Safety . . . . .	56
4.5	Responsibility and Restriction . . . . .	59
4.6	Safety Measures . . . . .	60
4.6.1	Treatments for Mode Transition Systems . . . . .	61
4.6.2	Treatments for Behavioural Property Assertions . . . . .	62
4.7	A Stop Criterion for Safety-oriented Validation . . . . .	65
4.8	Notes and Further Reading . . . . .	65

## 4.1. System Specification: A Safety-related Framework

We apply the framework of Section 2.4 to a *technical system* and its *environment*:

- **Definition 4.1 (World Model)** We define  $\mathcal{M} = \mathcal{A}_E \otimes \mathcal{A}_S$  to be the world model consisting of a system agent  $\mathcal{A}_S$  and an environment agent  $\mathcal{A}_E$ .

Adapted from Figure 3.2,  $\mathcal{M}$  abstracts from control software, sensors and actuators to the *system boundary*, see Figure 4.1.  $\mathcal{V}$  holds states, events and modes of physical or conceptual entities to be *monitored or controlled* by the system, the environment, both (e.g. temperature of a metal bar, fill level of a water tank) or none (e.g. state of



#### 4.1. System Specification: A Safety-related Framework

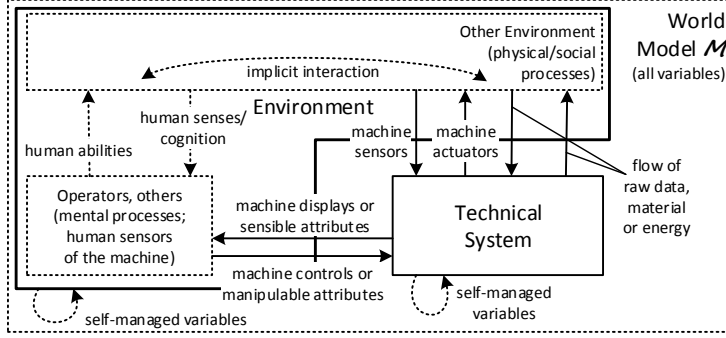


Figure 4.1: World model; channels (solid arrows) constitute the system boundary, dotted arrows indicate self-managed variables

fatigue strength, a constant threshold). The set  $\mathcal{V}_{if} \subseteq \mathcal{V}_e$ , given by

$$\mathcal{V}_{if} \triangleq (\text{ctr}(\mathcal{A}_E) \cap \text{mon}(\mathcal{A}_S)) \cup (\text{ctr}(\mathcal{A}_S) \cap \text{mon}(\mathcal{A}_E))$$

forms the *interface* between the two agents.  $\mathcal{V}_m$  is internal, that is,  $\mathcal{V}_m \cap \mathcal{V}_{if} = \emptyset$ . We call a variable *self-managed* if it is element of  $\mathcal{V}_m$ , models a timer, or is monitored and controlled by the same agent.  $\mathcal{M}$  uses *functional channels* to embody physical or information processes to be controlled and *control channels*<sup>1</sup> for sensors, controls, actuators and displays to observe and affect such processes. Both agents comprise three disjoint *aspects* (Definition 2.14):

- $\mathbb{M}_{\text{use}}$ , *functionality* or specified behaviour
- $\mathbb{M}_{\text{fail}}$ , a *defect model* comprising operational defects based on  $\mathbb{M}_{\text{use}}$
- $\mathbb{M}_{\text{save}}$ , *safety measures* for the treatment of hazards in  $\mathbb{M}_{\text{use}}$  and  $\mathbb{M}_{\text{fail}}$ .

We call an MTS (Definition 2.6) a *function* if it is part of  $\mathcal{A}_S$  and a *tactic* if it is part of  $\mathcal{A}_E$ . If an MTS is an element of

- $\mathbb{M}_{\text{use}}$ , it comprises *specified functional and control actions*. Its *modes* constitute distinct operational phases described by functional actions.
- $\mathbb{M}_{\text{fail}}$ , it comprises *defective actions*, and modes reachable by such actions.
- $\mathbb{M}_{\text{save}}$ , it comprises *safety actions* maintaining or achieving safe states, and modes reachable by such actions.

<sup>1</sup>The mechanisms underlying these channels can comprise, for example, electromechanical, electrothermal, thermomechanical, hydromechanical, pneumatic, electronic and software devices.

#### 4. Behavioural Safety: Concepts

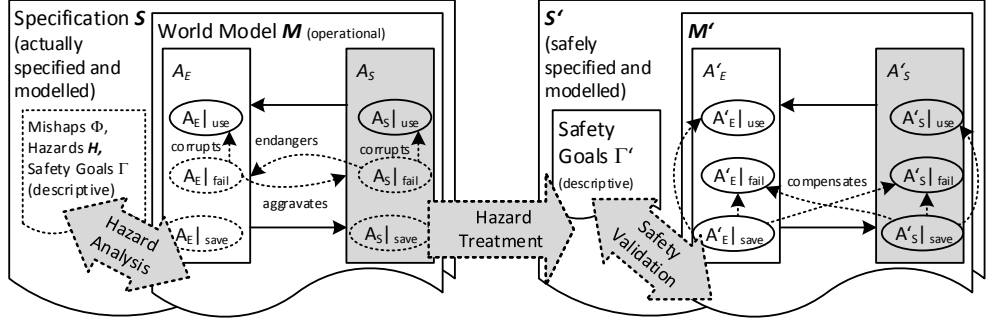


Figure 4.2.: Actual and safe description and representation of a specification

A defective action or a safety action can be either a functional or a control action. Furthermore, our assumption is that the following action priorities are realised:

$$\text{prio} = \begin{cases} [2, \infty) \text{ or low} & \text{for specified actions } (\mathbb{M}_{\text{use}}) \\ [1, 4] \text{ or medium} & \text{for defective actions } (\mathbb{M}_{\text{fail}}) \\ [0, 3] \text{ or high} & \text{for safety actions } (\mathbb{M}_{\text{save}}) \end{cases} \quad (4.1)$$

- **Definition 4.2 (Operational Situation)** We call an abstract state (Definition 2.11) an operational situation iff it is used to specify a set  $\Sigma_0$  of initial states for  $\mathcal{M}$ .

Operational situations can describe several states of the environment and the system at once, for example, a driving situation can comprise road and vehicle conditions.

For hazard analysis (Section 3.2), we consider the abstract states *defective*, *harm*, *hazardous*, *safe* and *operational*. *Behavioural properties* describe temporal, possibly causal, relationships among these states: *Hazards* characterise behaviour potentially leading to mishaps. *Safety goals*, denoted by the set  $\Gamma$ , specify the treatment of hazards. These goals separate required and unwanted behaviour from occasionally acceptable or unspecified behaviour. *Domain properties* help encode logical and physical relationships among variables indirectly updated by actions, and constrain valuations of  $\mathcal{V}$  to be maintained by actions. According to Figure 4.2, we consider a variant of the first pair of *system views* from Section 2.4:

**Actually specified and modelled** The *actual* description and representation of a specification  $\mathcal{S}$ .

**Safely specified and modelled** The *safe* description and representation of a specification  $\mathcal{S}'$ , to be finally achieved. We will use  $\mathcal{V}' \triangleq \mathcal{V}_{\text{safe}}$  instead of  $\mathcal{V}_{\text{ideal}}$ .

## 4.2. Safety-related Defects

This section develops defect modelling (Section 2.2.3) by describing defects as *deviations* between two artefacts, for example, a specification and a realisation.

Criterion	Facet	Occurrence
1. Agent		(s)ystem, (e)nvironment
2. Observation	a. Reproducibility	(s)ystematic, (r)andom, (s)emi-(s)ystematic
	b. System view*	(r)ealised and operated, specified and (m)odelled
	c. Causal chain pos.	(w)ithin an agent or (a)t its boundary
	d. Duration	(p)ermanent, (t)ransient
3. Cause:effect participation		1:0, 1:n, 1:n̄, 1:1, n:1, n:m
4. Coverage	a. Degree	(n)one, (p)artial, (c)omplete
	b. Responsibility	(s)ystem, (e)nvironment
5. Life cycle origin*		(s)pecification, (d)esign, (r)ealisation, (o)peration
6. Mode of performance		unacceptable (e)xecution or (s)uppression

Table 4.1.: Criteria for a taxonomy of safety-related defects; \*... multiple choices possible

**Definition 4.3 (Defect)** *A defect is an observable and unacceptable deviation of an actually observed artefact from an ideal artefact.* ⌈  
⌋

The concepts in Section 4.1 adopt this definition in two ways: The first way, provided by having  $\mathcal{M}$  and  $\Gamma$  in the two system views  $\mathcal{S}$  and  $\mathcal{S}'$ , leads to the following definition:

**Definition 4.4 (Specification Defect)** *A specification defect is a defect of  $\mathcal{S}$  with respect to  $\mathcal{S}'$ , that is, a deviation of the actual from the safe specification.* ⌈  
⌋

The second way, provided by the aspect  $\mathbb{M}_{\text{fail}}$ , motivates the next definition:

**Definition 4.5 (Operational Defect)** *An operational defect is a defect of  $\mathcal{W}$  with respect to  $\mathcal{M}'_{\text{use,save}}$ , that is, a deviation of the realisation from the safe world model without defects.* ⌈  
⌋

Let  $\widehat{\Gamma} \equiv \bigwedge_{\gamma \in \Gamma} \gamma$  denote the conjunction of all safety goals. Definition 4.4 motivates reasoning about  $\mathcal{S} \neq \mathcal{S}'$  and  $\mathcal{M} \not\models \widehat{\Gamma}'$ , Definition 4.5 whether  $\mathcal{W}$  conforms to  $\mathcal{M}'$ .

**Corollary 4.1** *Definition 4.4 yields that every inconsistency of the world model without defects and the safety goals constitutes a specification defect, denoted by  $\mathcal{M}_{\text{use,save}} \not\models \widehat{\Gamma}$ . Definition 4.5 implies that  $\mathbb{M}_{\text{fail}}$  is in the class of operational defects. The two sets of defects are neither disjoint, nor does one contain the other.*

### 4.2.1. Taxonomy

Defects can be characterised by the *criteria* given in Table 4.1 as explained below:

#### 4. Behavioural Safety: Concepts

1. **Agent** A defect can arise internally, as a (s)ystem defect, or externally, as an (e)nvironment defect, depending on where the deviations are caused or observed.
2. **Observation** This criterion comprises four facets:
  - a. Concerning *reproducibility*, we can distinguish *systematically* (reproducible stimuli and state, e.g. specification, design or realisation errors), *randomly* (unknown stimuli and state but knowable mode, e.g. hardware damage or material wear out) and *semi-systematically* (reproducible stimuli and mode but partially unknown state) recurring defects. Their first observed occurrence may seem to be spontaneous.
  - b. A defect can be *realised* (observed at runtime in  $\mathcal{W}$ , Definition 4.5), *modelled* (observed in  $\mathcal{S}$ , Definitions 4.4 and 4.5) or both at the same time.
  - c. A defect can show up *within* an agent (e.g. hardware or software component fault, human operator error) or *at its boundary*. According to the *causal chain position*, defects at the boundary are known as *failures*, defects within an agent as *faults*, *weaknesses or errors* (root causal factors of failures), and *erroneous states* (observations between faults and failures).
  - d. Concerning the *duration*, a defect can be *transient* (observable spontaneously at a certain point, diminishing or disappearing without treatment) or *permanent* (observable during the entire period of operation, staying unless treated).
3. **Cause:Effect Participation** There are *defects without effect* (1:0), *groups of dependent defects* and *defects caused by single or multiple factors*. Dependent defects can be split into *common cause* (1:n, multiple failures directly sharing the same causative fault), *common mode* (1:n̄, multiple similar failures directly sharing the same causative fault) and *cascading* (m:n, multiple failures indirectly sharing the same causative faults) failures. Regarding causal factors, one can differentiate between *single point* (1:1, caused by a single fault) and *multiple point* (n:1, caused by multiple faults) failures.
4. **Coverage** Defects can be (a) *completely, partially or not* (diagnostically) covered by safety measures of (b) the *system* or its *environment*. Partial coverage of a single point failure leaves over a *residual fault* not covered by a safety-related subsystem and leading to a failure. For multiple point failures, coverage spans *detected faults* (treated by a safety-related subsystem), *perceived faults* (observed by a human operator) and *latent faults* (not treated by a safety-related subsystem or human operator). *Safe faults* are either detected faults or they do not cause failures (1:0).
5. **Life Cycle Origin** Defects can originate from the stages of (*s*)*pecification* (wrong conception of the control problem, Definition 4.4), (*d*)*esign* (improper choice of architectural means or technologies), (*r*)*ealisation* (erroneous fabrication, implementation, integration or maintenance), and (*o*)*peration* (damage or wear out after inappropriate use or lack of maintenance and repair).

6. **Mode of Performance** Runs deviating from  $\mathcal{A}_S|_{\text{use}}$  or  $\Gamma$  (e.g. functional and timing failures) show *unacceptable*<sup>2</sup> *(e)xecutions or (s)uppressions* of specific actions in specific operational situations.

**Relationships among the Criteria** The criteria 1, 2b and 3 characterise the location where the impacts or effects of a defect can be observed, whereas criterion 5 describes locations of potential root causes of a defect. Criterion 2b covers whether a defect is modelled or realised, without regarding its causes. Criterion 2a captures state of knowledge and establishment of conditions for reproducing a defect, whereas criterion 2d cares about loss of these conditions. Hence, transient defects can be random defects. The criteria 2a and 4 match “probability of occurrence” and “detectability and controllability” from page 36. The criteria 5(s) and 6 address the Definitions 4.4 and 4.5 whereas the criteria 1 to 4 are based on Definition 4.5. Table A.15 applies the taxonomy to safety-related classes of defects.

#### 4.2.2. Representation

This section shows how operational defects can be *modelled* in  $\mathcal{M}$ , how such defects can be *described more generally* and which specification defects may *appear* in  $\Gamma$ .

**Representation of Operational Defects by  $\mathbb{M}_{\text{fail}}$**  Based on  $\mathbb{M}_{\text{use}}$ , we apply *MTS patterns* to obtain a *defect model* comprising

- *defective actions*: actions producing defective states, for example, by under-specification and indeterminacy using missing, erroneous or inconsistent guards and triggers, and ill-timed, non-deterministic or probabilistic deviations from specified effects;  $\mathcal{V}$  stays the same.
- *defective modes*: MTS fragments (Definition 2.8) with only defective actions;  $\mathcal{V}_m$ 's types are extended.
- *fault indicators*: values or variables affected by specified or defective actions to model defective states (cf. Table 4.2);  $\mathcal{V}$ 's types and  $\mathcal{V}$  are extended.

The elements of the defect model can represent, for example,

- systematic or random failures
- sensor and actuator faults (e.g. errors in measurement or state observation)
- unwanted mode dependencies (e.g. physical interaction or message passing)
- physical side effects
- unexpected stimuli (e.g. environment misbehaviour, intentional or unconscious maloperation, erroneous intervention).

<sup>2</sup>That is, for example, unintended, unattended, unexpected or forbidden.

#### 4. Behavioural Safety: Concepts

<i>pre</i>	<i>post</i>							
	$m_0$	$m_1$	$m_2$	$m_3$	$m_4$	...	$m_d$	...
$i_1$	$m_5/o_1$	$m_1/rej$	$m_f/fail$	$m_1/o_3$	$m_4/-$	...	$m_d/fail$	...
*	...	...	...	...	...	...	$m_d/fail$	...

Table 4.2.: Tabular representation of a transition system fragment with fault indicator values *rej* and *fail*, and a defective mode  $m_d$  with defective actions

Systematic failures can stem from *specification defects* which arise from implicit knowledge, mistakes in domain analysis or a wrongly perceived interface (e.g. wrong type abstraction, disregarded variables). Unexpected stimuli can stem from operational defects ( $\mathcal{A}_{E|_{fail}}$ ) of operators (e.g. mode confusion, distraction, fatigue, inattention) or other technical systems. Table A.2 provides MTS patterns applicable in  $\mathbb{M}_{fail}$ . Table 4.2 and Example 4.1 illustrate defect modelling.

- ⌈ **Example 4.1 (Operational Defects of a Car Airbag)** *The MTS shown in Example 2.1 applies twice the MTS pattern random/permanent where  $m_d$  instantiates to failing,  $fail_1$  to  $\tau$  and  $maintain_f$ , and  $fail_2$  to  $expand_f$  and  $\kappa$ .*

The following criteria are disregarded in  $\mathcal{M}$ : Criterion 2b is only considered helpful during V&V. The criteria 2c (i.e. defects within an agent, faults, erroneous states) and 3 are neglected as we omit structural modelling to apply these criteria. Nevertheless, criterion 3 covers unwanted dependencies (cf. Table A.2). Criterion 4 applies to the discussion of safety measures but is unnecessary to express further defects in  $\mathcal{M}$ . For criterion 5, we distinguish *s* and *o* but not *d*, *r* and *o*, as such details can only be observed via several stages of V&V. Hence, the criteria 2c, 3 and 5 are difficult or even impossible to represent and detect in  $\mathcal{S}$  or  $\mathcal{S}'$ .

**Generic Description of Defects** Using abstract states (Definition 2.11) and behavioural properties (Definition 2.15), states and runs can be declared to be *defective*. Definition 4.3, the defect taxonomy and the ways of defect representation yield:

- ⌈ **Definition 4.6 (Defective Mode and State)** *A mode  $m$  (Definition 2.7) is defective (i) if the behaviour of  $\mathbb{A}_m$  is prohibited by  $\Gamma$ , or (ii) if a subset of  $\mathbb{A}_m$  belongs to  $\mathbb{M}_{fail}$ . We use  $df$  to denote the abstract state defective including all states which (i) are declared as defective or (ii) carry a defective mode.*

**fault failure** By this definition, we can describe a *fault* as a part of a state in  $df$  which activates a defective mode. Furthermore, we define a *failure* to be a run  $\rho$  such that

$$\forall \rho' \in \llbracket \mathcal{M}' \rrbracket_{use} : \rho|_{\mathcal{V}_e} \neq \rho'|_{\mathcal{V}_e}$$

Hence, a failure is observable through  $\mathcal{V}_e$  and might visit states of  $df$ . Faults and failures are operational defects (Definition 4.5).

**Appearance of Specification Defects in  $\Gamma$**  The term *property defect* can refer to

- a *missing* or *wrong* property assertion, or an *inconsistent set* of assertions
- an *assumption* about the environment being wrong, incomplete or too strong, or a *guarantee* of the system being wrong or too weak (cf. Section 2.2.2).

More specifically, a property defect can incorporate, for example,

- an *implicit* or *violable independence*, *domain maintenance* or *run conformance* assumption
- *misperception of responsibility*, or a *low-grade* or *violable range maintenance guarantee* based on such an assumption.

### 4.3. Mishaps, Hazards and Causal Factors

This section investigates the relationship between *mishaps*, *hazards* and *causal factors* to be abducted by hazard analysis in  $\mathcal{M}$ .

**Definition 4.7 (Mishap)** A mishap  $\phi$  is a state constraint (Definition 2.4) denoting a sufficient condition for the harm state. Hence, given a set of mishaps  $\Phi$ , the disjunction  $\check{\Phi} \equiv \bigvee_{\phi \in \Phi} \phi$  defines the harm state. ⌈

**Definition 4.8 (Hazard)** A past formula  $\chi$  (Definition 2.15) with at least one variable in  $\mathcal{V}_f$  is called a hazard iff there exists a mishap  $\phi$  for which the formula ⌈

$$\text{EF}(\chi \rightarrow \text{P}_{>\text{Pr}}[\text{XF}\phi]) \quad (4.2)$$

is satisfied in  $\mathcal{M}$ , with an upper bound<sup>3</sup> of acceptable risk  $0 < \text{Pr} \ll 1$ . Then, we say that  $\chi$  is among the preconditions to risk  $\phi$ . Given a set of hazards  $\mathcal{H}$ ,  $\check{\mathcal{H}} \equiv \bigvee_{\chi \in \mathcal{H}} \chi$  makes up the hazardous state. ⌋

As depicted in Figure 4.3, Formula (4.2) states that  $\chi$  searches in each run of  $\mathcal{M}$ , starting at  $\sigma_0 \in \Sigma_0(\mathbf{E})$ , for a state  $\sigma_h$  such that  $\chi$  holds ( $\text{F}\chi$ ) and the probability of  $\phi$  occurring after  $\sigma_h$  ( $\text{XF}\phi$ ) is higher than  $\text{Pr}$  ( $\text{P}_{>\text{Pr}}[\text{XF}\phi]$ ). Note that the probability of occurrence of a hazard (discussed below on page 56) is irrelevant for its *identification*. Hence,  $\text{Pr}$  can be chosen independent of a specific hazard.  $\text{Pr}$  then represents the *risk threshold* which should be exceeded by  $\mathcal{M}$  when regarding any probabilistic relationship between two sets of states for  $\chi$  and  $\phi$ .

A mishap describes *harmful runs* ( $\rho \models \text{F}\phi$ ) modelling many, possibly related *causal factors*. According to Figure 4.4, these factors include defects of the system ( $\mathcal{A}_{\text{E}}|_{\text{use}} \otimes \mathcal{A}_{\text{S}}|_{\text{use, fail}}$ , cf. task T1 on page 36), the environment ( $\mathcal{A}_{\text{E}}|_{\text{use, fail}} \otimes \mathcal{A}_{\text{S}}|_{\text{use}}$ , T2), or both agents ( $\mathcal{M}|_{\text{use, fail}}$ ), denoted by  $\mathcal{H}_{\text{op}}$ , and specified behaviour of both agents ( $\mathcal{M}|_{\text{use}}$ , T1, T2), denoted by  $\mathcal{H}_{\text{pu}}$ . Causal factors can be combined to specify one or more *hazards*. Two questions can guide the characterisation of *hazardous runs* ( $\rho \models \check{\mathcal{H}}$ ):

causal factor

<sup>3</sup>The ALARP concept, mentioned on page 37, might constrain the choice of a guidance level for  $\text{Pr}$ .

#### 4. Behavioural Safety: Concepts

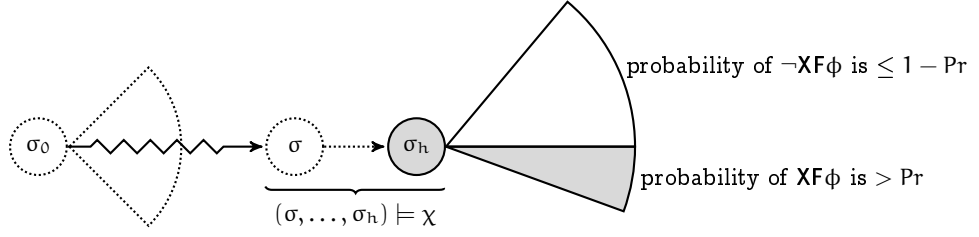


Figure 4.3.: A hazard  $\chi$  in the behavioural spectrum of  $\mathcal{M}$

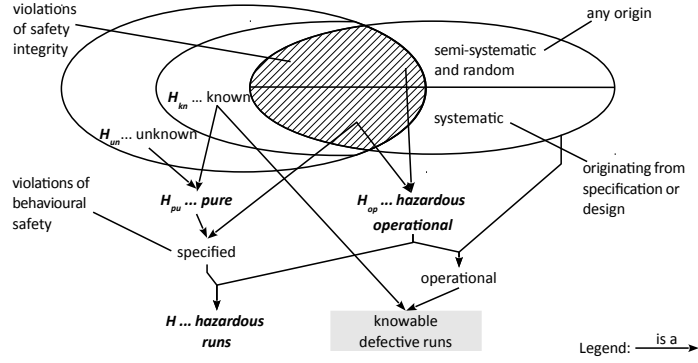


Figure 4.4.: A behaviour taxonomy for causal factor and hazard analysis

$\mathcal{H}_{op}$  Which of the knowable<sup>4</sup> operational defects are causal factors?

$\mathcal{H}_{pu}$  Which of the causal factors are pure specification defects?

The difference between  $\mathcal{H}_{op}$  and  $\mathcal{H}_{pu}$  stems from the regard of causal factors beyond  $\mathbb{M}_{fail}$  (cf. Corollary 4.1). By abduction,  $\Phi$  highlights parts of  $\mathcal{M}$  which indicate hazardous specification defects in  $\mathbb{M}_{use}$ . For example, analyses for task T2 can consider environment defects in combination with specified and defective system behaviour.

**State Guide Words for Mishap Identification and Modelling** Referring to Section 3.2, the following guide words help specify *state constraints* to model mishaps:

- *Areas or spaces* which could be *contaminated*, or where *objects* could *collide*, be *distracted*, *sounded*, *glared*, *burned*, *frozen*, *shot*, *suffocated* or *cooped up*.
- *Places or parts* where *objects* could *trip*, *fall*, *sink* or *bump*, or be *clamped*, *sheared*, *hit*, *scraped*, *cut*, *tired out*, *stuck* or *lost*.
- *Contact surfaces* where *objects* could be *burned*, *vibrated*, *dissolved*, *poisoned* or *electrically shocked*.

<sup>4</sup> $\Gamma$ ,  $\mathbb{M}_{fail}$  and  $\kappa$ -completion can be used to gain this knowledge.



### 4.3. Mishaps, Hazards and Causal Factors

Several state constraint patterns are described in Table A.3: Both the *hazardous element* under control of the system and the affected *asset* of the environment are captured by the variables,  $\mathcal{A}_S$  and  $\mathcal{A}_E$  share with each other. The *initiating mechanism* corresponds to one or more actions of  $\mathcal{A}_S$  and  $\mathcal{A}_E$  affecting these variables.

**Event, Mode and Action Guide Words for Hazard Identification** The following list shows exemplary *guide words for a functional or control event e*:

- *e not or incorrectly given/returned*
- *wrong timing* (e.g. too late/early) or *order of e*
- *e stopped too soon, or applied too long or even permanently*
- *too much/little, many/few, high/low, fast/slow, far/close or hot/cold e.*

The following list shows exemplary *guide words for a mode m and an action a*:

- *unintended, unattended, unexpected, untimely or denied* (i.e. hazardous)
  - *activation/start, deactivation/stop or change of m*
  - *execution or suppression of a* (cf. criterion 6 in Section 4.2.1)
- *hazardous side effect of a, or unwanted a* (e.g. explosion, overheat, fire)
- *maloperated a.*

These guide words help specify *past formulae* to search for causal factors (i.e. hazardous states, events, modes and actions) in harmful runs; Table A.1 provides several property patterns. Example 4.2 applies these patterns to Example 2.1.

**Example 4.2 (Mishap and Hazard Identification for a Car Airbag)** *Suppose an MTS  $\text{Cars} \equiv \mathcal{A}_S$  with a function  $\text{Airbag}_S$  and its environment  $\mathcal{A}_E$ , beyond the cutouts given by Example 2.1 and Example 4.1. The set of initial states is given by  $\Sigma_0 = \{\sigma_0\}$  with  $\sigma_0 = [\text{crashed} \mapsto \perp, \text{energy} \mapsto \text{on}, \text{gas} \mapsto \text{full}, \text{released} \mapsto \text{no}]$ .*

*First, we can use the guide word “distract” (Table A.3) to derive the mishap  $\phi_{\text{distract}}$ :*

$$\begin{aligned} \text{someEventIn}_{\text{Area}} &\equiv \text{released} \geq 2 \\ \text{someSituation} &\equiv \text{m}_{\text{Cars}} = \text{drive} \\ \text{entered}_{\text{person,Area}} &\equiv \text{at}_{\text{driver,seat}} \\ \phi_{\text{distract}} &\equiv \text{released} \geq 2 \wedge \text{m}_{\text{Cars}} = \text{drive} \wedge \text{at}_{\text{driver,seat}} \end{aligned}$$

*Second, we can use event, mode and action guide words from Table A.1 to transform Definition 4.8 into a search expression among the behavioural spectrum defined by  $\mathcal{M}$ . Suppose that 5% risk ( $\text{Pr} = 0.05$ ) is as low as reasonably practicable (ALARP). We can use the pattern “e not given” to derive  $\neg \bar{F}^{\leq 100\text{ms}}$  crashed.*

*We might refine our search using the pattern “unexpected activation of m” to derive a further search expression  $\bar{F}^{\leq 20\text{ms}}(\text{m}_{\text{Airbag}_S} = \text{failing})$ .*

#### 4. Behavioural Safety: Concepts

To derive a hazard from these two causal factors, we have to check in  $\mathcal{M}$  whether

$$\chi_{\text{unexpExp}} \equiv \neg \bar{F}^{\leq 100\text{ms}} \text{crashed} \wedge \bar{F}^{\leq 20\text{ms}} (\text{mAirbag}_S = \text{failing})$$

fulfils Definition 4.8 using the formula

$$\text{EF}(\chi_{\text{unexpExp}} \rightarrow \text{P}_{>0.05}[\text{XF}\phi_{\text{distract}}])$$

which expands to

$$\text{EF}((\neg \bar{F}^{\leq 100\text{ms}} \text{crashed} \wedge \bar{F}^{\leq 20\text{ms}} (\text{mAirbag}_S = \text{failing})) \rightarrow \text{P}_{>0.05}[\text{XF}(\text{released} \geq 2 \wedge \text{mCar}_S = \text{drive} \wedge \text{at}_{\text{driver,seat}})])$$

The quantities 100ms and 20ms can indicate time intervals to leave the hazardous or harm states, that is, to perform the treatment. Analogically, we can derive a mishap  $\phi_{\text{collide}}$  using the pattern “collide”: The mishap could then be observed using variables which model shock sensors built into the system. Nevertheless, the way how to detect a collision can be left open for the construction of safety measures.

**Hazard Characteristics** As indicated on page 36, a hazard  $\chi$  can be characterised by

**S**, the *severity* of a mishap  $\phi$  as  $\chi$ 's potential impact

**W**, the *probability* of occurrence of  $\chi$  or  $\phi$  without the measures of  $\mathbb{M}_{\text{save}}$

**A**, the *exposure* of vulnerable environment assets to  $\chi$  or  $\phi$ , and

**G**, the *detectability* and *controllability* by the environment or the system (without  $\mathcal{A}_S|_{\text{save}}$ ) if  $\chi$  or  $\phi$  occur.

For a mishap  $\phi$ , **S** can be estimated by accident analysts. Hence,  $\Phi$  represents expert knowledge. For **W**, the probabilities of  $\chi$  and  $\phi$  can be calculated from  $\mathcal{M}$  without  $\mathbb{M}_{\text{save}}$ . Restricted to the variables in  $\text{mon}(\mathcal{A}_E) \cup \text{ctr}(\mathcal{A}_E)$ , **A** can be seen as the joint probability of all operational situations where  $\chi$  and  $\phi$  may occur. **G** can be modelled in  $\mathcal{M}$ . These characteristics can be used to refine the classes of runs shown in Figure 4.4. Finally, *hazards* help identify safety requirements and, thus, safety measures.

### 4.4. Behavioural Safety

In this section, we discuss the derivation of *safety goals* from hazards and mishaps. Behavioural safety implies the *safe conception* of  $\mathcal{M}$ , reducing hazards and requiring an acceptable ratio of treated to known hazards. *Safety integrity* (Section 3.3) can be seen as the part of behavioural safety implying *safe realisation and operation* of the system, and requiring sufficiently reliable functional measures specified by  $\mathcal{A}_S$ . As shown in Figure 4.4, hazards describe runs which violate behavioural safety through *pure specification defects* and *operational defects*. Behavioural safety describes the *degree of freedom from hazardous defects* (Definition 4.3), safety integrity the *degree*

of freedom from operational system defects (Definition 4.5). Behavioural safety makes a realisation safe because of safe functionality and valid, reliable safety measures, though tolerating necessarily unreliable functionality. Safety integrity can leave a realisation hazardous, even if functionality and functional measures are reliable.

Behavioural safety can be seen as a facet of correctness of a system's functionality and as a behavioural property of a system and its environment. Specialising the notion of goal from page 12, safety goals specify this property to treat hazards and constrain  $\mathcal{M}$  to reduce specification defects (i.a. unexpected stimuli, T2 on page 36) and operational defects (i.a. system defects, T1).

**Definition 4.9 (Safety Goal)** Given a hazard  $\chi$  and a mishap  $\phi$  (Definitions 4.7 and 4.8), a safety goal  $\gamma$  is a behavioural property (Definition 2.15) defined as ┐

$$\gamma \equiv P_{\geq \text{Pr}}[\mathbf{G}\neg\chi] \quad (\text{avoidance goal}), \quad (4.3)$$

$$\gamma \equiv P_{\geq \text{Pr}}[\mathbf{G}\neg\phi \wedge \mathbf{GF}\neg\chi] \quad (\text{mitigation goal}) \text{ or} \quad (4.4)$$

$$\gamma \equiv P_{\leq 1-\text{Pr}}[\mathbf{F}((\phi \vee \chi) \mathbf{U}^{>k}\neg(\phi \vee \chi))] \quad (\text{alleviation goal}) \quad (4.5)$$

with  $k \in \mathbb{N}$  and  $0 \ll \text{Pr} \leq 1$ .<sup>5</sup> Let  $\Gamma$  be the set of safety goals and  $\hat{\Gamma} \equiv \bigwedge_{\gamma \in \Gamma} \gamma$ . ┘

An *avoidance goal* requires, with a lower bound of probability  $\text{Pr}$ , that the hazardous state is never reached. A *mitigation goal* requires, with a lower bound of probability  $\text{Pr}$ , that the harm state is never reached and the safe state is visited infinitely often. An *alleviation goal* demands, with an upper bound of probability  $\text{Pr}$ , that the harm or hazardous states are visited no longer than a duration  $k$  (cf. Figure 4.5). Note that, unlike in Definition 4.8,  $\text{Pr}$  represents the *risk threshold* to be met by  $\mathcal{M}$  for a specific safety goal. Example 4.3 applies Definition 4.9 to Example 4.2.

**Example 4.3 (Safety Goals for a Car Airbag and a Car)** From hazard  $\chi_{\text{unexpExp}}$ , we can derive a mitigation goal ┐

$$\begin{aligned} \gamma_{\text{treat.distract}} \equiv & P_{\geq 0.90}[\mathbf{G}\neg(\text{released} \geq 2 \wedge m_{\text{CarS}} = \text{drive} \wedge \text{at}_{\text{driver,seat}}) \\ & \wedge \mathbf{GF}\neg(\neg\bar{\mathbf{F}}^{\leq 100\text{ms}} \text{crashed} \wedge \bar{\mathbf{F}}^{\leq 20\text{ms}}(m_{\text{AirbagS}} = \text{failing})))] \end{aligned}$$

as defined by Formula (4.4). This goal represents a constraint for the safety measures to be taken to treat the hazard  $\chi_{\text{unexpExp}}$ .

Continuing from Example 4.2, the further development of an alleviation goal (Formula 4.5) for the mishap  $\phi_{\text{collide}}$  can be represented by the formula

$$P_{\geq 0.9999}[\mathbf{G}(\text{crashed} \leftrightarrow \mathbf{F}^{<350\text{ms}} \text{absorbedBy}_{\text{drv,airb}})]$$

The conjunct  $\text{absorbedBy}_{\text{drv,airb}} \equiv \text{released} = \text{full} \wedge \text{at}_{\text{driver,seat}}$  describes the outcome of the airbag, itself being a safety measure to alleviate  $\phi_{\text{collide}}$  (cf. Example 1.2). ┘

<sup>5</sup>The ALARP concept, mentioned on page 37, might constrain the choice of a guidance level for  $\text{Pr}$ .

#### 4. Behavioural Safety: Concepts

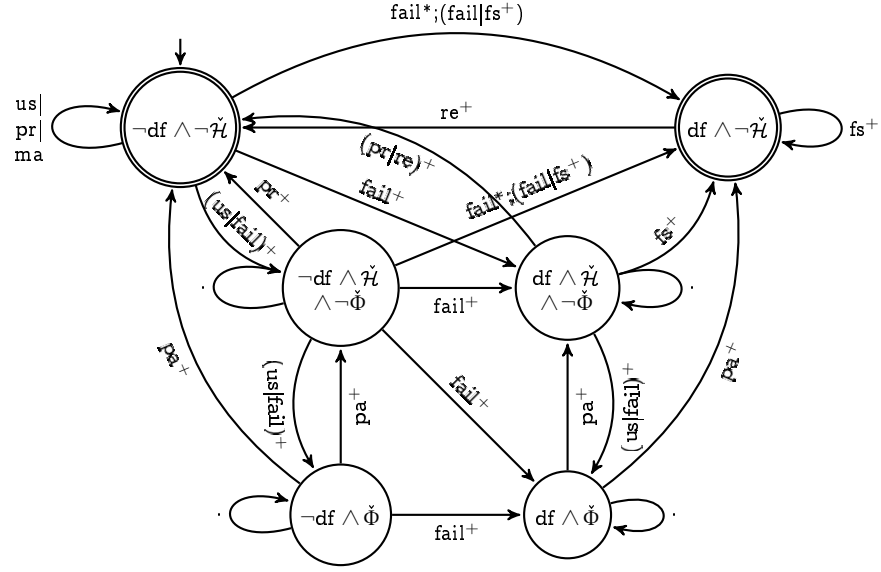


Figure 4.5.: Abstract state space  $\Sigma_\alpha$  for the abstract transition system  $\mathcal{M}_\alpha$  with operational ( $\neg df$ ), defective ( $df$ ), hazardous ( $\check{\mathcal{H}}$ ), harm ( $\check{\Phi}$ ) and safe ( $\neg\check{\mathcal{H}}$ ) states, and the *safety-related action classes* (us)age for  $\mathbb{M}_{use}$ -actions, fail for  $\mathbb{M}_{fail}$ -actions, (f)ail(s)afe, (pr)eventive, (re)pair, (ma)intenance and (pa)ssive for  $\mathbb{M}_{save}$ -actions, as abstracted from  $\mathcal{M}$  via  $\alpha: \mathbb{M} \rightarrow \mathbb{M}$ ; accepting states are encircled twice

□ **Definition 4.10 (Behavioural Safety)** For a specification  $\mathcal{S} = (\mathcal{V}, \mathcal{M}, \Gamma)$ , we define behavioural safety as the degree of freedom from known hazards, given by

$$\mathfrak{BS}(\mathcal{S}) \triangleq \frac{\max\{|\mathcal{G}| \mid \mathcal{G} \subseteq \Gamma \wedge \mathcal{M} \models \widehat{\mathcal{G}}\}}{|\Gamma|} \quad (4.6)$$

□ We speak of  $\Gamma$ -complete behavioural safety of  $\mathcal{S}$  if  $\mathfrak{BS}(\mathcal{S}) = 1$ .

**Hazard Analysis using an Abstract Transition System** Given  $\Phi$  and  $\mathcal{H}$ , with at least one  $\chi \in \mathcal{H}$  for each  $\phi \in \Phi$ , we define an abstract state space  $\Sigma_\alpha$  (Definition 2.11) according to Figure 4.5 for the remainder of this work. Furthermore, Figure 4.5 shows an abstract transition system  $\mathcal{M}_\alpha$  (Definition 2.12).  $\mathcal{M}_\alpha$  classifies runs of  $\mathcal{M}$ , starting from a given operational situation (Definition 4.2), as *abstract runs* possibly containing the *harm state*. Moreover, a mode of  $\mathcal{M}$  is *hazardous* whenever its behaviour intersects with the hazardous state in  $\Sigma_\alpha$ .

**Corollary 4.2** Behavioural safety amounts to verifying  $\mathcal{M} \models \widehat{\Gamma}$ , that is,  $\llbracket \mathcal{M} \rrbracket \subseteq \llbracket \widehat{\Gamma} \rrbracket$ . Safety goals specify safe runs of  $\mathcal{M}$  or abstract runs ending in an accepting state of  $\mathcal{M}_\alpha$ . We can deduce that an abstract run  $\rho_\alpha$  of  $\mathcal{M}_\alpha$  is safe if  $\rho_\alpha \models \widehat{\Gamma}$ .

## 4.5. Responsibility and Restriction

Responsibility comes into play whenever the environment and the system are able to autonomously *restrict* their behaviour observable in  $\mathcal{M}$  to fit  $\llbracket \widehat{\Gamma} \rrbracket$  without having to be constructively enforced. Responsibility indicates situations where such enforcements are required. The following discussion is dispensable if  $\llbracket \mathcal{M} \rrbracket \subseteq \llbracket \widehat{\Gamma} \rrbracket$ .

**Definition 4.11 (Assumption/Guarantee Pair)** *An assumption/guarantee (A/G) pair is a pair of coherent behavioural properties (Definition 2.15): an assumption about environment behaviour and an associated guarantee of system behaviour.*

Enhancing the notion of safety goal, one can state an assumption and a guarantee (Section 2.2.2) for each safety goal according to Definition 4.9:

**Definition 4.12 (A/G-based Safety Goal, Safety Requirement)** *We call a safety goal  $\gamma \in \Gamma$  A/G-based iff there exists an A/G pair  $(As, Gr)_\gamma$  denoting  $As \rightarrow Gr$  with  $(As \wedge Gr) \rightarrow \gamma$ . We also call  $(As, Gr)_\gamma$  a safety requirement.*

The conditional  $As \rightarrow Gr$  requires that the system  $\mathcal{A}_S$  fulfils its safety-related guarantees whenever the environment  $\mathcal{A}_E$  fulfils its assumptions. Moreover, A/G pairs induce *probabilistic responsibility relationships* and enable underspecification (i.e.  $\kappa$ -completion) in  $\mathcal{A}_S$  to support, for example, degrees of freedom for development:

On the one hand, responsibility can restrict the environment, the system can rely on  $(\llbracket As \rrbracket \cap \llbracket \mathcal{M} \rrbracket)$ . By convention, the system is no more required to fulfil certain guarantees if the environment violates the associated assumptions. Any run, where a *defect*, obstructing a guarantee, occurs after an *unexpected stimulus*, violating the associated assumption, ceases to fall under the responsibility of the system.

On the other hand, underspecification can be hazardous if the responsibility to avoid unexpected stimuli for the system is imposed on the environment. Hence, in situations where the environment abdicates from its responsibility, the system can be obliged to provide safety measures. The responsibility to reduce hazardous reactions should then be taken by the system.

**Definition 4.13 (Practicable Behavioural Safety)** *Given that  $\Gamma$  only contains A/G-based safety goals according to Definition 4.12, we adapt Definition 4.10 to obtain*

$$\mathfrak{BS}_{\text{pract}}(\mathcal{S}) \triangleq \frac{\max\{|\mathcal{G}| \mid \mathcal{G} \subseteq \Gamma \wedge \mathcal{M} \models \bigwedge_{\gamma \in \mathcal{G}} (As, Gr)_\gamma\}}{|\Gamma|} \quad (4.7)$$

In spite of this definition, a practicable set of A/G pairs should impose safety measures in the system in as many as *reasonably practicable situations* where  $\mathcal{A}_E$  violates the assumptions. Then,  $\mathfrak{BS}_{\text{pract}}$  expresses that  $\mathcal{S}$  is *practicably safe* although  $\Gamma$ 's A/G pairs tolerate hazardous runs beyond these situations.

The *interest* to fulfil an assumption correlates with the *value* (e.g. safety measure) of the associated guarantee. We can strip down assumptions into conjuncts such that it is

#### 4. Behavioural Safety: Concepts

possible to handle easily violable conjuncts (i.e. too strong, cf. page 53) and to exclude heavily violable conjuncts (i.e. too weak) from *responsibility assessment*. This step can raise the possibility to err in estimating the *strength of assumptions* and their probability of being violated in  $\mathcal{W}$ . Moreover, one may come across *misperceptions of responsibility* where strong assumptions are ignored by human operators and weak assumptions imply the automation paradox (Section 1.1). Weakened assumptions require more extensive (also *conservative*) safety measures to be realised by the system than strong assumptions do. Supplementary, Example 4.4 applies Definition 4.12.

▮ **Example 4.4 (Negotiation of Responsibilities for a Car Airbag)** *Suppose that we have the safety goal  $\gamma_{\text{treat.distract}}$  from Example 4.3. We can derive the A/G pair*

$$\begin{aligned} (\text{As}, \text{Gr})_{\text{treat.distract}} &\equiv P_{<\chi}[\text{AsBody}] \rightarrow P_{\geq\gamma}[\text{GrBody}] && \text{with} \\ \text{As} &\equiv P_{<\chi}[\text{AsBody}] \equiv P_{<0.001}[\text{Fcrashed}] \\ \text{Gr} &\equiv P_{\geq\gamma}[\text{GrBody}] \equiv \gamma_{\text{treat.distract}} \end{aligned}$$

▮ *The formula AsBody shows that the essential responsibility to treat  $\phi_{\text{distract}}$  and  $\chi_{\text{unexpExp}}$  is left over to the system.*

In summary, the strengthening of guarantees (constraining the set of hazardous reactions), the weakening of assumptions (relaxing the set of specified stimuli), and the regard of unexpected behaviour (complementing the set of specified stimuli by unexpected stimuli and specifying safe reactions for the latter) are advisable to make  $S$  safe. The proposed method helps obtain a safe specification with respect to the A/G pairs. In particular, A/G pairs can express the *treatment of maloperation*. This way, practicable behavioural safety covers *safety by treatment of maloperation*.

### 4.6. Safety Measures

This section discusses how to achieve practicable behavioural safety. Given that  $S = (\mathcal{V}, \mathcal{M}, \Gamma)$ , we can distinguish two *strategies of hazard treatment*:

1. *Prevention* by avoiding causal factors (e.g. defects) and, thus, hazards.
2. *Mitigation* by handling tolerated causal factors through *preventive measures*, and by alleviating mishaps through *passive measures*.

▮ **Definition 4.14 (Treatment)** *Let  $S' = (\mathcal{V}', \mathcal{M}', \Gamma')$  be a specification. We call  $S \rightsquigarrow_{\text{T}} S'$  a treatment of a hazard  $\chi$  and its mishap  $\phi$  towards the A/G-based safety goal  $\gamma \in \Gamma \cup \Gamma'$  iff the transformation T (Section 2.2.1) establishes the condition*

$$\mathfrak{BS}_{\text{pract}}(S) < \mathfrak{BS}_{\text{pract}}(S') \wedge \mathcal{M} \not\models (\text{As}, \text{Gr})_{\gamma} \wedge \mathcal{M}' \models (\text{As}, \text{Gr})_{\gamma}$$

▮

This definition allows transformations according to the strategies to be applied to the *operational* (Section 4.6.1) and *descriptive* (Section 4.6.2) part of  $S$ .

### 4.6.1. Treatments for Mode Transition Systems

To follow the two strategies in  $\mathcal{M}$ , hazards can be treated in two ways:

1. Reduction of specification defects (Definition 4.4) such that  $\mathcal{M}|_{\text{use}} \models \widehat{\Gamma}'$ .
2. Compensation of operational defects ( $\mathbb{M}_{\text{fail}}$ , Definition 4.5) by constructing safety measures in  $\mathcal{A}_{\text{S}}|_{\text{save}}$  and assigning operator responsibilities to  $\mathcal{A}_{\text{E}}|_{\text{save}}$ .

The next two paragraphs introduce two basic techniques, called *completion* and *indeterminisation* of a transition system:

*Completion* aims at fully specified modes in  $\mathcal{M}$  to reduce hazardous  $\kappa$ -completion.  $\mathbb{M}_{\text{fail}}$  can add defective behaviour to  $\mathbb{M}_{\text{use}}$ .  $\mathbb{M}_{\text{save}}$  can add safe behaviour whenever  $\mathbb{M}_{\text{use}}$  allows  $\kappa$ -completion:  $\mathcal{A}_{\text{E}}|_{\text{save}}$  enhances  $\mathcal{A}_{\text{E}}|_{\text{use, fail}}$  and  $\mathcal{A}_{\text{S}}|_{\text{save}}$  enhances  $\mathcal{A}_{\text{S}}|_{\text{use, fail}}$ .  $\mathcal{M}|_{\text{use, save}}$  reduces defective system behaviour and permissive environment behaviour. For task T2 (see page 36), completion enables safety measures for unexpected stimuli.

*Indeterminisation* complements defective actions (i.e.  $\text{pre} \equiv \phi, \text{trig} \equiv \perp$ , permissive) by *safety actions* (i.e.  $\text{pre} \equiv \phi, \text{trig} \equiv \top$ , coercive). By Equation (4.1) on page 48, this indeterminacy is assumed to be resolved in  $\mathcal{W}'$  such that the actions in  $\mathbb{M}_{\text{save}}$  are chosen prior to those in  $\mathbb{M}_{\text{use}}$  or  $\mathbb{M}_{\text{fail}}$ .

**Patterns** In order to treat causal factors in both agents,  $\mathcal{M}$  can be enhanced by

1. *fail-safe control actions* in  $\mathcal{A}_{\text{S}}|_{\text{save}}$ , depending on system defects, and
2. *preventive or passive actions* in  $\mathbb{M}_{\text{save}}$ , independent of system defects.

Among the fail-safe control actions, we can distinguish *fail-operational control actions*, which activate alternative parts of  $\mathbb{M}_{\text{use}}$  to maintain functionality, from *fail-silent control actions*, which deactivate parts of  $\mathbb{M}_{\text{use}}$  to achieve stability<sup>6</sup> (Section 3.3). Preventive actions help treat reducible hazards regarding causal factors of the system and the environment at once; mitigation takes place *in the hazardous state*. Passive actions help treat irreducible hazards where the harm state can still be alleviated; mitigation takes place *in the harm state*.

*Completion* applies to both strategies, *indeterminisation* helps with the conception of preventive and passive safety measures. Table A.4 describes several MTS patterns, for example, *repair MTS* and *safety cover*:

*Repair MTS in  $\mathcal{A}_{\text{S}}|_{\text{save}}$* : To reduce random operational defects (Section 4.2.1), we can construct an MTS  $\mathcal{M}_{\text{repair}}^{\text{S}}$ .  $\mathcal{M}_{\text{repair}}^{\text{S}}$  has a repair mode<sup>7</sup>, idling the defective mode  $m_{\text{d}}$  of  $\mathcal{M}_{\text{def}}$  through an action  $\epsilon_{\text{f}}$ , and a control action  $\text{re}_{\text{finish}}$  to return  $\mathcal{M}_{\text{def}}$  to an operational mode  $m$ .  $\text{delay}$  can define the maximum time required to repair  $\mathcal{M}_{\text{def}}$ .

*Safety cover*: To reduce maloperation, a cover in  $\mathbb{M}_{\text{save}}$  modifies possible actions of  $\mathcal{A}_{\text{E}}$  such that the hazardous state is no more easily reachable.

<sup>6</sup>Hence, in a safe mode, the system can be defective.

<sup>7</sup>*Repair actions* belong to  $\mathbb{M}_{\text{save}}$ , ideally start from safe modes, and take place in  $\mathcal{A}_{\text{E}}$  and  $\mathcal{A}_{\text{S}}$ .

### 4.6.2. Treatments for Behavioural Property Assertions

This section explains the treatment of missing or defective *avoidance goals*—see Equation (4.3)—which are considered as hazardous specification defects.

A hazard  $\chi$  can be treated by *adding* or *revising* an *A/G-based avoidance goal*  $\gamma$  (Figure 4.6a): If  $\gamma \notin \Gamma$  ( $\chi$  motivates  $\gamma$ ), we can extend  $\Gamma$  by adding  $\gamma$ . If  $\gamma \in \Gamma$  ( $\chi$  obstructs  $\gamma$ ), we can revise a violable assumption or guarantee by an alternative decomposition of  $\gamma$ . In both cases, we are required to make the assumption and the guarantee explicit. In order to revise a safety goal, we can

1. identify unexpected and hazardous behaviour
2. weaken a violable assumption and
3. strengthen a low-grade guarantee.

If we can not rely on the environment to fulfil the assumption, the system may take *responsibility*. The treatment of property defects can require changes in  $\mathcal{A}_S$  and  $\mathcal{A}_E$ .

**Pattern** The *completion* pattern helps treat specification defects such as violable assumptions and misperceptions of responsibility (cf. page 53): Given a hazard  $\chi$ , a safety goal  $\gamma$  and an A/G pair  $(\mathcal{A}_S, \text{Gr})_\gamma$ , Table 4.3 depicts property defects and treatments in  $\Gamma$ . The treatments consist of identifying an assumption, being too strong, or a guarantee, being too weak, by characterising hazardous unexpected stimuli (Ex) and defective reactions (Df). This assessment leads to the derivation of preventive weakened assumptions  $\underline{\mathcal{A}}_S$ , and of preventive strengthened guarantees  $\underline{\text{Gr}}$  including a descriptively specified safety measure  $\text{Gr}'$ . Suppose the following axioms hold

- for definition:  $\underline{\mathcal{A}}_S \equiv \mathcal{A}_S \vee \text{Ex}$ ,  $\underline{\text{Gr}} \equiv \text{Gr} \wedge \text{Gr}'$
- as assumptions:  $\neg(\text{Df} \wedge \text{Gr})$ ,  $\text{Gr}' \rightarrow \neg\chi$ ,  $\neg(\text{Df} \wedge \text{Gr}')$ ,  $\neg(\text{Ex} \wedge \mathcal{A}_S)$ ,  $\neg(\gamma \wedge \chi)$ ,  $((\text{Gr} \vee \text{Df}) \wedge \chi) \not\vdash \perp$

**Corollary 4.3** *The treatment for case 2 in Table 4.3 captures the most general of these cases and yields a revised A/G pair  $(\underline{\mathcal{A}}_S, \underline{\text{Gr}})_\gamma$ .*

By the formula  $\underline{\mathcal{A}}_S \rightarrow \underline{\text{Gr}} \wedge (\underline{\mathcal{A}}_S \wedge \underline{\text{Gr}}) \rightarrow \gamma$ , the safety goal  $\gamma$  can now be alternatively imposed on  $\mathcal{M}$ . From the axioms, we derive  $\mathcal{A}_S \rightarrow \underline{\mathcal{A}}_S$  and  $\underline{\text{Gr}} \rightarrow (\text{Gr} \wedge \neg\chi)$  for compatibility with the original safety goal in the cases 1 and 2 of Table 4.3. Given that  $\mathcal{V} = \mathcal{V}_{\text{specified}}$  and  $\mathcal{V}' = \mathcal{V}_{\text{safe}}$ , Figure 4.6b illustrates the revision of A/G-based safety goals based on  $\llbracket \mathcal{V} \cup \mathcal{V}' \rrbracket$ .

*Completion* helps treat property defects such as independence, domain maintenance and run conformance assumptions<sup>8</sup> easily violable by  $\mathcal{A}_E$ : For *violable independence assumptions*, the treatments for the cases 1 and 4 can introduce variables in  $\mathcal{V}'$  which may still be disregarded in  $\mathcal{M}$ . The treatments require to extend  $\mathcal{S}$  via  $\mathcal{V} \rightsquigarrow_{\top} \mathcal{V}'$  and

<sup>8</sup>See page 53. As described in Appendix A.8, such assumptions disclose an invalid system boundary (Tables A.18b and A.19b), invalid ranges (Table A.19a) or invalid behaviour of  $\mathcal{A}_S$ .



Case (with respect to $\chi$ and $\mathcal{M}$ )	Assessment Result (w.r.t. $\chi$ )	Treatment (i.a. preventive)
0. $\forall \rho \in [\mathcal{M}] :$ $\rho \models \text{As} \rightarrow (\text{Gr} \wedge \neg \chi)$	$\mathcal{M}$ as specified, Gr safe, no decision on As	none
1. $\exists \rho \in [\mathcal{M}] :$ $\rho \models \text{As} \rightarrow (\text{Gr} \wedge \chi)$	$\mathcal{M}$ generates hazardous reactions, Gr too weak, no decision on As	$\text{As} \rightarrow \underline{\text{Gr}}$ , safe strengthened guarantee
2. $\exists \rho \in [\mathcal{M}] :$ $\rho \models (\neg \text{As} \wedge \text{Ex}) \rightarrow (\text{Gr} \wedge \chi)$	$\mathcal{M}$ generates hazardous reactions for unexpected stimuli, As too strong and hazardous, Gr hazardous	$\underline{\text{As}} \rightarrow \underline{\text{Gr}}$ , weakened assumption and safe strengthened guarantee
3. $\exists \rho \in [\mathcal{M}] :$ $\rho \models (\neg \text{As} \wedge \text{Ex}) \rightarrow (\neg \text{Gr} \wedge \chi)$	$\mathcal{M}$ generates hazardous reactions for unexpected stimuli, As as in case 2, no decision on Gr, Df negligible	$\underline{\text{As}} \rightarrow \text{Gr}'$ , weakened assumption and safe but deviating guarantee
4. $\exists \rho \in [\mathcal{M}] :$ $\rho \models \text{As} \rightarrow (\neg \text{Gr} \wedge \chi \wedge \text{Df})$	$\mathcal{M}$ generates hazardous defective reactions, no decision on As and Gr	$\text{As} \rightarrow \text{Gr}'$ , safe but deviating guarantee

Table 4.3.: Property defects for hazard  $\chi$  requiring revision of an A/G-based safety goal  $\gamma$ 

to complete  $\mathcal{S}$  using variables in  $\mathcal{V}' \setminus \mathcal{V}$ . To reduce the change impact on  $\mathcal{M}$ , one can instead *indeterminise*<sup>9</sup>  $\mathcal{A}_S$  by  $\mathbb{M}_{\text{save}}$ . The treatments for the cases 1 and 4 handle assumptions or guarantees not even to be obstructed or violated to be hazardous. The treatments for the cases 2 and 3 also apply for *violable domain maintenance* and *run conformance assumptions*.

Table 4.3 shows a perspective where the system takes responsibility. Treatment can be different if the *responsibility is shifted* to the environment: For the cases 1 and 4, we can *strengthen the assumptions*. For the cases 2 and 3, we can *ignore* the handling of unexpected stimuli. Such a shift, however, may raise *misperceptions of responsibility*. Finally, Example 4.5 shows how MTS patterns improve a specification.

<sup>9</sup>See Section 4.6.1. As described in Appendix A.8 and Table A.18c, each realisation of  $\mathcal{A}'_S$  resolves this indeterminacy by taking the right choices according to the given priorities and probabilities.

#### 4. Behavioural Safety: Concepts

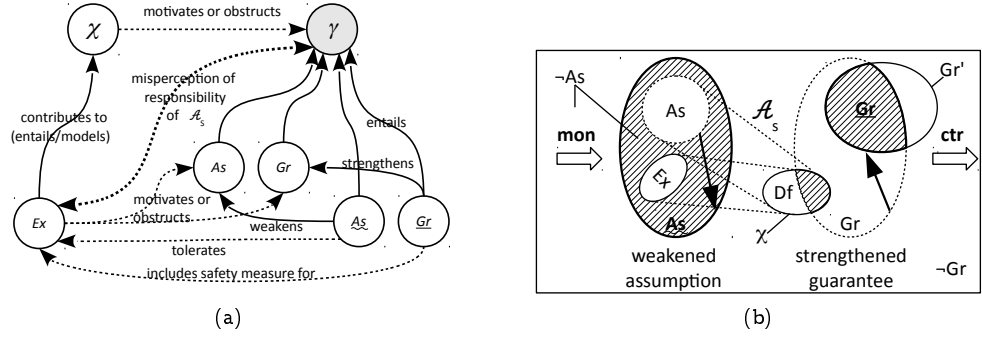
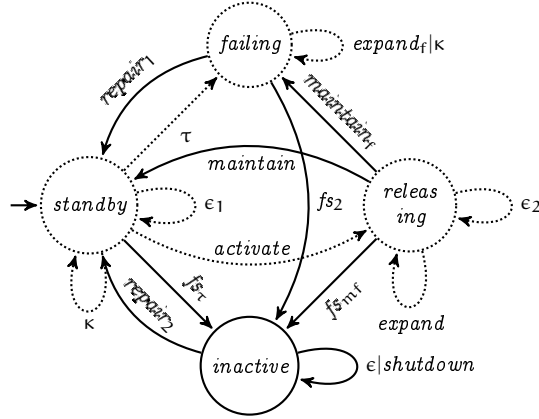


Figure 4.6.: Revision of a safety goal  $\gamma$ : (a) logical relationships, (b) sets of runs

Example 4.5 (Safety Measures for a Car Airbag) The safety goal  $\gamma_{treat.distract}$  of Example 4.3 constrains the instantiation of MTS patterns for the treatment of the referenced hazards. Example 2.1 applies the pattern “repair action” where  $re$  instantiates to  $repair_1$ . In addition, we now use the pattern “fail-safe actions/fail-silent” twice (i.e. for the defective control actions  $\tau$  and  $maintain_f$ ;  $re$  instantiates to  $repair_2$ ).



Remarks: To reify active and inactive physical states of the airbag mechanism, we introduced the variable  $active : \mathbb{B}$ . The action  $fs_2$  does only compensate  $\kappa$ -completion but leaves a 1% probability for  $expand_f$ .  $pre_{re}$  is left undefined as  $A_E$  is implicit. Practicable behavioural safety ( $\mathcal{B}\mathcal{E}_{pract}(S) = 1$ ) of the airbag specification requires the successful check of  $\Gamma = \{\gamma_{treat.distract}\}$ .

Safety Action Specifications of the MTS Airbag<sub>S</sub>  $\oplus$  Airbag<sub>S<sub>s</sub></sub> (only actions in solid lines)

Label	pre	delay	trig	prio	$\pi$	post
$fs_\tau$	$\top$	0	$\perp$	1	1	NOP
$maintain$	$gas \neq full$	10	$\perp$	1.5	.95	$gas := full$
$maintain_f$	$gas \neq full$	10	$\perp$	1.5	.01	$gas := defective$
$fs_{mf}$	$gas \neq full$	10	$\perp$	1.5	.04	NOP
$fs_2$	$\top$	0	$\top$	1.5	.99	NOP
$repair_1$	$\top$	0	$\perp$	1	1	$gas := full, released := no$
$\epsilon$	$\neg active$	0	$\top$	1	1	NOP
$shutdown$	$active$	0	$\top$	1	1	$active := \perp$
$repair_2$	$pre_{re}$	0	$\perp$	1	1	$gas := full, released := no, active := \top$

#### 4.7. A Stop Criterion for Safety-oriented Validation

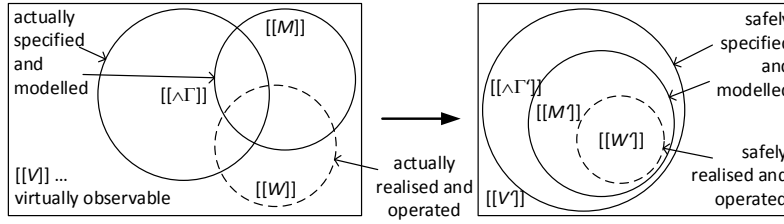


Figure 4.7.: Consistency of  $\Gamma$ ,  $\mathcal{M}$  and  $\mathcal{W}$  after a chain of treatments  $\mathcal{S} \rightsquigarrow_* \mathcal{S}'$

### 4.7. A Stop Criterion for Safety-oriented Validation

Based on the set  $\mathcal{H}$  (Definition 4.8),  $\mathcal{M} \not\models \widehat{\Gamma}'$  indicates that  $\mathcal{S}$  is hazardous ( $\mathcal{S} \neq \mathcal{S}'$ ); the more that has to be done to identify hazards, the stronger it can be stated that  $\mathcal{S}$  is safe ( $\mathcal{S} = \mathcal{S}'$ ). The successive improvement of behavioural safety can be regarded as a chain of treatments  $\mathcal{S} \rightsquigarrow_* \mathcal{S}'$  (Figure 4.7). The enhancement of  $\Gamma$  and  $\mathcal{M}$  towards  $\mathcal{S}'$  with  $\mathcal{M}' \models \widehat{\Gamma}'$  can suffice to declare  $\mathcal{S} = \mathcal{S}'$ . Note that defect treatment by  $\mathbb{M}_{\text{save}}$  leaves a residual defect model in  $\mathcal{S}'$ . In summary, the Definitions 4.10 and 4.13 yield a stop criterion for treatments (Definition 4.14) in an iterative procedure.

### 4.8. Notes and Further Reading

Control loops or plant settings similar to Definition 4.1 and Figure 4.1 can be found in the literature.<sup>10</sup> Operational situations have been modelled in Dobi et al. (2013).

The criteria for the defect taxonomy are motivated by IEC Std. 61508 (2011), ISO Std. 26262 (2011), Börcsök (2011), Neumann (1995) and Gleirscher (2011, 2013b): Börcsök investigates criterion 2a using the term *recurrence*. Parnas et al. (1990) and Snooke and Price (2011) support the treatment of software failures as random failures unless their causative faults are detected. For the criteria 3 and 4, dependent defects are discussed by Börcsök, and single and multiple point failures in ISO Std. 26262 (2011). For criterion 5, Damm and Peikenkamp (2004) discriminate *systemic* and *physical failures*. For criterion 6, Pock (2012) considers *spurious trips* (unacceptable execution or commission of system actions) and *failures on demand* (unacceptable suppression or omission of system actions; see also McDermid 2002). For diagnosis at runtime, Lunze (2010) regards errors in measurement and state observation. As human operators are often responsible for recognising hazardous events and reacting with safe control, Leveson et al. (1997) study the detection of *mode confusions*<sup>11</sup> making operators prone to *maloperation*; see Example 4.6. Maloperation is often not reducible to a *misperception of responsibility* whereas a lack of safety measures

<sup>10</sup>See, e.g. Feather 1987, Henzinger 2000, Jürgensohn 2007, Leveson 1995, 2012, Pyle 1991, Secchi et al. 2007.

<sup>11</sup>Mode confusions can be seen as an appearance of *adaptivity* as discussed by Broy et al. (2009).

#### 4. Behavioural Safety: Concepts

usually is.

- ⌈ **Example 4.6 (Maloperation of Electric Light)** *The observation of “light is off” could make a human assume, reason, act and err twice: (i) “no electricity  $\rightarrow$  light in deactivated mode*  
 ⌋  *$\rightarrow$  operate switch” or (ii) “broken bulb  $\rightarrow$  light in defective mode  $\rightarrow$  exchange bulb.”*

State guide words were inspired by Luksch (2012), mode, action and event guide words by HAZOP (Section 3.2) and Stålhane et al. (2012). In addition to guide words, Rasmussen (1997) describes categories of hazardous elements and causal factors. My notion of hazard goes along the lines of Pyle, Rasmussen, Leveson (1995) and Ericson (2005). Definition 4.8, however, accepts safety reasoning to be carried through inductively or deductively. Moreover, it does not entail  $\chi \rightarrow \phi \vee \chi \leftarrow \phi$  because  $\phi$ , by further abduction, may be caused by other hazards and  $\chi$  does not necessarily have to lead to  $\phi$ . The calculation of hazard characteristics (see page 56) from MARKOV chains is discussed, for example, in (Böröcsök 2011, Section 13.2).

*Safety properties* are used in formal approaches to testing and reactivity.<sup>12</sup> Although *avoidance goals* (Definition 4.9) are closest to Lampert’s notion (cf. Section 3.1), the present approach takes a probabilistic view and considers *mitigation goals* with a liveness conjunct as well as *alleviation goals*. As emphasised in the Sections 1.1 and 4.4, such goals characterise safety as the *degree of freedom from hazards* and mishaps. In contrast, functional safety is concerned about the reliable design and realisation of electronic and software-intensive safety measures as opposed to the identification of such measures and the specification of their behaviour. In summary,  $\hat{\Gamma}$  can be seen as a variant of Pyle’s SAFE(U, V, W) predicate (Pyle 1991).

The Definitions 4.12 and 4.13 were motivated by Broy (1998) who applies A/G style to assert relationships among behaviours; obstructions of and inconsistencies among property assertions are investigated by van Lamsweerde (2009). Probabilistic computation tree logic has been applied in safety analysis of technical systems by Johnson (1993); probabilistic reachability analysis is explored, for example, by Forejt et al. (2011) and Baier and Katoen (2008).

The *realisation of MTS patterns* (Section 4.6.1) by safety-related subsystems depends on the kind of defect. This realisation can require *state observation or runtime diagnosis* of behavioural properties (Definition 2.15) such as

- system defects, for the achievement of fault tolerance of the system, or
- environment defects, for preventive or passive measures in  $\mathcal{A}_S|_{\text{safe}}$ .

To such patterns, Martinus (2004) and Gärtner (1999) show techniques<sup>13</sup> and mechanisms for fault tolerance. Moreover, fault indicators (see page 51) abstract from state observation, fault detection and diagnosis (Section 2.2.3). Such variables help specify safety measures, resolve indeterminacy and reify priorities among actions. As an abstraction from constructive measures (Figure 3.2a,b),  $\mathcal{A}_S|_{\text{save}}$  can be realised by any

<sup>12</sup>See, e.g. Broy and Stølen 2001, D’Ippolito et al. 2011, Peled et al. 1999, Peleska 1996, Rusu et al. 2005.

<sup>13</sup>Several techniques can be required to make  $\mathbb{M}_{\text{use}}$  safe and compensate  $\mathbb{M}_{\text{fail}}$ .

technology, for example, by software. Measures for task T2 (see page 36) can incorporate an environment model; measures in  $\mathcal{A}_{E|_{\text{safe}}}$  usually consist of training, guidance (e.g. warning signs and signals) and protection mechanisms as in  $\mathcal{A}_{S|_{\text{safe}}}$ . Luksch (2012) describes safety directives to be followed by human operators to justify strong assumptions. Pyle (1991) explains two kinds of functional measures, called *interlocks* and *guards*, to be implemented by computer-based safety-related subsystems.

Broy and Stølen (2001) define a basis for *refinement* calculi to be applied to verify the application of transition system and property treatments. For Section 4.6.2, van Lamsweerde (2009) discusses property patterns for goal refactoring and refinement, for example, refinement due to lack of control or monitoring, alternative refinement, treatment of goal conflicts and obstacles, and optimisation.

Suppose that we have a refinement relation  $\rightsquigarrow \subseteq \mathbb{M} \times \mathbb{M}$ . We write  $\mathcal{M} \rightsquigarrow \mathcal{M}'$  (spoken “ $\mathcal{M}$  is refined to  $\mathcal{M}'$ ”) to denote  $\llbracket \mathcal{M} \rrbracket \supseteq \llbracket \mathcal{M}' \rrbracket$ . A defect model can be obtained, for example, through refinement of an underspecified (i.e. either incomplete or non-deterministic)  $\mathcal{M}$ , or through indeterminisation of  $\mathcal{M}$ . In the latter case,  $\mathcal{M}|_{\text{use},\text{fail}}$  might be no refinement of  $\mathcal{M}|_{\text{use}}$  because defective actions can generate runs deviating from the set of specified runs  $\llbracket \mathcal{M}|_{\text{use}} \rrbracket$  (cf. Definition 4.6). Indeterminisation (Section 4.6.1) can violate refinement according to Broy and Stølen even if we assume safety actions to be performed prior to hazardous actions. Nevertheless, if we can expect that  $\mathcal{M}|_{\text{use}} \rightsquigarrow \mathcal{M}'|_{\text{use}}$  and  $\mathcal{M}|_{\text{use}} \rightsquigarrow \mathcal{M}'|_{\text{use},\text{fail},\text{save}} = \mathcal{M}'$ , refinement verification should take place to show these two relationships.

A validated specification  $S'$  can be expected to be a conditional refinement of  $S$ , given that no defective or hazardous action is chosen in any realisation of  $S'$ . In other words, any hazardous but correct action possible in  $\mathcal{W}$  might change or even cease in being performed in  $\mathcal{W}'$ . Hence, *hazard treatments* cannot generally be considered as refinements of a system’s functionality that maintain correctness (see also Gärtner 1999). In summary, Definition 4.14 requires a transformation to maintain or increase the degree of behavioural safety.

## Behavioural Safety: Procedure

This chapter describes a possible procedure for using the presented concepts. Figure 5.1 gives an overview, and Figure 5.2 at the end of this chapter depicts the stages, steps and sub-steps of the procedure including the elaborated artefacts.

### Contents

5.1	Modelling Stage: Understand System	68
5.1.1	Step 1: Model System and Environment	68
5.1.2	Step 2: Derive Defect Model	70
5.2	Analysis Stage: Identify Hazards	71
5.2.1	Step 3: Identify Potential Mishaps	71
5.2.2	Step 4: Assess Causal Factors	71
5.3	Assurance Stage: Improve System Functionality	72
5.3.1	Step 5: Specify Safety Goals	72
5.3.2	Step 6: Plan and Design Safety Measures	72
5.4	Notes and Further Reading	73

## 5.1. Modelling Stage: Understand System

In this stage, Step 1 sketches<sup>1</sup> how to build  $\mathcal{M}|_{\text{use}}$ , and Step 2 focuses the derivation of defects from  $\mathbb{M}_{\text{use}}$  and their modelling in  $\mathbb{M}_{\text{fail}}$ .

### 5.1.1. Step 1: Model System and Environment

*Prerequisite:* Start with a set  $\mathcal{R}$  of property assertions ( $\mathcal{R}_{\text{p}}$ ) and functional requirements ( $\mathcal{R}_{\text{f}}$ ). *Task:* Set up  $\mathcal{M}|_{\text{use}}$  and  $\Gamma$  by performing requirements elicitation, analysis and specification (Section 2.1). *Invariant:* Maintain  $\mathcal{M}|_{\text{use}} \models \widehat{\Gamma}$ .

<sup>1</sup>Methods applied for system specification (Section 2.2.2) are no primary subject of investigation.

## 5.1. Modelling Stage: Understand System

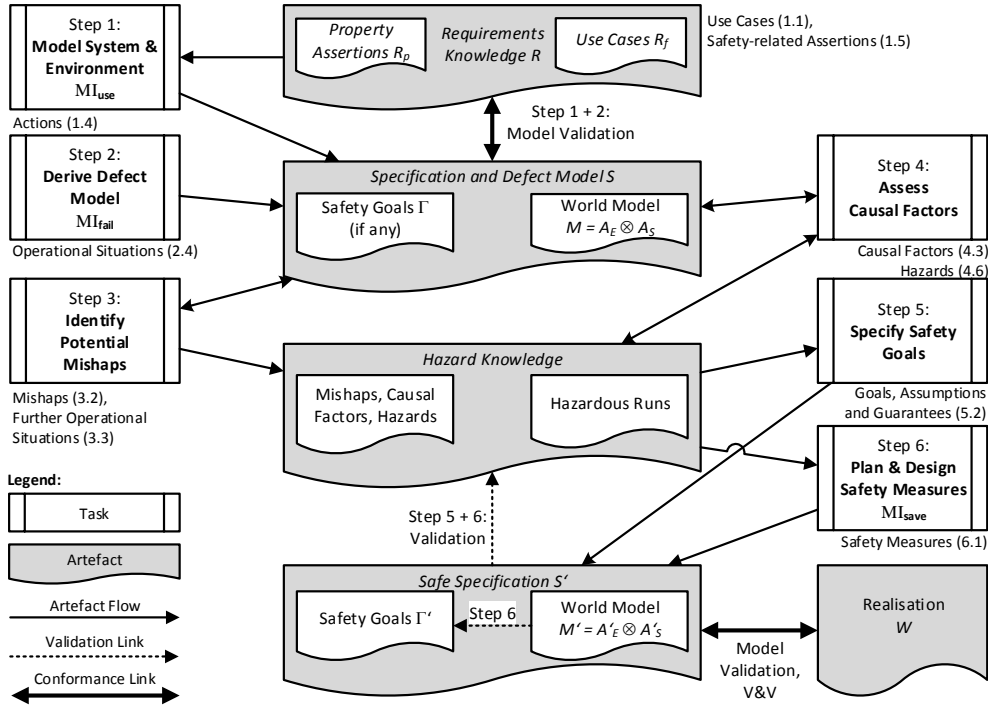


Figure 5.1.: Overview of the procedure

- 1.1. Transform  $\mathcal{R}_f$  into use cases.
- 1.2. To model *interaction* via the safety-related interface, determine  $\mathcal{V}$  from preliminary physical interface designs and from domain analysis. Take note of domain properties. Here and in the following steps, take care that the abstraction in  $\mathcal{M}$  stays efficiently high<sup>2</sup> but reasonably precise for validation.
- 1.3. Derive *tactics* of  $A_E|_{use}$  and *functions* of  $A_S|_{use}$  from the use cases (Step 1.1) and from  $\mathcal{R}_p$ . According to Definition 4.1, arrange functions and tactics in two hierarchies by extracting their *independent, concurrent and common* parts at an appropriate<sup>2</sup> level of abstraction.
- 1.4. Going top-down the two hierarchies, specify *abstract modes and actions* (Definitions 2.11 and 2.12) of the system and the environment at selective levels of abstraction. Going bottom-up the hierarchies, model the *behaviour* of both agents: for each MTS at the lowest level and for each abstract MTS, identify *actions* and their *repetition and order* by the help of *modes*, again at an ap-

<sup>2</sup>Finding this level requires experience in domain modelling, expert knowledge and should be governed by usefulness for safety analysis as well as size and complexity of the resulting MTS.

## 5. Behavioural Safety: Procedure

appropriate level of abstraction.<sup>2</sup> Capture *states* by using and extending  $\mathcal{V}$  and its types.

- 1.5. You may start with knowledge of *safety goals* (Definition 4.9). If applicable, classify the property assertions in  $\mathcal{R}_p$  into (i) safety-related assertions and (ii) other assertions. Formalise class (i) to initialise  $\Gamma$ .

Repeat the Steps 1.1 to 1.4 to achieve reasonably complete hierarchies. *Postcondition:*  $\mathcal{M}|_{\text{use}}$  contains all functionality to be analysed. The procedure allows including results from earlier iterations (e.g. safety measures) into  $\mathcal{M}|_{\text{use}}$ .

### 5.1.2. Step 2: Derive Defect Model

*Task:* Model operational defects (Definition 4.5, Section 4.2.2). Based on  $\mathcal{M}|_{\text{use}}$ , guess realistic (i.e. design- or realisation-based) defects and set up  $\mathbb{M}_{\text{fail}}$  using the following techniques:

- 2.1. Extend  $\mathcal{V}$  by *fault indicators* (e.g. functional and mode channels) to capture physical *side effects* in  $\mathbb{M}_{\text{use}}$ . Modify actions and MTSs with the help of Table A.2. Such knowledge may also be derived from Step 3.2.
- 2.2. Capture defective actions and modes by abstraction from, for example,
  - a) *reliability analysis*<sup>3</sup> of an architecture design
  - b) *failed test runs* during system testing or
  - c) *hazardous specification defects* known from Step 4 of previous iterations.

Use Table A.2 to derive *fragments* in  $\mathbb{M}_{\text{fail}}$  for a), b) and c).

- 2.3. Identify *operational situations* (Definition 4.2) to describe *initial states* to be manipulated through *fault injection* and to be used in Step 4. These situations can be derived from use cases (i.e. their preconditions and goals) in Step 1.1 or from Step 3.3 of previous iterations.

- 2.4. Introduce *fragments* by *mutation* of  $\mathcal{M}$  according to Table A.2:

- a) *Identify and exploit underspecification* of  $\mathcal{M}|_{\text{use, fail}}$  by identifying lack of domain coverage and weakening assumptions (Step 4.4) on monitored variables. Transform this *incompleteness* being subject of  $\kappa$ -completion into defective actions.
- b) *Add indeterminacy*<sup>4</sup>, for example, *stimuli-driven* from disregarded channels and *fault-driven* from disregarded self-managed variables (Step 2.1).
- c) *Document dependencies* by adding mode channels (Step 2.1).

*Postcondition:* Various operational defects are modelled as fragments in  $\mathbb{M}_{\text{fail}}$ .

<sup>3</sup>Techniques such as FMEA and FTA for answering these questions are discussed in Section 3.2.

<sup>4</sup>Defective actions overrule specified actions because of the assumed priorities, cf. Equation (4.1).



## 5.2. Analysis Stage: Identify Hazards

This stage guides the reasoning about *hazards*. Step 3 helps study *mishaps* and Step 4 their *causal factors*.

### 5.2.1. Step 3: Identify Potential Mishaps

*Task:* Identify state constraints to specify mishaps. For each action  $a$  (Step 1.4) descending top-down the system's hierarchy (Step 1.3):

- 3.1. Determine whether  $a$  affects channels defined in the Steps 1.2 and 2.1.
- 3.2. Specify *mishaps*  $\phi$  using *state guide words* from Table A.3 based on the channels affected by  $a$ . Where needed, extend  $\mathcal{V}$  and its types to reify the *harm state*. Return to Step 2.1 to capture side effects identified here.
- 3.3. Select and refine *operational situations* for each  $\phi$ . As in Step 2.3, previous iterations and preconditions of use cases (Step 1.1) may help.

*Postcondition:* Mishaps are described to define the harm state.

### 5.2.2. Step 4: Assess Causal Factors

*Assumption:* Runs leading to mishaps contain some information about causal factors.

*Task:* Identify causal factors based on  $\mathcal{M}|_{\text{use, fail}}$ . Based on Section 2.3.2, refine and combine these factors to form hazards. Find out how, when and why a hazard can occur (e.g. because an assumption about  $\mathcal{A}_E$  is violated). For each pair of mishap  $\phi$  (Step 3.2) and operational situation  $\sigma_\alpha$  (Step 3.3):

- 4.1. Set  $\Sigma_0 = \{\sigma \mid \sigma_\alpha(\sigma)\}$ . From any computation tree CT (Definition 2.2) returned from successfully checking  $\mathcal{M}|_{\text{use, fail}} \models \mathbf{EF}\phi$ , determine a set  $\text{CT}' \subset \text{CT}$  of *shortest and most probable* runs satisfying  $\mathbf{F}\phi$ .
- 4.2. In each run  $\rho \in \text{CT}'$ , check for *causal factors*: top-down the hierarchies (Step 1.3), apply *guide words* from Table A.1 to search for *causal events*  $e$  in  $\rho|_{\mathcal{V}_e}$ , *modes*  $m$ , *actions*  $a$  and *complex actions* (Definition 2.10) observable in  $\rho$ .
- 4.3. To *filter and refine* these search results, use further guide words.
- 4.4. *Classify* each *causal factor* by checking, according to Figure 4.4, whether it
  - obstructs assumptions or guarantees in  $\Gamma$  (cf. case distinction in Table 4.3). Then, put it into the set  $\mathcal{H}_{\text{kn}} \cup \mathcal{H}_{\text{op}}$  of *hazardous knowable defects*. Reduce the causal factor to an environment or system defect (Section 4.2.1).
  - makes use of  $\mathbb{M}_{\text{fail}}$  (Step 2) or  $\kappa$ -completion. Then, put it into the set  $\mathcal{H}_{\text{op}}$  of *hazardous operational defects*. The defect taxonomy in Section 4.2.1 allows further classification.

## 5. Behavioural Safety: Procedure

- neither obstructs assumptions or guarantees in  $\Gamma$  nor uses  $\mathbb{M}_{\text{fail}}$ . Then, put it into the set  $\mathcal{H}_{\text{un}}$  of *hazardous unknown specification defects*.

4.5. *Specify hazards*: combine causal factors in a past formula<sup>5</sup>  $\chi$  separating *hazardous* from safe performance of  $a$ , activation of  $m$  or occurrence of  $e$ . Where needed, extend  $\mathcal{V}$  and its types to reify the *hazardous state*.

4.6. *Prioritise* the hazard  $\chi$  by *quantification* according to page 56.

*Postcondition*: Causal factors are known and combined to obtain the set  $\mathcal{H}$  of hazards.

## 5.3. Assurance Stage: Improve System Functionality

In this stage, Step 5 and Step 6 guide through the transformation  $S \rightsquigarrow_* S'$  to treat hazards known from the analysis stage.

### 5.3.1. Step 5: Specify Safety Goals

*Task*: Perform a treatment  $\Gamma \rightsquigarrow_* \Gamma'$  (Section 4.6.2) by transforming  $\mathcal{H}$  and  $\Phi$  into a *specification of safe behaviour*. For each hazard  $\chi \in \mathcal{H}$  and each mishap  $\phi \in \Phi$ :

- 5.1. Based on Definition 4.9, transform  $\chi$  or  $\phi$  or both into a *safety goal*  $\gamma$ . This step integrates the results of the Steps 1.5, 3.2 and 4.5.
- 5.2. *Clarify responsibilities* for  $\gamma$ : Derive an A/G pair<sup>6</sup>  $(As, Gr)_\gamma$  according to Definition 4.12. *Assign* the resulting *assumption* and *guarantee* to the tactics and functions related to each action  $a$  and event  $e$  as referenced by  $\chi$  (Steps 4.2 and 4.3). Where needed, extend  $\mathcal{V}$  and its types to reify the *safe state*.
- 5.3. Based on Step 4.6, annotate  $\gamma$  with the risk at which it is allowed to be violated.
- 5.4. By *proving*  $\widehat{\Gamma} \not\vdash \perp$ , assure that the safety goals are consistent. The *check* of Step 4.4 and  $\mathcal{M} \models (\widehat{\Gamma} \wedge \chi)$  determines whether  $\chi$  requires existing safety goals to be *revised*. The same has to be done at the level of A/G pairs.

*Postcondition*: An A/G-based safety goal for each hazard and mishap is specified.

### 5.3.2. Step 6: Plan and Design Safety Measures

*Prerequisite*: To simplify the procedure,  $\mathbb{M}_{\text{fail}}$  and  $\mathbb{M}_{\text{save}}$  have to be empty at the start of an iteration. Hence, transfer an  $\mathbb{M}_{\text{save}}$ , derived in a previous iteration, to  $\mathbb{M}_{\text{use}}$  and remove or hide the part<sup>7</sup> of  $\mathbb{M}_{\text{fail}}$  treated by  $\mathbb{M}_{\text{save}}$ . *Task*: Perform a *treatment*

<sup>5</sup>Such an expression can refer to (abstract) states (events, modes) and actions missing, being active and performed by *functions* and *tactics* in a hazardous combination.

<sup>6</sup>Based on Step 4.6, this step identifies *reasonably practicable situations* as discussed in Section 4.5.

<sup>7</sup>This part is compensated by existing safety measures formerly in  $\mathbb{M}_{\text{save}}$  and now in  $\mathbb{M}_{\text{use}}$ . Verification and architecture analysis (Step 2.2) assures that this compensation is realised correctly.

$\mathcal{M} \rightsquigarrow_* \mathcal{M}'$  to increase behavioural safety (Section 4.4). For each A/G-based safety goal  $\gamma \in \Gamma$ , take measures (i.e. functions, tactics, actions and modes):

- 6.1. Use  $\gamma$  and results from Step 4.2 to modify  $\mathbb{M}_{\text{use}}$  and compensate  $\mathbb{M}_{\text{fail}}$  by  $\mathbb{M}_{\text{save}}$  using the patterns from Table A.4. Take care to identify and update MTSs related (i.a. via mode channels) to MTSs affected by superimposition of  $\mathbb{M}_{\text{save}}$ .
- 6.2. If Step 6.1 results in infeasible transformations, choose an alternative way of resolving responsibility: repeat Step 5.2, identify alternative decompositions<sup>8</sup>, replace defective A/G pairs using Table 4.3, and proceed with Step 6.1.
- 6.3. By Definition 4.13, *estimate* the degree  $\mathfrak{BS}_{\text{pract}}(\mathcal{S})$  of behavioural safety. Given a set of initial states or an operational situation: Do the safety measures bring  $\mathcal{M}$  back to the *safe state*? Is the safe state *stable*, do the safe modes of  $\mathcal{M}$  assure a stable safe state? Given modularity of composition and behavioural refinement: does a safety measure perform as *independent* as required by  $\Gamma$ ? Identify *dead modes* by checking for *unproductive* behaviour fulfilling  $\text{EFG}\epsilon.\text{post}$ . Identify *underspecification* by checking for *chaotic* behaviour fulfilling  $\text{EGF}\kappa.\text{post}$ . Unless  $\mathfrak{BS}_{\text{pract}} = 1$ , return to Step 4.1.

*Postcondition:* The specified safety measures result in  $\mathfrak{BS}_{\text{pract}} = 1$  and  $\mathcal{S}$  can be declared to be  $\mathcal{S}'$ .

## 5.4. Notes and Further Reading

For Step 1, Broy (2005, 2010) and Schätz (2008) describe ways to build a specification based on a behavioural model. Through the Steps 1 and 5,  $\mathcal{R}_f$  and  $\mathcal{R}_p$  can become aligned by using  $\mathcal{V}$  in  $\mathcal{M}$  and  $\Gamma$ .

Supporting Step 4.4, Section 4.5 investigates the *violation* of an A/G-based safety goal  $\gamma$  by  $\mathcal{M}|_{\text{use, fail}}$ , and Section 4.6.2 discusses the *obstruction* of  $\gamma$  by a hazard  $\chi$ . For the Steps 5.1 and 5.2, Dwyer et al. (1999) and Dobi et al. (2013) discuss how to derive behavioural property assertions. Step 5.3 corresponds to determining *integrity classes* (cf. page 36) for safety goals. These classes can represent reliability requirements for safety measures. Some of the patterns to be applied in Step 6.1 define such requirements for  $\mathcal{A}_S|_{\text{save}}$ .

For Step 6.3, Schulz and Peleska (2010) apply probabilistic reachability analysis for train control systems: within the *safe state*, these systems are often required to avoid any unnecessary activation of a *stable mode*.

---

<sup>8</sup>Determine, which parts of  $\mathbb{M}_{\text{use}}$  and which uncompensated parts of  $\mathbb{M}_{\text{fail}}$  can be addressed by the system, the environment or both agents.

## 5. Behavioural Safety: Procedure

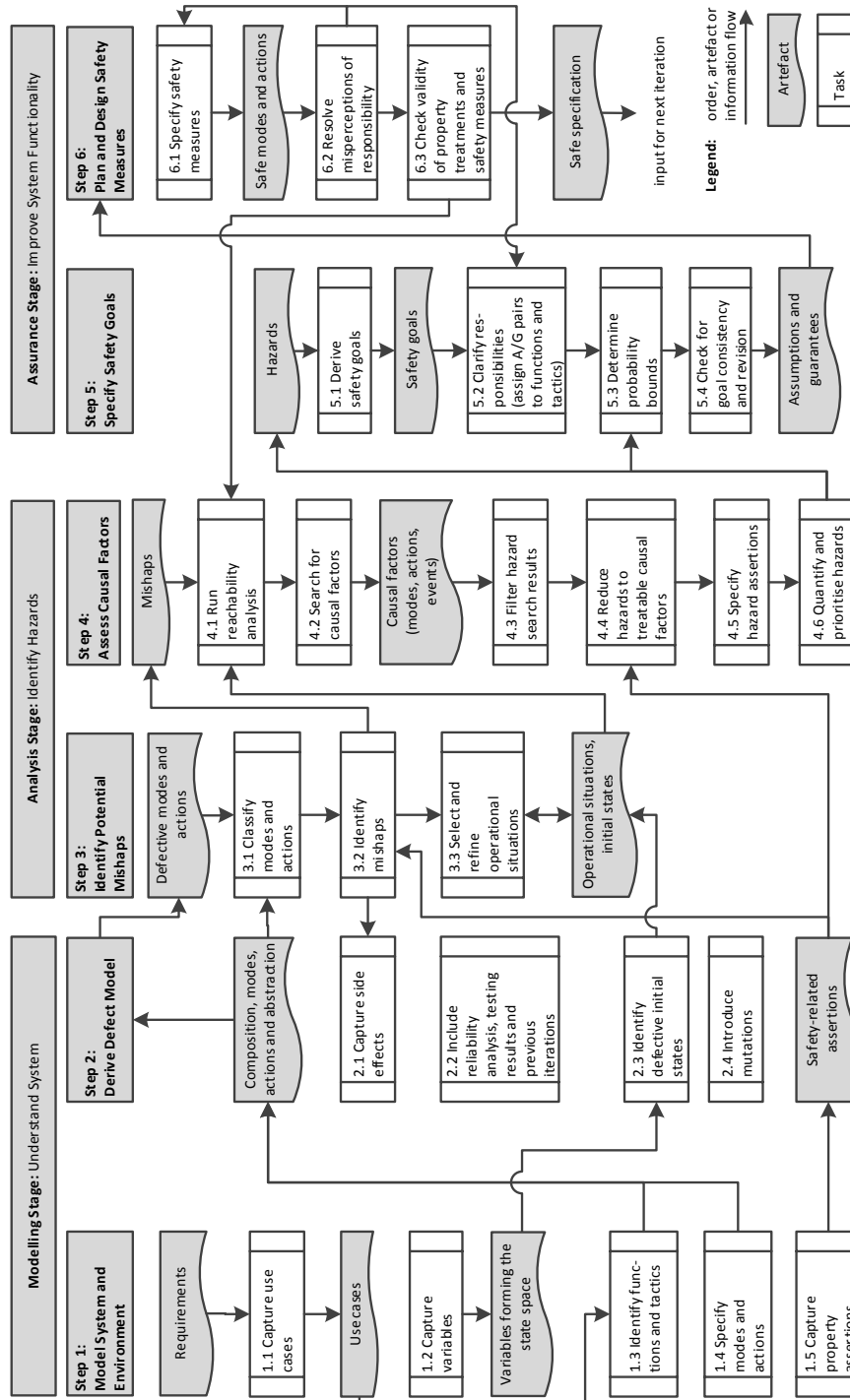


Figure 5.2.: Overview of stages, steps and sub-steps of the procedure

## Part II.

# Application and Evaluation

People have said that computing is a fast moving subject and what they mean is that the wheel of re-incarnation goes faster.

---

*(Roger M. Needham 2001)*

This chapter discusses two applications of the proposed method: an automated teller machine and a commercial road vehicle.

### Contents

6.1	Pilot Case: Automated Teller Machine	76
6.1.1	Step 1: Model System and Environment	76
6.1.2	Step 2: Derive Defect Model	81
6.1.3	Step 3: Identify Potential Mishaps	82
6.1.4	Step 4: Assess Causal Factors	82
6.1.5	Step 5: Specify Safety Goals	84
6.1.6	Step 6: Plan and Design Safety Measures	86
6.2	Approval Case: Commercial Road Vehicle	90
6.2.1	Step 1: Model System and Environment	90
6.2.2	Step 2: Derive Defect Model	94
6.2.3	Step 3: Identify Potential Mishaps	95
6.2.4	Step 4: Assess Causal Factors	96
6.2.5	Step 5: Specify Safety Goals	99
6.2.6	Step 6: Plan and Design Safety Measures	100

## 6.1. Pilot Case: Automated Teller Machine

The unit of analysis in the pilot case is an *automated teller machine (ATM)*.

### 6.1.1. Step 1: Model System and Environment

**Artefact  $\mathcal{R}$  – Use Cases and Property Assertions** As a prerequisite, the contents of  $\mathcal{R}$  are derived from an example discussed by Broy et al. (2012).

## 6.1. Pilot Case: Automated Teller Machine

Use case	Withdraw cash (modelled below as $WC_E, WC_S$ )
Goal	The customer is served cash money.
Scope	$\mathcal{A}_S$ , level: primary task in $\mathcal{A}_{E use}$ , primary actor: $\mathcal{A}_E$
Preconditions	There is enough money on the customer's bank account.
Minimal Guarantees	The ATM provides cash if no errors occurred during the transaction, otherwise an error message is displayed and the balance is left unchanged.
Success Guarantees	The ATM provides the requested amount of cash and the account balance is updated accordingly.
Trigger	1. The customer inserts his or her EC card into the ATM card slot.
Description (interaction sequence)	2. The ATM displays the default service options. 3. The customer selects "cash withdrawal." 4. The ATM displays several predefined amounts and an option for a custom amount of cash. 5. The customer selects an "amount" option. 6. The ATM processes the request by reducing the customer's balance by the selected amount and returns both cash and the EC card.

Table 6.1.: ATM use case "withdraw cash" as a prerequisite for Step 1

**Step 1.1** The Tables 6.1 and 6.2 describe two simplified use cases of an ATM.<sup>1</sup>

**Step 1.2** From Step 1.1, we derive variables to whom we assign types, for example,

$$\begin{aligned}
 \text{type}(\text{position}_{\text{hands}}) &= \{\text{inCashSlot}, \text{atPanel}, \text{atCardslot}, \text{awayFromATM}, \dots\} \\
 \text{type}(\text{position}_{\text{cash}}) &= \{\text{inStorage}, \text{inCashSlot}, \text{inHands}, \text{unknown}\} \\
 \text{type}(\text{lid}_{\text{cashslot}}) &= \{\text{open} \equiv 5, \text{closed} \equiv 0, \text{inTransition} \in \{1, \dots, 4\}\} \\
 \text{type}(\text{selection}_{\text{panel}}) &= \{\text{pin}(P) \mid P \in \mathbb{N}\} \cup \{\text{amount}(A) \mid A \in \mathbb{N}\} \cup \\
 &\quad \{\text{depose}, \text{withdraw}\} \\
 \text{type}(\text{content}_{\text{cashslot}}) &= \{\text{cash}(C) \mid C \in \mathbb{N}\} \cup \{\text{empty}\}
 \end{aligned}$$

**Artefact  $\mathcal{V}$  – Variables** Table 6.3 describes the variables for safety analysis including the channels modelling the safety-related interface of an ATM according to Figure 4.1. This table also contains variables identified in Steps 2, 3 and 4 of the procedure.

We need to make domain properties explicit when building the world model: Variables in parentheses are ignored to reduce the need for properties, such as

$$\text{position}_{\text{hands}} = \text{inCashSlot} \rightarrow \text{obstacle}_{\text{cashslot}}$$

<sup>1</sup>We use the notation  $\langle \text{MTS} \rangle_{\langle \text{Agent} \rangle \langle \text{Aspect} \rangle}$ : E for  $\langle \text{Agent} \rangle$  denotes the environment, S the system. The  $\langle \text{Aspect} \rangle$  qualifier is empty for  $\mathbb{M}_{use}$ , f for  $\mathbb{M}_{fail}$  and s for  $\mathbb{M}_{save}$ . For example,  $\text{DEP}_{Ef}$  stands for an MTS which is part of a use case and models a defective fragment of the tactic DEP in  $\mathbb{M}_{fail}$ .

## 6. Case Study

Use case	Depose cash (modelled below as $DEP_E, DEP_S$ )
Goal	The customer deposited cash to his or her bank account via the ATM.
Scope	$\mathcal{A}_S$ , level: primary task in $\mathcal{A}_E _{use}$ , primary actor: $\mathcal{A}_E$
Preconditions	The customer has a bank account.
Minimal Guarantees	If the ATM cannot store the deposited cash then it provides an error message and returns the cash immediately.
Success Guarantees	The ATM correctly transforms the inserted and stored cash into an update of the customer's account balance.
Trigger	1. The customer inserts his or her ATM card into the card slot.
Description (interaction sequence)	2. The ATM displays the default service options. 3. The customer selects "cash deposit." 4. The ATM asks the customer to insert cash into the cash slot. 5. The customer inserts cash into the cash slot. 6. The ATM counts the inserted cash, displays the calculated amount, adds it to the customer's account balance and returns the ATM card.

Table 6.2.: ATM use case "depose cash" as a prerequisite for Step 1

which reifies the concept of *physical obstacles* (D24), and

$$\begin{aligned} \text{position}_{\text{card}} = \text{inATM} &\leftrightarrow \text{status}_{\text{cardslot}} = \text{pulledIn} \wedge \\ \text{position}_{\text{card}} = \text{atSlot} &\leftrightarrow \text{status}_{\text{cardslot}} = \text{atSlot} \end{aligned}$$

which constrains the valuations of variables related to *card position* (D8). It is often possible to revise the model to eliminate redundant variables and still reflect observability and controllability.

**Step 1.3** The two agents  $\mathcal{A}_E$  and  $\mathcal{A}_S$  specify behaviour of a bank customer as the human operator at the side of the environment and the ATM with the IT infrastructure of the bank at the side of the system. Our information for the world model is provided by  $\mathcal{R}_f$ , that is, a set of use cases. Note that this model is set up regarding the ATM as both a banking service and a mechatronic device. Applying Definition 4.1 for the aspect  $\mathbb{M}_{use}$ , we can set

$$\mathcal{A}_E|_{use} = ID_E \oplus DEP_E \oplus WC_E \quad (6.1)$$

$$\mathcal{A}_S|_{use} = (SRV_S \oplus DEP_S \oplus WC_S) \otimes ID_S \otimes LID_S \quad (6.2)$$

$$\mathcal{M}|_{use} = \mathcal{A}_E|_{use} \otimes \mathcal{A}_S|_{use} \quad (6.3)$$

For sake of simplicity, we only consider one initial state partially given by

$$\Sigma_0 = \{\sigma_0\} \text{ with } \sigma_0 = [\text{lid}_{\text{cashslot}} \mapsto \text{closed}, \text{position}_{\text{hands}} \mapsto \text{awayFromATM}, \dots]$$

**Step 1.4** The pair of MTSs  $WC_E$  and  $WC_S$  for use case "withdraw cash" is depicted in Figure 6.1. Table A.12 describes these MTSs as explained in Section 2.3.1. The



### 6.1. Pilot Case: Automated Teller Machine

$\cap$	$\not\subseteq$	$\text{mon}(\mathcal{A}_E) \cup \text{ctr}(\mathcal{A}_E)$	$\text{mon}(\mathcal{A}_E) \setminus \text{ctr}(\mathcal{A}_E)$	$\text{ctr}(\mathcal{A}_E) \setminus \text{mon}(\mathcal{A}_E)$	$\text{mon}(\mathcal{A}_E) \cap \text{ctr}(\mathcal{A}_E)$
$\not\subseteq$ $\text{mon}(\mathcal{A}_S) \cup \text{ctr}(\mathcal{A}_S)$	‡		(no variables identified)	(no variables identified)	$\text{position}_{\text{card}},$ $(\text{position}_{\text{hands}})$
$\text{mon}(\mathcal{A}_S) \setminus \text{ctr}(\mathcal{A}_S)$		$\text{status}_{\text{cardslot}},$ $\text{pin}_{\text{card}}$	(no variables identified)	$\text{selection}_{\text{panel}},$ $(\text{obstacle}_{\text{cashslot}})$	$\text{position}_{\text{hands}}$
$\text{ctr}(\mathcal{A}_S) \setminus \text{mon}(\mathcal{A}_S)$	‡		$\text{lid}_{\text{cashslot}},$ $\text{content}_{\text{cashslot}},$ $\text{content}_{\text{display}},$ $\text{voltage}_{\text{casing}}$	‡	‡
$\text{mon}(\mathcal{A}_S) \cap \text{ctr}(\mathcal{A}_S)$		$\text{content}_{\text{deposit}},$ $\text{balance}_{\text{account}},$ $\text{amount}_{\text{tempStore}}$	$\text{lid}_{\text{pressure}_{\text{cashslot}}}$	(no variables identified)	$\text{position}_{\text{cash}}$

Table 6.3.: Variables in  $\mathcal{V} \setminus \mathcal{V}_m$  for safety analysis; ‡... unnecessary in this model

use case “depose cash” is modelled similarly. Note that the insert action in Figure 6.1 shows up twice, in  $WC_E$  and in  $DEP_E$ . Multiple occurrences will be fixed by the superimposition  $WC_E \oplus DEP_E$ .

**Artefact  $\mathcal{M}$  – Functions and Tactics  $\mathbb{M}_{\text{use}}$**  Figure 6.1 shows several MTSs and Table A.12 provides further details about them.

**Step 1.5  $\mathcal{R}_p$**  provides one safety goal, and we disregard goals resulting from Step 5 or a previous iteration.

**Artefact  $\Gamma$  – Safety-related Assertions** The informal description of  $\Gamma$  is imported from  $\mathcal{R}_p$  and shown in Table A.11, the formal description is developed in the next steps.

**Discussion** Based on the system boundary from Step 1.2, the boundary of a function can be derived from its operational specification  $\mathcal{M}$ . It is convenient to impose boundaries as a constraint for the development of an operational specification. The example neither explains what an operational specification allows nor discusses changes of the system boundary.

The two hierarchies resulting from Step 1.3 can be asymmetric. The example, however, approximates symmetry to ease the conduct of the method and to support traceability.  $SRV_E$  can be extracted from  $WC_E$  and  $DEP_E$  to reduce redundancy. (Issues of tool data integrity and optimisation are out of scope.)

For Steps 1.3 and 1.4, interaction sequences can be encoded into pre constraints or by a set of modes. Use cases to be performed in sequence can be modelled by several modes and actions. Actions that are required to be performed independently (i.e. minimum use of mode channels) can be put into concurrent MTSs. Modes can be reused by superimposition.

## 6. Case Study

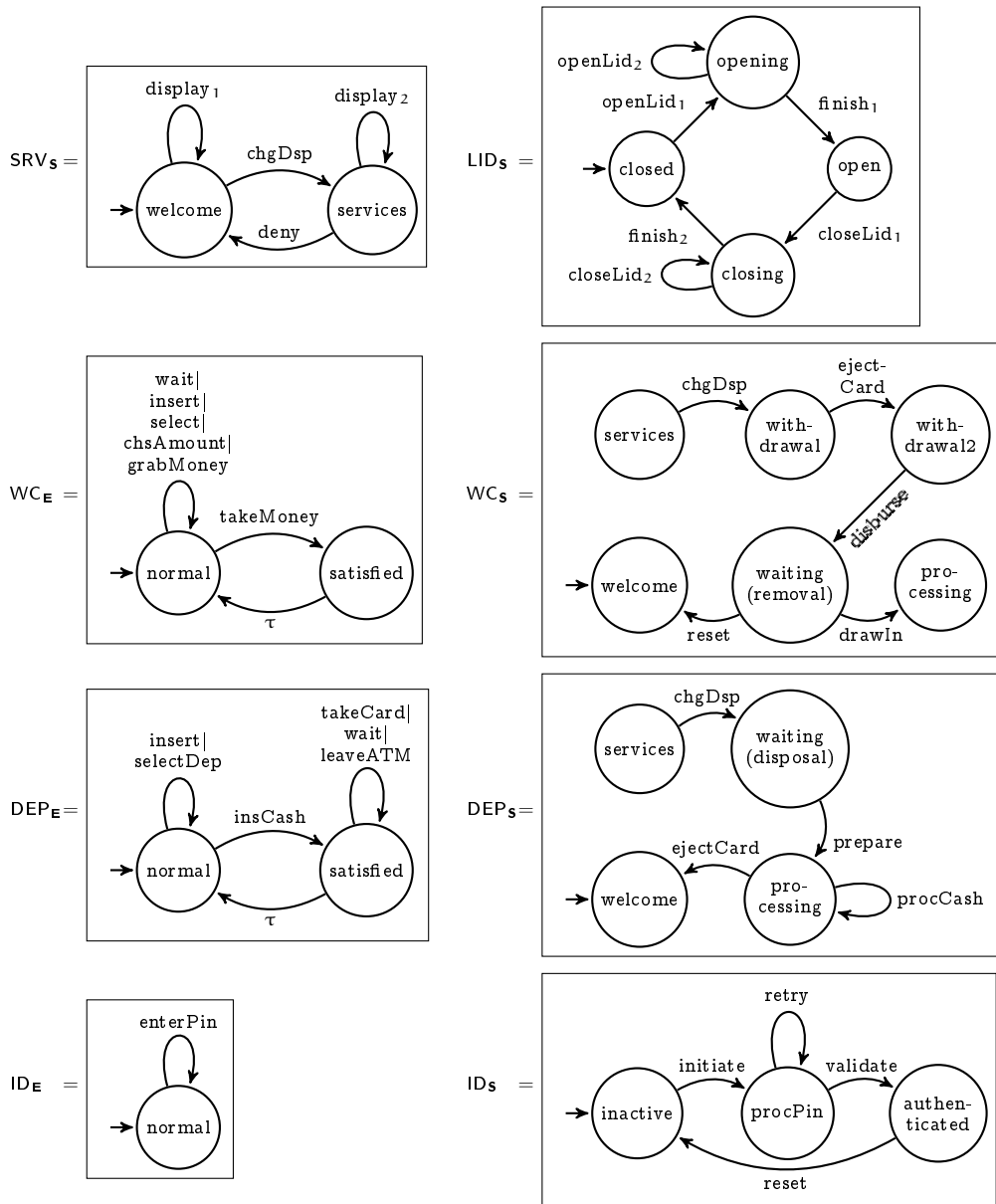
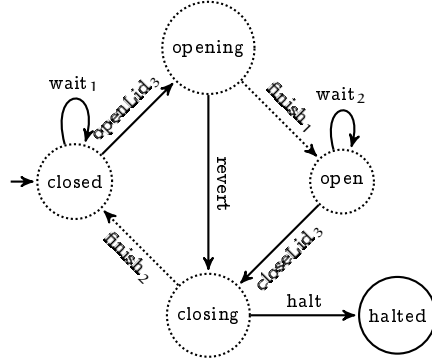


Figure 6.1.: Pairs of MTSs in  $M_{use}$ ;  $a_1 | \dots | a_n$  represents  $n$  collapsed transition arrows as a notational convention due to lack of space (cf. Definition 2.10)


 Figure 6.2.:  $LID_{Sf}$  as a part of the defect model  $M_{fail}$ 

For Step 1.5, goals being part of the use cases in  $\mathcal{R}_f$  are omitted in both the Table A.11 and the model.

### 6.1.2. Step 2: Derive Defect Model

I neglected side effects (Step 2.1) and reliability analyses (Step 2.2) as these techniques are out of scope of the case study.

**Steps 2.3 and 2.4** Mutations can capture defective behaviour of physical actions: For  $closeLid_1$  in the open mode of the function  $LID_S$ , we add the actions  $wait_2$  and  $closeLid_3$  which carry a mutated state constraint  $pre \equiv \top$  and are subject of choice with a probability of 50%.  $LID_{Sf} \in M_{fail}$  (Figure 6.2) applies the patterns “random” and “currently unacceptable execution” from Table A.2 in their “permanent, non-deterministic” variants to model the two defects. The choice between defective and specified actions is non-deterministic because a  $pre \equiv \top$  covers the mode open as opposed to  $t_{WC_S} \geq 2 \vee m_{SRV_S} = welcome \vee m_{DEP_S} = processing$  of  $closeLid_1$  (Table A.12). The choice among the two defective actions, however, incorporates the “probabilistic” variant of the “random” pattern. The same pattern is applied to  $closeLid_2$ ,  $openLid_1$  and  $openLid_2$  using  $wait_1$ ,  $openLid_3$ ,  $halt$  and  $revert$ .

$LID_{Sf}$  represents defects after which the ATM randomly closes and opens the lid. Evidence for the possibility of such failures can be provided by defect analysis of the system design (Step 2.2).

**Artefact  $\mathcal{M}$  – Defect Model  $M_{fail}$**  The defect model is depicted in Figure 6.2, actions are specified in Table A.12.  $M_{fail}$  extends  $\mathcal{M}|_{use}$  as follows:

$$\mathcal{A}_E|_{use, fail} = ID_E \oplus DEP_E \oplus WC_E \quad (6.4)$$

$$\mathcal{A}_S|_{use, fail} = (SRV_S \oplus DEP_S \oplus WC_S) \otimes ID_S \otimes (LID_S \oplus LID_{Sf}) \quad (6.5)$$

## 6. Case Study

### 6.1.3. Step 3: Identify Potential Mishaps

**Step 3.1** Going through the model, we identify the control actions  $\text{openLid}_2$  and  $\text{closeLid}_2$  to affect the movement of physical parts of the ATM.

**Step 3.2** By applying the state guide word “clamp”, we can identify the mishap *squeezed hands* (M7):

$$\begin{aligned} \text{position}_{\text{MachineParts}} \neq \text{wide} &\equiv \text{lid}_{\text{cashslot}} \neq \text{open} \\ \text{occupiedBy}_{\text{MachineParts,BodyParts}} &\equiv \text{position}_{\text{hands}} = \text{inCashSlot} \\ \phi_{\text{sqzHnd}} &\equiv \text{lid}_{\text{cashslot}} \neq \text{open} \\ &\quad \wedge \text{position}_{\text{hands}} = \text{inCashSlot} \end{aligned}$$

Using the guide word “electrically shock”, we obtain the mishap *electric shock* (M13):

$$\begin{aligned} \text{voltage}_{\text{ContactSurface}} = \text{high} &\equiv \text{voltage}_{\text{casing}} = \text{underCurrent} \\ \text{touchedBy}_{\text{ContactSurface,BodyParts}} &\equiv \text{position}_{\text{hands}} = \text{atPanel} \\ \phi_{\text{elShock}} &\equiv \text{voltage}_{\text{casing}} = \text{underCurrent} \\ &\quad \wedge \text{position}_{\text{hands}} = \text{atPanel} \end{aligned}$$

**Artefact  $\Phi$  – Mishaps** The informal description is shown in Table A.11, the formal description is provided here.

**Step 3.3** Consider the operational situation  $\sigma_{\text{use}}$  (Definition 4.2) for  $\mathcal{M}$  with

$$\sigma_{\text{use}}(\sigma) = \begin{cases} \top, & \text{if } \sigma = \sigma_0 \\ \perp, & \text{else} \end{cases}$$

**Discussion** Consider mishap M13 in Step 3.2: we might return to Step 2.1 to elicit side effects as additional effects of existing actions or as an additional MTS producing these effects depending on certain modes. For sake of brevity, I only discuss the assessment and treatment of  $\phi_{\text{sqzHnd}}$  in the following.

### 6.1.4. Step 4: Assess Causal Factors

**Step 4.1** Model checking of  $\mathcal{M}$  (including the defect model and  $\kappa$ -completion) can compute how the *mishap*  $\phi_{\text{sqzHnd}}$  can be reached. From such reachability analysis, we can derive three complex actions (Definition 2.10),  $(.+;\text{grabMoney};\text{wait}^+)$  of  $\text{WC}_E$ ,  $(.+;\text{drawIn})$  of  $\text{WC}_S$  and  $(.+;\text{closeLid}_2^+)$  of  $\text{LID}_S$ , concurrently executed in  $\mathcal{M}$ . Figure 6.3 shows a run starting in  $\sigma_{\text{use}}$  and its hazardous actions concerning  $\phi_{\text{sqzHnd}}$ . Figure 6.4 shows further hazardously performable actions related to  $\phi_{\text{sqzHnd}}$ .

**Steps 4.2 and 4.3** The set of runs resulting from Step 4.1 can now be searched for hazards, for example, within the last 20 states of the runs (i.e.  $k = 20$ ).

Our search starts from the conjuncts of  $\phi_{sqzHnd}$ . Using the event guide word “start of  $e$ ” for the event  $e \equiv \text{lid}_{\text{cashslot}} \neq \text{open}$  as mentioned in  $\phi_{sqzHnd}$ , we start looking for causal factors:

$$\begin{aligned} \text{cf}_1 &\equiv \bar{F}^{\leq 20}(e \wedge \bar{X}\neg e) \\ &\equiv \bar{F}^{\leq 20}(\text{lid}_{\text{cashslot}} \neq \text{open} \wedge \bar{X}\neg \text{lid}_{\text{cashslot}} \neq \text{open}) \\ &\equiv \bar{F}^{\leq 20}(\text{lid}_{\text{cashslot}} \neq \text{open} \wedge \bar{X}\text{lid}_{\text{cashslot}} = \text{open}) \end{aligned}$$

Next, we use the action guide word “unexpected execution of  $\text{closeLid}_2$ ”:

$$\begin{aligned} \text{cf}_2 &\equiv \bar{F}^{\leq 20}(\bar{X}(\neg \text{enabl} \vee \neg \text{closeLid}_2.\text{pre}) \wedge \text{closeLid}_2.\text{post}) \\ &\equiv \bar{F}^{\leq 20}(\bar{X}(\neg(\text{m}_{\text{LID}_s} = \text{closing}) \vee \neg(\text{lid}_{\text{cashslot}} \neq \text{closed}))) \\ &\quad \wedge (\exists x : \bar{X}(\text{lid}_{\text{cashslot}} = x) \wedge \text{lid}_{\text{cashslot}} = x - 1)) \end{aligned}$$

Either of these formulae leads to the mode closing. Using the mode guide word “activation of  $m$ ” yields the search expression:

$$\begin{aligned} \text{cf}_3 &\equiv \bar{F}^{\leq 20} \text{m}_{\mathcal{M}} = m \\ &\equiv \bar{F}^{\leq 20} \text{m}_{\text{LID}_s} = \text{closing} \end{aligned}$$

$\text{cf}_3$  discloses several actions as being hazardous:  $\text{closeLid}_1$ ,  $\text{revert}$  and  $\text{closeLid}_3$ . Taking the pattern “change of  $m$ ” from Table A.1, we can now look for

$$\text{cf}_{3'} \equiv \bar{F}^{\leq 20}(\bar{X}(\text{m}_{\text{LID}_s} = \text{opening}) \wedge \text{m}_{\text{LID}_s} = \text{closing})$$

Furthermore, in order to capture causal factors in the environment, we apply the pattern “ $e$  stopped too late” for both the  $\phi_{sqzHnd}$ -event  $e \equiv \text{position}_{\text{hands}} = \text{inCashSlot}$  and the orientation event  $e' \equiv \text{lid}_{\text{cashslot}} \neq \text{open}$ :

$$\begin{aligned} \text{cf}_4 &\equiv \bar{F}^{\leq 20}(\neg e \wedge \bar{X}(e \bar{U}(e' \wedge \bar{X}(\neg e')))) \\ &\equiv \bar{F}^{\leq 20}(\neg \text{position}_{\text{hands}} = \text{inCashSlot} \\ &\quad \wedge \bar{X}(\text{position}_{\text{hands}} = \text{inCashSlot} \\ &\quad \quad \bar{U}(\text{lid}_{\text{cashslot}} \neq \text{open} \wedge \bar{X}(\neg \text{lid}_{\text{cashslot}} \neq \text{open})))) \end{aligned}$$

$\text{cf}_4$  combines environment and system behaviour. The event guide word “start of  $\text{position}_{\text{hands}} = \text{inCashSlot}$ ” yields a formula for a temporally less recent search:

$$\text{cf}_5 \equiv \bar{F}^{\leq 20}(\text{position}_{\text{hands}} = \text{inCashSlot} \wedge \bar{X}\neg \text{position}_{\text{hands}} = \text{inCashSlot})$$

**Step 4.4** For example, the check of  $\text{cf}_{3'} \wedge \text{cf}_5$  confirms the possibility of a specification defect because of violable environment responsibility (maloperation), and a random system failure because of realisation defects in  $\mathcal{A}_S$ . The hazardous run shown in Figure 6.3 contains a failure of  $\text{WC}_S$  where the lid is closed too early.

## 6. Case Study

Step 1	Steps 3 and 4			Step 4.6				Step 5.3
$\mathcal{M}$	Mishap $\in \Phi$	Operational Situation	Hazard $\in \mathcal{H}$	S	W	A	G	IC
LIDs	$\phi_{sqzHnd}$	$\sigma_{use}$	-	1	-	-	-	-
LIDs	$\phi_{sqzHnd}$	$\sigma_{use}$	$\chi_{unexpClos}$	1	l	m	h	medium

Table 6.4.: Hazard assessment similar to DIN 19250, IEC 61508 and ISO 26262; S... severity, W... probability, A... exposure, G... detectability and controllability, IC... integrity class, l/m/h... low/medium/high; cells in grey indicate related content

**Step 4.5** We combine some of the formulae  $cf_1$  to  $cf_5$ , derived from the search for causal factors, to specify unexpected and hazardous runs. For example, we can define *unexpected closure* ( $H26$ ) as follows:

$$\chi_{unexpClos} \equiv cf_{3'} \wedge cf_5 \quad (6.6)$$

To state another hazard according to Definition 4.8, we have to check the formula

$$\mathbf{EF}(\chi_{unexpClos} \rightarrow \mathbf{P}_{>0.01}[\mathbf{XF}\phi_{sqzHnd}])$$

in  $\mathcal{M}$  for an upper bound of 1% for ATM risks.

**Step 4.6** Table 6.4 exemplifies the estimation of the characteristics by which an integrity class can be determined in Step 5.3.

**Artefact  $\mathcal{H}$  – Hazards** The result of hazard assessment is depicted in Table 6.4. Note that this pilot case does neither claim nor aim to provide exhaustive analysis results.

**Discussion** For any action  $a$  referred to in the formulae of Steps 4.2 and 4.3, Definition 2.15 guides the transform of the function  $a.post : \mathcal{E}|_{\text{ctr}(\mathcal{M})}$  into an action effect formula. Many of the formulae shown above can be simplified. Formula rewriting, however, is an issue out of scope of the present work. Furthermore, in an automated setting, the results for Step 4.6 can be computed from  $\mathcal{M}$ .

### 6.1.5. Step 5: Specify Safety Goals

**Steps 5.1** The ATM is usable if we accept states which fulfil either  $cf_{3'}$  or  $cf_5$ . Hence, we derive the mitigation goal (Definition 4.9) *safe interaction* ( $G9$ ):

$$\begin{aligned}
 \gamma_{\text{treat.unexpClos}} &\equiv P_{\geq \text{Pr}}[\mathbf{G}\neg\phi_{\text{sqzHnd}} \wedge \mathbf{GF}\neg\chi_{\text{unexpClos}}] \\
 &\equiv P_{\geq 0.99}[\mathbf{G}\neg(\text{lid}_{\text{cashslot}} \neq \text{open} \\
 &\quad \wedge \text{position}_{\text{hands}} = \text{inCashSlot}) \\
 &\quad \wedge \mathbf{GF}\neg(\bar{\mathbf{F}}^{\leq 20}(\bar{\mathbf{X}}(\text{m}_{\text{LID}_s} = \text{opening}) \wedge \text{m}_{\text{LID}_s} = \text{closing})) \\
 &\quad \wedge \bar{\mathbf{F}}^{\leq 20}(\text{position}_{\text{hands}} = \text{inCashSlot} \\
 &\quad \wedge \bar{\mathbf{X}}\neg\text{position}_{\text{hands}} = \text{inCashSlot})]
 \end{aligned}$$

We might be even more tolerant by stating an alleviation goal (Definition 4.9):

$$\gamma'_{\text{treat.unexpClos}} \equiv P_{\leq 1-0.99}[\mathbf{F}(\phi_{\text{sqzHnd}} \vee \chi_{\text{unexpClos}}) \mathbf{U}^{>5}\neg(\phi_{\text{sqzHnd}} \vee \chi_{\text{unexpClos}})]$$

**Step 5.2** To the safety goal  $\gamma_{\text{treat.unexpClos}}$ , we assign an A/G pair  $(\text{As}_1, \text{Gr}_1)_{\text{treat.unexpClos}}$  including the assumption (“proper use”,  $\text{As}_{19}$ )

$$\text{As}_1 \equiv P_{<0.05}[\mathbf{F}(\text{position}_{\text{hands}} = \text{inCashSlot} \mathbf{U}^{>20}\text{position}_{\text{hands}} \neq \text{inCashSlot})]$$

and the guarantee ( $\text{Gr}_{10}$ )

$$\text{Gr}_1 \equiv P_{\geq 0.99}[\mathbf{GF}\neg(\bar{\mathbf{F}}^{\leq 20}(\bar{\mathbf{X}}(\text{m}_{\text{LID}_s} = \text{opening}) \wedge \text{m}_{\text{LID}_s} = \text{closing}))]$$

In this responsibility relationship,  $\text{As}_1$  suggests mitigation of maloperation and  $\text{Gr}_1$  suggests mitigation of defects in the mode opening to be developed in Step 6.1.

From  $\gamma'_{\text{treat.unexpClos}}$ , we can derive the guarantee

$$\text{Gr}_2 \equiv P_{\leq 0.01}[\mathbf{F}(\phi_{\text{sqzHnd}} \mathbf{U}^{>5}\neg\phi_{\text{sqzHnd}})]$$

$\text{Gr}_2$  suggests a passive safety measure at the side of the system to be developed in Step 6.1. As an advantage, we obtain an assumption  $\text{As}_2 \equiv \top$  which reduces the responsibility of the environment when using  $\text{LID}_s$ .

One can see that both assumptions are controllable by the environment and both guarantees are controllable by the system.

**Steps 5.3 and 5.4** The integrity classes of the functions and tactics including the safety measures are kept in Table 6.4 column IC. As we have only one safety goal, we can omit Step 5.4.

**Artefact  $\Gamma$  – Safety Goals, A/G pairs** Both existing and derived safety goals, and A/G pairs are visualised in Figure 6.5 and kept in Table A.11.

## 6. Case Study

**Discussion** Although usual in requirements engineering (Section 2.1), I disregarded relationships among the assertions based on their content and breakdown, as repeatedly suggested in the Sections 4.3 to 4.5 and Section 4.6.2. Nevertheless, the meanings of the relationships depicted in Figure 6.5 can be derived from Chapter 4, the Sections 5.2 and 5.3, and Figure 7.1.

### 6.1.6. Step 6: Plan and Design Safety Measures

After having performed hazard analysis, we can derive the safety measures by regarding the A/G pairs stated in Step 5, for example,  $(AS_1, Gr_1)_{treat.unexpClos}$ .

**Step 6.1** By superimposing  $AS_{S_s} \in \mathbb{M}_{save}$  onto  $LID_S \oplus LID_{S_f}$ , we operationally specify an “attenuation mechanism” from Table A.4 including obstacle detection: the function  $AS_{S_s}$  realises the safety guarantee  $Gr_{10}$  which contributes to the safety goal  $G_9$ .

**Artefact  $\mathcal{M}$  – Safety Measures  $\mathbb{M}_{save}$**  The findings of Steps 3 and 4 are used to introduce safety measures which alter both agents, as defined by the Equations (6.4) and (6.5), as follows:

$$\mathcal{A}_E = ID_E \oplus DEP_E \oplus (WC_E \oplus AS_{E_s}) \quad (6.7)$$

$$\begin{aligned} \mathcal{A}_S &= (SRV_S \oplus DEP_S \oplus WC_S) \\ &\quad \otimes ID_S \otimes (LID_S \oplus LID_{S_f} \oplus AS_{S_s}) \end{aligned} \quad (6.8)$$

These MTSs are shown in Figure 6.6. More details are given in Table A.12.

**Step 6.2** The assumptions  $As_{19}$  and  $As_{27}$  are both weak, because they are easily violable. Assumption  $As_{19}$  can be even weaker than assumption  $As_{27}$ : we can expect the bank customer more often following the behaviour of  $AS_{E_s}$  obeying assumption  $As_{27}$  instead of following the behaviour of  $WC_{E_f}$ .

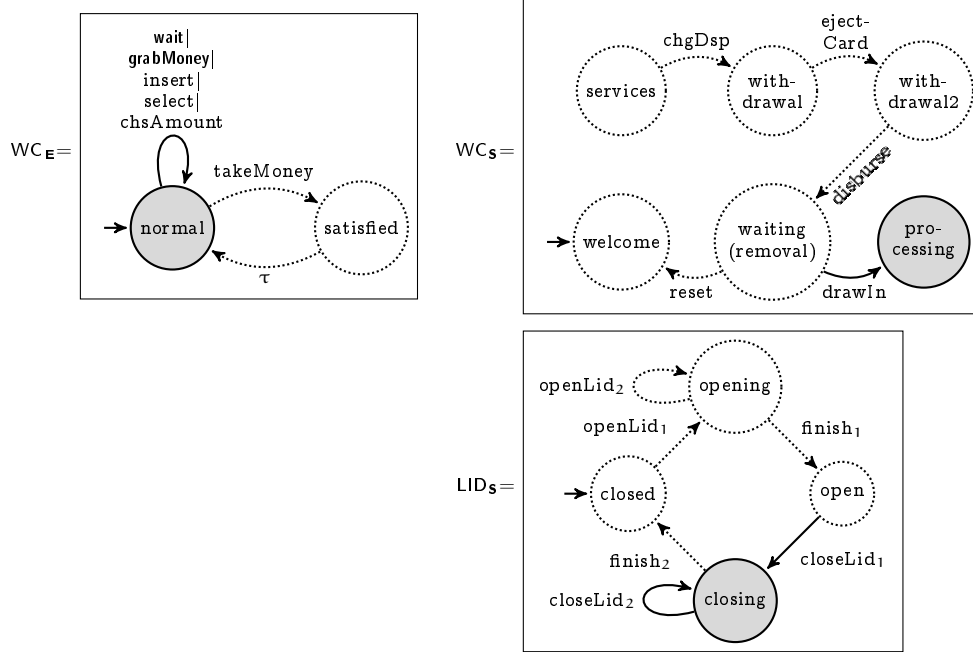
**Step 6.1 (again)** We introduce the tactic  $AS_{E_s} \in \mathbb{M}_{save}$  superimposed onto  $WC_E$ .  $AS_{E_s}$  applies the pattern “fail-safe action” in its “fail-operational” variant to the environment.  $AS_{E_s}$  helps determine the strength of assumption 19 and derive assumption 27.

In Step 5.2, we started to address misperceptions of responsibility by making assumptions and guarantees explicit. The safety goal  $G_9$ , however, can be obstructed by violating the assumption  $As_{19}$ . This obstruction corresponds to the cases 2 and 3 in Table 4.3 and would be an argument for taking  $\gamma'_{treat.unexpClos}$ , the weakened assumption  $As_2$  and the strengthened guarantee  $Gr_2$ , as explained above.

**Step 6.3** The safety measures in  $\mathcal{M}$  can now be checked against  $\gamma'_{treat.unexpClos}$ . The tool support required for this, however, is out of scope of this work.



6.1. Pilot Case: Automated Teller Machine



MTSs/Variables	$t_1$	$t_2 = t_1 + 1$	...	$t_3 \geq t_1 + 5$	...	$t_4 \geq t_1 + 7$
$WC_E$	$\frac{+}{\rightarrow}_O$	$\frac{\text{grabMoney}}{\rightarrow}_O$	$\frac{\text{wait}^+}{\rightarrow}_O$	$\frac{\text{wait}}{\rightarrow}_O$	$\frac{\text{wait}^+}{\rightarrow}_O$	$\frac{\text{wait}}{\rightarrow}_O$
$mWC_E$						normal
$WC_S$	$\frac{+}{\rightarrow}_O$			$\frac{\text{drawIn}}{\rightarrow}_O$		
$mWC_S$	waiting			processing		
$LID_S$	$\frac{\text{finish}}{\rightarrow}_O$	$\frac{\kappa}{\rightarrow}_O$	$\frac{\kappa^+}{\rightarrow}_O$	$\frac{\kappa}{\rightarrow}_O$	$\frac{\text{closeLid}^+}{\rightarrow}_O$	$\frac{\text{closeLid}}{\rightarrow}_O$
$mLID_S$	open				closing	closing
$\text{content}_{\text{display}}$				welcScreen		
$\text{position}_{\text{hands}}$		inCashSlot				inCashSlot
$\text{lid}_{\text{cashslot}}$	open			open	somePos	somePos
$\text{content}_{\text{cashslot}}$			cash(X)			

Figure 6.3.: Actions of  $\mathcal{M}$  generating a run which violates safety goal G9 by mishap M7

6. Case Study

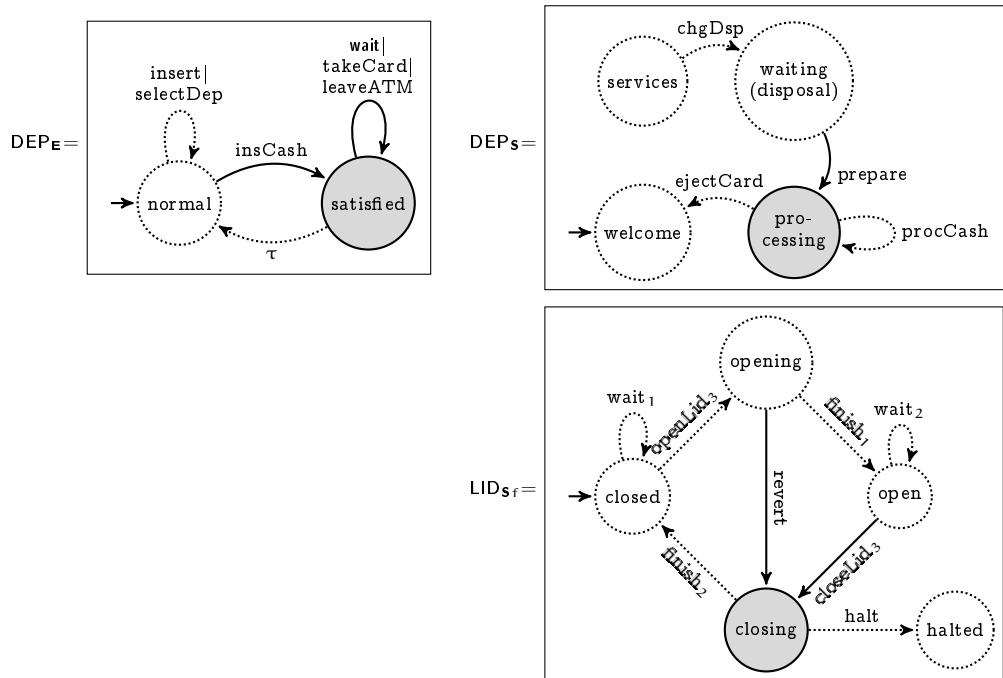


Figure 6.4.: Three concurrently executed complex actions (solid arrows) of  $\mathcal{M}$

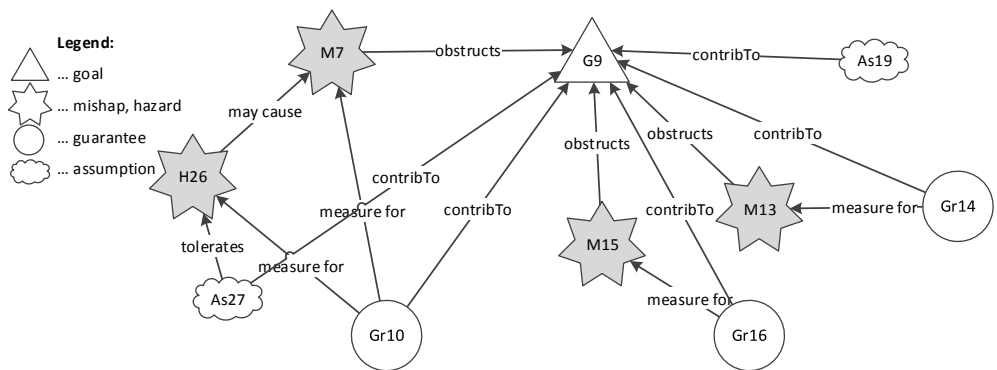


Figure 6.5.: Assertion graph showing relationships between property assertions in  $\Gamma$  built up during the Steps 1.5, 3, 4, 5.1 and 6

6.1. Pilot Case: Automated Teller Machine

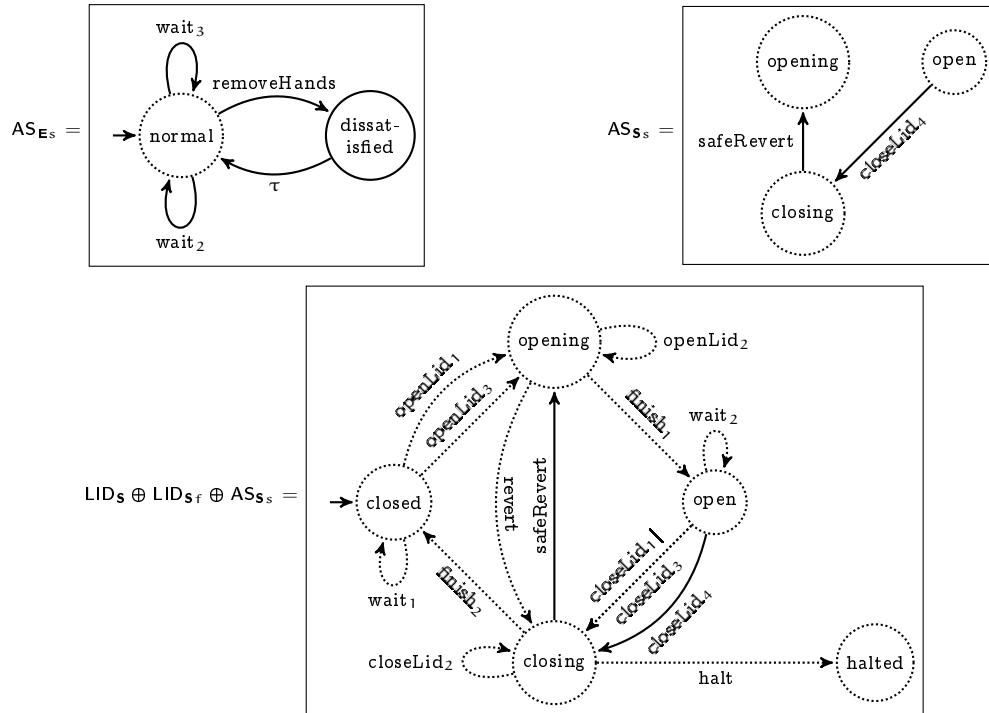


Figure 6.6.: Pair of MTSs comprising safety measures in  $\mathcal{A}_s|_{save}$  and the superimposed function  $LID_s \oplus LID_{S_f} \oplus AS_{S_s}$

## 6. Case Study

Use Case	Use truck (modelled below as $Missions_E, Trucks$ )
Goal	G27: The truck fulfils one of its purposes.
Scope	$\mathcal{A}_S$ ; level: primary task in $\mathcal{A}_E _{use}$ ; primary actor: $\mathcal{A}_E$
Preconditions	Sufficient amount of energy (i.e. fuel, charge of battery).
Minimal Guarantees	The truck protects the trucker, the goods and the environment.
Success Guarantees	The truck helps accomplish the trucker's mission.
Trigger	1. The trucker activates the vehicle by applying the key.
Description (interaction sequence)	2. He or she performs other use cases, for example, "park at steep hill" or "use brakes", to accomplish his or her missions. 3. The vehicle reacts correctly to his or her commands. 4. The trucker deactivates the vehicle.

Table 6.5.: CRV use case "use truck"

Use Case	Park at steep hill (modelled below as $Park_E, DriveMoves$ )
Goal	G5: The truck is parked at a steep hill.
Scope	$\mathcal{A}_S$ ; level: primary task in $\mathcal{A}_E _{use}$ ; primary actor: $\mathcal{A}_E$
Preconditions	The truck is driving near a free and proper parking lot.
Minimal Guarantees	None.
Success Guarantees	The truck is parked in a parking lot at a steep hill, compatible with the above goal or the superior mission objective.
Trigger	1. The trucker stops in front of a parking lot at a steep hill.
Description (interaction sequence)	2. He or she uses the gas pedal, steering wheel, clutch, gears, brakes (see use case "use brakes") and rear mirrors to place the truck into the lot.

Table 6.6.: CRV use case "park at steep hill"

## 6.2. Approval Case: Commercial Road Vehicle

For the approval case, the unit of analysis ( $\mathcal{A}_S$ ) is a simplified *commercial road vehicle* (also CRV or truck) and its environment, including the driver and a section of road ( $\mathcal{A}_E$ ). Hazard analysis will take place for prominent use cases, functions instead of technical parts of a truck, and the risks stemming from these functions in specific operational situations (i.e. driving situations). The approval case is documented according to the procedure in Chapter 5.

### 6.2.1. Step 1: Model System and Environment

**Artefact  $\mathcal{R}$  – Use Cases and Property Assertions** For the set  $\mathcal{R}$ , knowledge about the automotive and transportation domains has been gathered from two research collaborations, one with ITK Engineering AG<sup>2</sup> (Dobi et al. 2013) and another one with BMW

<sup>2</sup>[www.itk-engineering.de](http://www.itk-engineering.de)

## 6.2. Approval Case: Commercial Road Vehicle

Use Case	Use brakes (modelled below as $Missions_E, AccBrakes$ )
Goal	G10: The trucker is able to use the brake, for example, at traffic lights and stop signs.
Scope	$\mathcal{A}_S$ ; level: primary task in $\mathcal{A}_{E use}$ ; primary actor: $\mathcal{A}_E$
Preconditions	An object gets near or onto the truck route, or a waypoint or the end of the route is reached.
Minimal Guarantees	The truck is slowing down.
Success Guarantees	The truck is correctly slowing down or coming to a stable halt.
Trigger	1. The trucker actuates the brake pedal.
Description (interaction sequence)	2. The truck decreases its speed accordingly. 3. Optional: When the truck comes to a halt, the trucker decides to activate the stop brake.

Table 6.7.: CRV use case “use brakes”, always included by use case “park at steep hill”

AG<sup>3</sup> (Gleirscher and Fuhrmann 2012, Gleirscher et al. 2014). Further data are documented in (Gleirscher 2013b). After requirements analysis,  $\mathcal{R}_f$  informally describes use cases (i.e. driving missions and situations) of a truck.  $\mathcal{R}_p$  contains informal property assertions (i.e. goals and safety-related assertions) for these use cases (cf. Table A.13).

**Step 1.1** I derived three use cases including driver tactics from  $\mathcal{R}_f$ , see “use truck” in Table 6.5, “park at steep hill” in Table 6.6 and “use brakes” in Table 6.7.

**Step 1.2** Suppose that the world is a linear area of length 100 where  $\mathcal{A}_S$  is located in and can move forward and backward. For simplified reasoning as envisaged for validation, we derive two variables for abstract physical motion, speed and position. Analogically, for the complete airbag control loop, the system boundary includes the variables impact, deformed, crashed and released. Note that these variables can be enriched and structured by an underlying entity model, for example, including *area*, *vehicle* and *airbag*.

**Artefact  $\mathcal{V}$  – Variables** According to Figure 4.1, Table 6.8 describes the set of variables for safety analysis including the channels modelling the safety-related interface of the CRV. For sake of brevity, instead of declaring each enumerative type, I provide only three examples:

$$\begin{aligned} \text{type}(\text{load}) &= \{\text{none}, \text{medium}, \text{crit}, \text{max}, \text{overload}\} \\ \text{type}(\text{speed}) &= \{0, \text{low} \equiv 1, \text{medium} \equiv 2, \text{high} \equiv 3\} \times \{\text{fwd}, \text{bwd}\} \\ \text{type}(\text{position}) &= \{x \mid x \in -49..50\} \end{aligned}$$

<sup>3</sup>[www.bmw.de](http://www.bmw.de)

## 6. Case Study

$\cap$	$\not\subset \text{mon}(\mathcal{A}_E) \cup \text{ctr}(\mathcal{A}_E)$	$\text{mon}(\mathcal{A}_E) \setminus \text{ctr}(\mathcal{A}_E)$	$\text{ctr}(\mathcal{A}_E) \setminus \text{mon}(\mathcal{A}_E)$	$\text{mon}(\mathcal{A}_E) \cap \text{ctr}(\mathcal{A}_E)$
$\text{mon}(\mathcal{A}_S) \cup \text{ctr}(\mathcal{A}_S)$	$\ddagger$	deformed : $\mathbb{B}$ , absorbedBy <sub>drv,airb</sub> : $\mathbb{B}$	at <sub>driver,seat</sub> : $\mathbb{B}$ , at <sub>obj,pillion</sub> : $\mathbb{B}$	position <sub>key</sub>
$\text{mon}(\mathcal{A}_S) \setminus \text{ctr}(\mathcal{A}_S)$	status <sub>keyslot</sub> , gas, energy	impact	(no variables identified)	position <sub>brakepedal</sub> , position <sub>clutchpedal</sub> , position <sub>gaspedal</sub> , position <sub>steerwheel</sub> , position <sub>gearlever</sub> , switch <sub>stopbrake</sub> : $\mathbb{B}$ , load, speed <sub>otherObj</sub> , position <sub>otherObj</sub> , crashed : $\mathbb{B}$ ,
$\text{ctr}(\mathcal{A}_S) \setminus \text{mon}(\mathcal{A}_S)$	$\ddagger$	speed, position <sub>reartid</sub> , released : $\mathbb{B}$ , status <sub>warnLamp</sub> : $\mathbb{B}$	$\ddagger$	$\ddagger$
$\text{mon}(\mathcal{A}_S) \cap \text{ctr}(\mathcal{A}_S)$	active : $\mathbb{B}$	(no variables identified)	repaired : $\mathbb{B}$	position

Table 6.8.: Variables in  $\mathcal{V} \setminus \mathcal{V}_m$  for safety analysis;  $\ddagger$  . . . unnecessary in this model

**Step 1.3** Applying Definition 4.1 for the aspect  $\mathbb{M}_{\text{use}}$ , the following equations provide an excerpt and simplification of the functionality of a road vehicle as elaborated in (Gleirscher and Fuhrmann 2012):

$$\begin{aligned} \text{Mission}_E &= \text{Park}_E \oplus \dots \\ \mathcal{A}_E|_{\text{use}} &= \text{Mission}_E \otimes \text{otherObjMove}_E \end{aligned} \quad (6.9)$$

$$\text{ActDeact}_S = \text{ActDeactVeh}_S \oplus \text{ActDeactEng}_S$$

$$\text{AccBrakes}_S = \text{StopBrake}_S \otimes \alpha(\text{Brakes}_S \otimes \text{Accelerates}_S)$$

$$\text{DriveMoves}_S = \text{Steers}_S \otimes \text{AccBrakes}_S \otimes \alpha(\text{Declutch}_S \otimes \text{AdjTransm}_S)$$

$$\mathcal{A}_S|_{\text{use}} \equiv \text{Trucks}_S = \text{ActDeact}_S \otimes \alpha(\text{DriveMoves}_S) \otimes \text{Airbags}_S \quad (6.10)$$

$$\mathcal{M}|_{\text{use}} = \mathcal{A}_E|_{\text{use}} \otimes \mathcal{A}_S|_{\text{use}} \quad (6.11)$$

The use case “use truck” (Table 6.5) indirectly describes parts of  $\mathcal{A}_E|_{\text{use}}$  as an interaction sequence with the aspect  $\mathcal{A}_S|_{\text{use}}$ . “Park at steep hill” (Table 6.6) exemplifies interaction of  $\text{Park}_E$  and  $\mathcal{A}_S|_{\text{use}}$ . “Use brakes” (Table 6.7) describes  $\mathcal{A}_E|_{\text{use}}$  using  $\text{AccBrakes}_S$ .

**Step 1.4** The composite levels of the hierarchy help identify abstract actions observable at the interface of a truck by co-executing its tactics and functions. For example, a truck can be moved or driven, loaded or unloaded.  $\alpha(\text{DriveMoves}_S)$ ,<sup>4</sup> shown in Figure 6.7a, contains an abstract action “move” given in three dynamics. The abstract and *complex actions*  $\text{start}^+$  and  $\text{stop}^+$  (Definition 2.12) include system actions triggered

<sup>4</sup>Abstracted via  $\alpha : \mathbb{M} \rightarrow \mathbb{M}$  from  $\text{DriveMoves}_S$ .

## 6.2. Approval Case: Commercial Road Vehicle

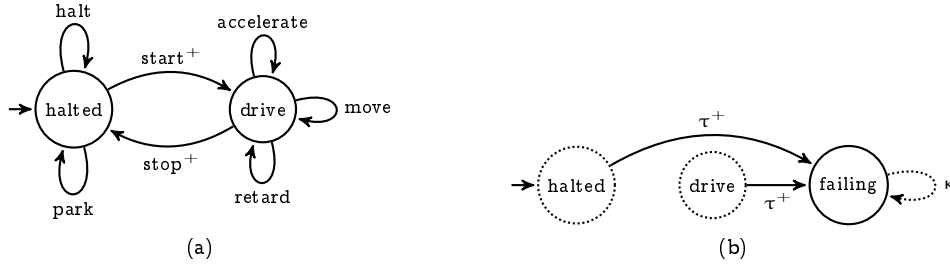


Figure 6.7.: MTSs for (a) the function  $\alpha(\text{DriveMoves})$  and (b) its defects  $\alpha(\text{DriveMoves}_f)$

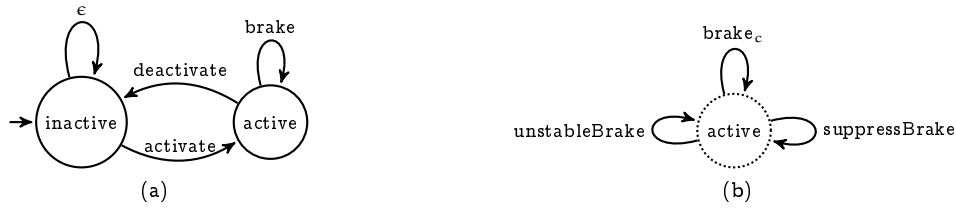


Figure 6.8.: MTSs for (a) the function  $\text{StopBrake}_S$  and (b) its defects  $\text{StopBrake}_{S_f}$

by an abstract event called `userOperation` as well as other system actions. Derived from Table 6.7, the Figures 6.8a and 6.9a show the functions  $\text{StopBrake}_S$  and  $\text{Airbag}_S$ .

**Artefact  $\mathcal{M}$  – Functions and Tactics  $\mathbb{M}_{\text{use}}$**  Table A.14 shows a cutout of the CRV actions.

**Step 1.5** In  $\mathcal{R}_p$ , we identified the safety-related assertions “*get salvaged after accident*” (G22), “*avoid accident*” (G40) and “*protect driver/passenger*” ( $\gamma_{\text{safeAirbag}}$ ).

**Artefact  $\Gamma$  – Safety-related Assertions** Table A.13 shows several goals which reflect operational situations of a CRV. These goals guide the preconditions as well as the minimal and success guarantees of the CRV use cases.

**Discussion** For Step 1.1, we can consider additional use cases and driving situations such as the use of vehicle extensions or trailers. Concerning Step 1.2, I neglected variables such as wheel rotation or engine heat. In Step 1.3, the tactics in  $\text{Missions}_E$  can be constructed from the use cases. The modelling of these tactics, however, is omitted in the example. For Step 1.4, the truck model can be extended by other vehicle functions such as a tank, a crane, door control, adaptive cruise control (Lochmann and Gleirscher 2009), pre-crash safety, a window opener, a steering wheel lock or a wiper.

## 6. Case Study

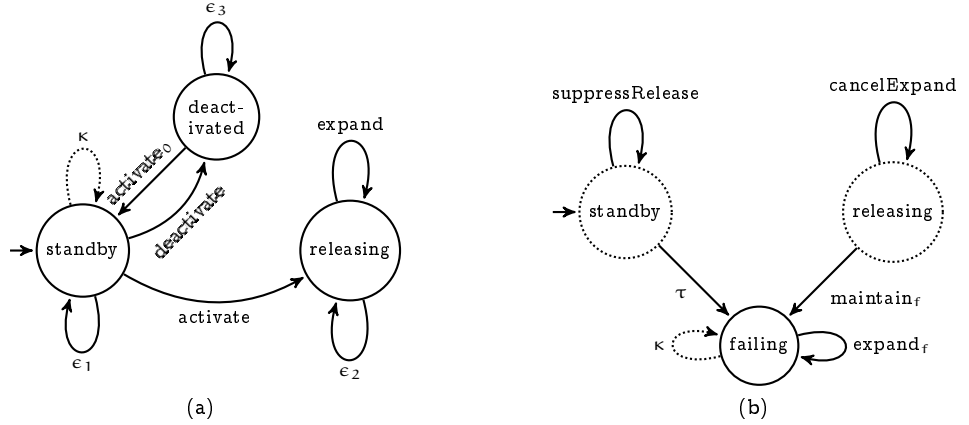


Figure 6.9.: MTSs for (a) the function  $\text{Airbag}_S$  and (b) its defects  $\text{Airbag}_{Sf}$

### 6.2.2. Step 2: Derive Defect Model

**Step 2.4**  $\alpha(\text{DriveMove}_{Sf})$  in Figure 6.7b applies the pattern “random/permanent” from Table A.2 ( $\text{fail}_1$  instantiates to  $\tau^+$ ) to model a defective mode being superimposed via  $\alpha(\text{DriveMove}_S) \oplus \alpha(\text{DriveMove}_{Sf})$ . This pattern introduces an abstract action  $\tau^+$  of the mode drive in  $\alpha(\text{DriveMove}_S)$  leading to the mode failing in  $\alpha(\text{DriveMove}_{Sf})$ .

Using the probabilistic variant of the pattern “random/transient (alternative)”, Figure 6.8b shows potential failures as a defective fragment  $\text{StopBrake}_{Sf}$  ( $\text{fail}$  instantiates to  $\text{suppressBrake}$ ,  $\text{unstableBrake}$  and  $\text{brake}_c$ ).

Figure 6.9b shows the defect model  $\text{Airbag}_{Sf}$  applying the pattern “random/permanent” twice (see Example 4.1) and the pattern “random/transient (alternative)” to obtain the defective actions  $\text{suppressRelease}$  and  $\text{cancelExpand}$ . The kind of defect encoded by these defective actions captures sensor and actuator faults.

**Artefact  $\mathcal{M}$  – Defect Model  $\mathbb{M}_{\text{fail}}$**  The identified defects are modelled in the Figures 6.7b, 6.8b and 6.9b, actions are specified in Table A.14. Based on Definition 2.13, we add the aspect  $\mathbb{M}_{\text{fail}}$  to the system as follows:

$$\begin{aligned} \text{AccBrake}_S &= (\text{StopBrake}_S \oplus \text{StopBrake}_{Sf}) \otimes \alpha(\text{Brake}_S \otimes \text{Accelerate}_S) \\ \mathcal{A}_{S|_{\text{use}, \text{fail}}} &= \text{ActDeact}_S \otimes (\alpha(\text{DriveMove}_S) \oplus \alpha(\text{DriveMove}_{Sf})) \\ &\quad \otimes (\text{Airbag}_S \oplus \text{Airbag}_{Sf}) \end{aligned} \quad (6.12)$$

$$\mathcal{M}_{S|_{\text{use}, \text{fail}}} = \mathcal{A}_{E|_{\text{use}, \text{fail}}} \otimes \mathcal{A}_{S|_{\text{use}, \text{fail}}} \quad (6.13)$$

**Discussion** As in the pilot case, I left out the modelling of side effects (Step 2.1) and the consult of reliability analyses (Sections 2.2.3 and 3.2; Step 2.2): I chose simplicity of the case study instead of a precise defect model. Nevertheless, assume that these analyses confirm  $\mathcal{A}_{S|_{\text{fail}}}$  as derived in Step 2.4, for example, by providing characteristics of the



physical action `suppressBrake`. Li (2014), for example, outlines power train control failures in road vehicles and supports hardware-in-the-loop testing as a measure to avoid potential causes of such failures. We consider  $\Sigma_0$  without defective states. Using defective initial states from Step 2.3, however, can reduce reasoning effort in Step 4.1 because of fewer and shorter runs satisfying any hazard.

### 6.2.3. Step 3: Identify Potential Mishaps

**Step 3.1** From Step 1.4, we know that the truck action `move` affects speed and position. This action has physical impact and can potentially be involved in mishaps.

**Step 3.2** Let `low`  $\equiv 1$ , `short`  $\equiv 2$  and  $|\text{speed}| \in \mathbb{N}$ . Regarding  $\alpha(\text{DriveMove}_S)$  with the action `move`, the guide word “collide” (in its second variant) from Table A.3 captures a mishap as a combination of *too small distances* and *too high relative velocities* by a state constraint

$$\phi_{\text{collide}} \equiv |\text{speed} - \text{speed}_{\text{otherObj}}| > \text{low} \wedge |\text{position} - \text{position}_{\text{otherObj}}| < \text{short}$$

For `AirbagS`, the guide words “pump”, “hit” and “distract” yield “bump of passenger into vehicle interior” ( $\phi_{\text{bump}}$ ), “driver or co-driver directly or indirectly hit by released airbag” ( $\phi_{\text{harmExp}}$ ) and “distraction of driver by released airbag.” For now, consider

$$\phi_{\text{distract}} \equiv \text{released} \geq 2 \wedge m_{\alpha(\text{DriveMove}_S)} = \text{drive} \wedge \text{at}_{\text{driver,seat}}$$

**Step 3.3** The conduct of use case “park at steep hill” (Table 6.6) yields an operational situation “*the truck is standing in a steep parking lot and the stop brake is activated*” that is modelled via the state constraint  $\sigma_{\text{steepPLot}}$ . For sake of simplicity, we omitted further variables such as the constitution of the road surface (e.g. temperature, ice, water) or the environment (e.g. wind, gravity, nearby objects, road downgrade and route section), and the physical state of the vehicle (e.g. age, maintenance).  $\sigma_{\text{packedVeh}}$  captures the situation “*a bottle or a child safety seat is positioned in front of the co-driver’s panel or at the pillion.*” Based on Definition 4.2, we define

$$\begin{aligned} \sigma_{\text{steepPLot}} &\equiv m_{\text{StopBrake}_S} = \text{active} \wedge \text{load} = \text{max} \wedge \dots, \\ \sigma_{\text{drive}} &\equiv m_{\alpha(\text{DriveMove}_S)} = \text{drive} \wedge \text{at}_{\text{driver,seat}} \wedge \dots, \\ \sigma_{\text{packedVeh}} &\equiv \text{at}_{\text{obj,pillion}} \wedge \text{load} = \text{max} \wedge \dots, \\ \sigma_{\text{unmanned}} &\equiv \neg \text{at}_{\text{driver,seat}} \end{aligned}$$

**Artefact  $\Phi$  – Mishaps** The informal description is shown in Table A.13, the results of the Steps 3.2 and 3.3 in Table 6.9. The set of variables has been extended by  $\text{at}_{\text{driver,seat}}$ ,  $\text{speed}_{\text{otherObj}}$  and  $\text{position}_{\text{otherObj}}$ , as recorded in Table 6.8.

## 6. Case Study

**Discussion** The identification of operational situations (Step 3.3) is further discussed by Dobi et al. (2013). Additional situations are motivated by the goals G1–27 in Table A.13. Defining operational situations can be seen as an alternative to exhaustive environment modelling. Nevertheless, I can recommend combining the modelling of both environment behaviour and operational situations.

### 6.2.4. Step 4: Assess Causal Factors

**Step 4.1** Tool supported reachability analysis for the mishaps  $\phi_{\text{collide}}$  and  $\phi_{\text{distract}}$  and the operational situations  $\sigma_{\text{steepPLot}}$  and  $\sigma_{\text{packedVeh}}$  provides a set of runs.

**Steps 4.2 to 4.5** These runs make it possible to *search backwards from mishaps to hazardous, defective actions, states, events and modes, starting from abstract actions, continuing with composite actions and down to the level of actions:*

*Search for causal factors of  $\phi_{\text{collide}}$ :* Starting from  $\phi_{\text{collide}}$ , we might look for the pattern “unexpected execution of move” from Table A.1. A search for the variation “unattended execution of move” can look for causal factors in runs, that is, effects of move happening without foreseen action of the trucker:

$$\begin{aligned} \chi_{\text{unattMove}} \equiv & \bar{F}^{\leq 10s} (\bar{X}(\neg \text{userOperation} \vee \neg \text{move.pre}) \\ & \wedge (\text{move.post} \vee m_{\alpha(\text{DriveMove}_s)} = \text{drive})) \end{aligned} \quad (6.14)$$

We regard move without  $M_{\text{save}}$  and derive hazards, that is, possibilities of how and when move is hazardously performable according to  $\chi_{\text{unattMove}}$ . Figure 6.7 suggests the derivation of specified, hazardous and safe refinements of the abstract actions  $\text{start}^+$ ,  $\text{stop}^+$  and  $\tau^+$  using the MTSs comprising  $\alpha(\text{DriveMove}_s)$ . Formula (6.14) is refined by  $\text{move.pre} \equiv -49 < \text{position} - \text{speed} < 55$  and, depending on  $\mathcal{A}_E$ , we can define

$$\text{userOperation} \equiv \text{position}_{\text{gaspedal}} = \text{pressed} \vee \text{switch}_{\text{stopbrake}} \vee \dots$$

The analysis of  $\alpha(\text{DriveMove}_s)$ , containing the action move, at a lower level of composition shows, for example: the mode active and the action brake of  $\text{StopBrake}_s$  are responsible to maintain speed = 0 or to contribute to the action park. Thus, the hierarchy suggests the refined use of the guide word “unexpected suppression of brake”:

$$\begin{aligned} \text{cf}_{\text{supprBrake}} \equiv & \bar{F}^{\leq 10s} (\bar{X}(m_{\text{StopBrake}_s} = \text{active} \wedge \text{brake.pre}) \\ & \wedge \neg(\text{brake.post} \wedge m_{\text{StopBrake}_s} = \text{active})) \\ \equiv & \bar{F}^{\leq 10s} (\bar{X}(m_{\text{StopBrake}_s} = \text{active} \wedge \top) \\ & \wedge \neg(((\bar{X}\top) \wedge \text{speed} = 0) \wedge m_{\text{StopBrake}_s} = \text{active})) \end{aligned} \quad (6.15)$$

Hence, we are looking for runs such that the actions  $\text{suppressBrake}$  and  $\text{unstableBrake}$  of  $\text{StopBrake}_{sf}$  (Figure 6.8b) contribute to  $\chi_{\text{unattMove}}$  and potentially to mishap  $\phi_{\text{collide}}$ , for example, in the operational situation  $\sigma_{\text{steepPLot}} \wedge \sigma_{\text{unmanned}}$ .

## 6.2. Approval Case: Commercial Road Vehicle

To perform a further search step, we can use guide words such as “maloperation of AccBrakes” (e.g. acceleration in spite of objects in the way), “maloperation of StopBrakes” (e.g. forgetting to activate the stop brake), or “maloperation of brake” (e.g. because of tired driver). Finally, this backward and descending search for a combination of causal factors leads to the refined hazard

$$\chi'_{\text{unattMove}} \equiv \bar{F}^{\leq 10s} \text{userOperation} \wedge (\text{cf}_{\text{supprBrake}} \vee \dots) \quad (6.16)$$

Note that the absence of userOperation is now included using “e not given”. As we are unable to assume causality in the sense of  $\text{cf}_{\text{supprBrake}} \rightarrow \chi_{\text{unattMove}} \rightarrow \phi_{\text{collide}}$ , we proceed to the check of Definition 4.8 in  $\mathcal{M}$ :

$$\text{EF}(\chi'_{\text{unattMove}} \rightarrow P_{>0.01}[\text{XF}\phi_{\text{collide}}])$$

By Figure 4.4, we declare  $\text{cf}_{\text{supprBrake}} \in \mathcal{H}_{\text{op}}$  because suppressBrake, identified in Step 2.4, contributes to  $\chi'_{\text{unattMove}}$ .

*Search for causal factors of  $\phi_{\text{distract}}$ :* As shown in Example 4.2 by

$$\chi_{\text{unexpExp}} \equiv \bar{F}^{\leq 100\text{ms}} \text{crashed} \wedge \bar{F}^{\leq 20\text{ms}} (\text{m}_{\text{Airbags}} = \text{failing})$$

a release during normal drive ( $\sigma_{\text{drive}}$ ) is considered a causal factor for distraction. Regarding  $\phi_{\text{distract}}$  with  $\text{Airbags} \oplus \text{Airbags}_{\text{sf}}$ , the application of the guide word “unexpected execution of expand” from Table A.1 yields the causal factor

$$\begin{aligned} \chi'_{\text{unexpExp}} &\equiv \bar{F}^{\leq 100\text{ms}} (\bar{X}(\neg \text{enabl} \vee \neg \text{expand.pre}) \\ &\quad \wedge (\text{expand.post} \vee \text{m}_{\text{Airbags}} = \text{releasing})) \\ &\equiv \bar{F}^{\leq 100\text{ms}} (\bar{X}(\neg (\text{m}_{\text{Airbags}} = \text{releasing} \wedge \text{crashed}) \vee \text{gas} = \text{empty}) \\ &\quad \wedge ((\exists x, y : \bar{X}(\text{released} = x \wedge \text{gas} = y) \\ &\quad \quad \wedge \text{released} = x + 1 \wedge \text{gas} = y - 1) \vee \text{m}_{\text{Airbags}} = \text{releasing})) \end{aligned}$$

Note that for the search pattern  $\chi'_{\text{unexpExp}}$  to work, we do not require  $\text{Airbags}_{\text{sf}}$ . Now, we can proceed to the check of Definition 4.8 in  $\mathcal{M}$ :

$$\text{EF}(\chi'_{\text{unexpExp}} \rightarrow P_{>0.05}[\text{XF}\phi_{\text{distract}}])$$

Applying the event guide word “e happened”, we can directly specify a hazard for  $\phi_{\text{collide}}$  based on a previously identified mishap

$$\chi_{\text{distract}} \equiv \bar{F}^{\leq 5s} \phi_{\text{distract}}$$

and check

$$\text{EF}(\chi_{\text{distract}} \rightarrow P_{>0.05}[\text{XF}\phi_{\text{collide}}])$$

**Artefact  $\mathcal{H}$  – Hazards** The identified hazards enhance Table 6.9 which then shows the combination of mishaps, operational situations and hazards.

## 6. Case Study

Step 1		Steps 3 and 4		Step 4.6				Step 5.3
$\mathcal{M}$	Mishap $\in \Phi$	Oper. Situation	Hazard $\in \mathcal{H}$	S	W	A	G	IC
$\alpha(\text{DriveMoves})$	$\phi_{\text{collide}}$	$\sigma_{\text{steepPLot}}$	$\chi_{\text{unattMove}}$	h	m	h	m	high
$\alpha(\text{DriveMoves})$	$\phi_{\text{collide}}$	$\sigma_{\text{steepPLot}}$	$\chi'_{\text{unattMove}}$	h	m	h	m	high
$\alpha(\text{DriveMoves})$	$\phi_{\text{collide}}$	$\sigma_{\text{steepPLot}} \wedge \sigma_{\text{unmanned}}$	$\chi'_{\text{unattMove}}$	h	m	h	m	high
$\alpha(\text{DriveMoves})$	$\phi_{\text{collide}}$	$\sigma_{\text{drive}}$	$\chi_{\text{distract}}$	h	l	h	m	high
Airbags	$\phi_{\text{distract}}$	$\sigma_{\text{drive}}$	$\chi'_{\text{unexpExp}}$	h	l	h	l	high
Airbags	$\phi_{\text{bump}}$	$\sigma_{\text{drive}}$	$\chi_{\text{supprExp}}$	h	l	h	l	high
Airbags	$\phi_{\text{harmExp}}$	$\sigma_{\text{drive}} \wedge \sigma_{\text{packedVeh}}$	$\chi_{\text{harmExp}}$	m	l	h	l	medium

Table 6.9.: Hazard assessment similar to DIN 19250, IEC 61508 and ISO 26262; S... severity, W... probability, A... exposure, G... detectability and controllability, IC... integrity class, l/m/h... low/medium/high; cells in grey indicate related content

**Step 4.6** We can now assess runs according to Figure 4.4. The estimation of hazard characteristics, as described on page 56, to state  $\chi'_{\text{unattMove}} \in \mathcal{H}$  can be made by automated probabilistic reasoning: for Airbags, we might calculate the probabilities of “*crash after airbag release while drive*” ( $P_{=?}[\chi'_{\text{unexpExp}}]$ ) and “*improper release after crash while drive*” ( $P_{=?}[\chi_{\text{supprExp}} \vee \chi_{\text{harmExp}}]$ ).

**Artefact  $\mathcal{H}$  – Classified Hazards** The results of mishap identification (Step 3) and hazard assessment (Step 4) are depicted in Table 6.9. Moreover, this table exemplifies the estimation of the hazard characteristics.

**Discussion** Beginning with a mishap, causal factor search tracks effects of hazardous actions of the environment and the system. Other causal factors for  $\phi_{\text{collide}}$  such as the event *improper wheel momentum* (vehicle torsion), and the abstract states *high speed-distance ratio* and *sliding* have been disregarded. Nevertheless, such causal factors would be subject of analysis in practical settings.

By repeating the Steps 4.2 to 4.5, a *search for causal factors* of  $\phi_{\text{bump}}$  can involve the guide words “unexpected suppression of expand,” “missed, early or late released  $\neq$  no,” or “wrong timing of released  $\neq$  no” ( $\chi_{\text{supprExp}}$ ). This repetition, however, seems to refuse new insight into the method and is, hence, omitted. For any action  $a$  referred to in the formulae of Steps 4.2 to 4.5, the semantics on page 19 and Definition 2.15 guide the transform of the function  $\text{a.post} : \mathcal{E}|_{\text{ctr}(\mathcal{M})}$  into an action effect formula.

As this case study focuses on modelling and the method, I left out the automation of Step 4.6. To perform this step, I can recommend, for example, probabilistic model checkers as mentioned in Section 2.2.1.

### 6.2.5. Step 5: Specify Safety Goals

**Steps 5.1 and 5.2** *Specifying and decomposing a safety goal for  $\phi_{\text{collide}}$* : As with Step 4.2, we could first express a safety goal based on abstract modes and actions of  $\alpha(\text{DriveMoves})$  and the hazard  $\chi_{\text{unattMove}}$ . Having the refined hazard  $\chi'_{\text{unattMove}}$ , we can immediately descent into the modes and actions, and derive an *avoidance goal* “avoid unattended driveaway” according to Equation (4.3):

$$\begin{aligned}
\gamma_{\text{treat.collide}} &\equiv P_{\geq 0.99}[\mathbf{G}\neg\chi'_{\text{unattMove}}] \\
&\equiv P_{\geq 0.99}[\mathbf{G}\neg(\neg\bar{\mathbf{F}}^{\leq 10s} \text{userOperation} \\
&\quad \wedge (\bar{\mathbf{F}}^{\leq 10s} (\bar{\mathbf{X}}(\text{m}_{\text{StopBrakes}} = \text{active} \wedge \top) \\
&\quad \quad \wedge \neg((\bar{\mathbf{X}}\top) \wedge \text{speed} = 0) \wedge \text{m}_{\text{StopBrakes}} = \text{active})) \\
&\quad \vee \dots))] \tag{6.17}
\end{aligned}$$

$\gamma_{\text{treat.collide}}$  divides behaviour fulfilling  $\chi'_{\text{unattMove}}$  into acceptable (e.g. no driver input given, regular maintenance) and unacceptable (e.g. irregular maintenance) runs. Following Table 6.9 and to specify a responsibility relationship, we assign  $\gamma_{\text{treat.collide}}$  to  $\alpha(\text{DriveMoves})$  and the overall  $\mathcal{A}_E$ . For  $\text{StopBrakes}$ , we derive the safety requirement  $(\text{As}, \text{Gr})_{\text{treat.collide}}$  with

$$\begin{aligned}
\text{As} &\equiv P_{< x}[\text{AsBody}] \equiv \top \\
\text{Gr} &\equiv P_{\geq y}[\text{GrBody}] \equiv \gamma_{\text{treat.collide}}
\end{aligned}$$

The ultimate weakening of  $\text{As}$  leaves all responsibility to treat  $\phi_{\text{collide}}$  over for  $\alpha(\text{DriveMoves})$ . Alternatively, we can specify a *mitigation goal* such that the trucker is required to not leave alone the truck for longer than a certain period if  $\text{load} = \text{max}$ . This assumption would make many occurrences of  $\chi'_{\text{unattMove}}$  more probably controllable to reduce the risk of  $\phi_{\text{collide}}$ .

*Specifying and decomposing a safety goal for  $\phi_{\text{distract}}$* : As opposed the previous safety goal, the uncontrollability of hazard  $\chi'_{\text{unexpExp}}$  by  $\mathcal{A}_E$  requires the choice of an *avoidance goal* (Equation (4.3)) instead of a mitigation goal:

$$\begin{aligned}
\gamma_{\text{treat.distract}} &\equiv P_{\geq 0.999}[\mathbf{G}\neg\chi'_{\text{unexpExp}}] \\
&\equiv P_{\geq 0.999}[\mathbf{G}\neg(\bar{\mathbf{F}}^{\leq 100\text{ms}} (\bar{\mathbf{X}}(\neg(\text{m}_{\text{Airbags}} = \text{releasing} \wedge \text{crashed}) \\
&\quad \vee \text{gas} = \text{empty})) \\
&\quad \wedge ((\exists x, y : \bar{\mathbf{X}}(\text{released} = x \wedge \text{gas} = y) \\
&\quad \quad \wedge \text{released} = x + 1 \wedge \text{gas} = y - 1) \\
&\quad \vee \text{m}_{\text{Airbags}} = \text{releasing})))]
\end{aligned}$$

We can derive a safety requirement  $(\text{As}, \text{Gr})_{\text{treat.distract}}$  with an ultimately weakened assumption and a guarantee as follows:

$$\begin{aligned}
\text{As} &\equiv P_{< x}[\text{AsBody}] \equiv \top \\
\text{Gr} &\equiv P_{\geq y}[\text{GrBody}] \equiv \gamma_{\text{treat.distract}}
\end{aligned}$$

## 6. Case Study

*Specifying and decomposing a safety goal for  $\phi_{\text{bump}}$ :* Here, we again formulate an *avoidance goal* based on hazard  $\chi_{\text{supprExp}}$ :

$$\gamma_{\text{treat.bump}} \equiv \text{P}_{\geq 0.999}[\mathbf{G}\neg\chi_{\text{supprExp}}]$$

We can impose a responsibility relationship: Airbag<sub>S</sub> detects the mode failing and turns inactive, indicated to the operator by a variable  $\text{status}_{\text{warnLamp}}$ .  $\mathcal{A}_E$  takes care of repairing the airbag mechanism and returns it to standby. The derivation of a safety requirement  $(\text{As}, \text{Gr})_{\text{treat.bump}}$  (with e.g.  $\text{As} \equiv \text{status}_{\text{warnLamp}} \rightarrow \text{Frepaired}$ ) works analogical to  $(\text{As}, \text{Gr})_{\text{treat.distract}}$  and is omitted.

**Step 5.3** Having clarified responsibility, in Table 6.9 column IC, we can assign the integrity class “high”. This class might, for example, be associated with ASIL D in ISO Std. 26262 (2011).

**Step 5.4** The joint achievement of safety goals requires the check for consistency of  $\hat{\Gamma} \not\perp$ . This expression expands to

$$\gamma_{\text{treat.collide}} \wedge \gamma_{\text{treat.distract}} \wedge \gamma_{\text{treat.bump}} \not\perp$$

**Artefact  $\Gamma$  – Safety Goals, A/G Pairs** The informal description of  $\Gamma$  is imported from  $\mathcal{R}_p$  and shown in Table A.13, the formal description has been discussed in the Steps 1.5, 3.2, 4.5, 5.1 and 5.2.

**Discussion** The safety goal “protect any persons, goods and the environment” is motivated by Table 6.5 and requires “no collision” ( $\neg\phi_{\text{collide}}$ ). Returning to Section 1.2,  $\gamma_{\text{treat.distract}}$  and  $\gamma_{\text{treat.bump}}$  suggest a safety goal for each practicable Airbag<sub>S</sub> by stating “the front airbag must be released iff an actual front crash occurs and be deactivated if a system defect occurs.” This goal can be asserted as

$$\gamma_{\text{safeAirbag}} \equiv \text{P}_{\geq 0.9999}[\mathbf{G}(\text{crashed} \leftrightarrow \mathbf{F}^{<350\text{ms}} \text{absorbedBy}_{\text{drv}, \text{airb}})]$$

As this case study focuses on modelling and method, I omitted automation of Step 5.4. I recommend automation to perform this step, for example, by theorem provers and satisfiability solvers as mentioned in Section 2.2.1.

### 6.2.6. Step 6: Plan and Design Safety Measures

In this step, measures against the identified mishaps and hazards are specified.

**Step 6.1** *Preventive measure fulfilling  $\gamma_{\text{treat.collide}}$ :* The integrity class “high” for StopBrake<sub>S</sub> requires the treatment of the hazard  $\chi'_{\text{unattMove}}$ . Guided by the A/G pair  $(\text{As}, \text{Gr})_{\text{treat.collide}}$ , the pattern “fail-safe actions” in the variant “fail-operational/indeterminisation” from Table A.4 was applied to the functional actions  $\text{suppressBrake}$  and  $\text{unstableBrake}$ . The fail-safe actions  $\text{detDefBrk}$  and  $\text{safeBrake}$  of the

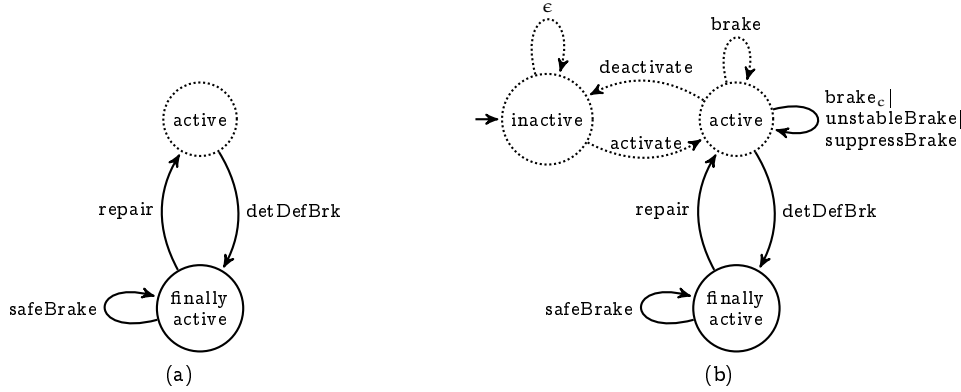


Figure 6.10.: The safety measure  $\text{StopBrake}_{S_s}$  (a) and the MTS  $\text{StopBrake}_S \oplus \text{StopBrake}_{S_f} \oplus \text{StopBrake}_{S_s}$  (b) according to  $\mathcal{A}_S$

function  $\text{StopBrake}_{S_s}$  (Figure 6.10a) represent a fail-operational mechanism suited to mask unacceptable executions of `suppressBrake` or `unstableBrake`. Indeterminacy is finally reduced by action priorities. The two fail-safe actions result in a safer version of  $\text{StopBrake}_S$  (Figure 6.10b).

We need to realise IC “high” for  $\text{StopBrake}_{S_s}$ . Defect analysis (Step 2 based on Section 3.2) would explain technical causes why the action `suppressBrake` can happen.  $\text{StopBrake}_{S_s}$  has to undergo V&V towards reliable handling of these causes to avoid  $\chi_{\text{unexpMove}}$ .

*Passive measure fulfilling  $\gamma_{\text{treat.collide}}$ :* Regarding T2 (cf. page 36),  $\text{Truck}_S$  keeps a passenger protected from injury out of collisions. From an earlier iteration,  $\text{Airbags}_S$  is given as a safety-related subsystem for reducing driver or passenger injury during a collision and, thus, treating  $\phi_{\text{collide}}$  by harm alleviation. Moreover, the treatment of  $\phi_{\text{distract}}$  reduced another causal factor of  $\phi_{\text{collide}}$  as shown in Table 6.9.

Regarding T1 and  $\phi_{\text{distract}}$ ,  $\text{Airbags}_S$  was assigned the integrity class “high” which has to be verified for the airbag device developed later.  $(\text{As}, \text{Gr})_{\text{treat.distract}}$  and  $(\text{As}, \text{Gr})_{\text{treat.bump}}$  shift responsibility to  $\text{Airbags}_S$  and suggest two measures (a,b) specified as follows:

*Preventive measure fulfilling  $\gamma_{\text{treat.distract}}$  (a):*  $\text{Airbags}_S$  is considered safe if its sensors estimate correct (i.e. sufficiently representing the geometry of the area and the direction of each vehicle collision event) and reliable (i.e. only and exactly detecting actual collision events in the detectable range) states of the world. To improve airbag safety, Figure 6.11 shows one application of the pattern “*repair action*” (see  $\text{repair}_1$ ) and two applications of the pattern “*fail-safe actions/fail-silent*” (see  $\text{fs}_\tau$ ,  $\text{fs}_{mf}$ ,  $\text{fs}_2$ , shutdown and  $\epsilon$ ). To treat  $\chi'_{\text{unexpExp}}$ ,  $\text{Airbags}_S$  contributes to the reliability property “unexpected release below a certain upper bound of probability.”  $\text{Airbags}_S$  reduces any unexpected release before and whenever a defective mode of  $\text{Airbags}_f$  becomes active. Example 2.1 shows a less complete treatment of this hazard.

## 6. Case Study

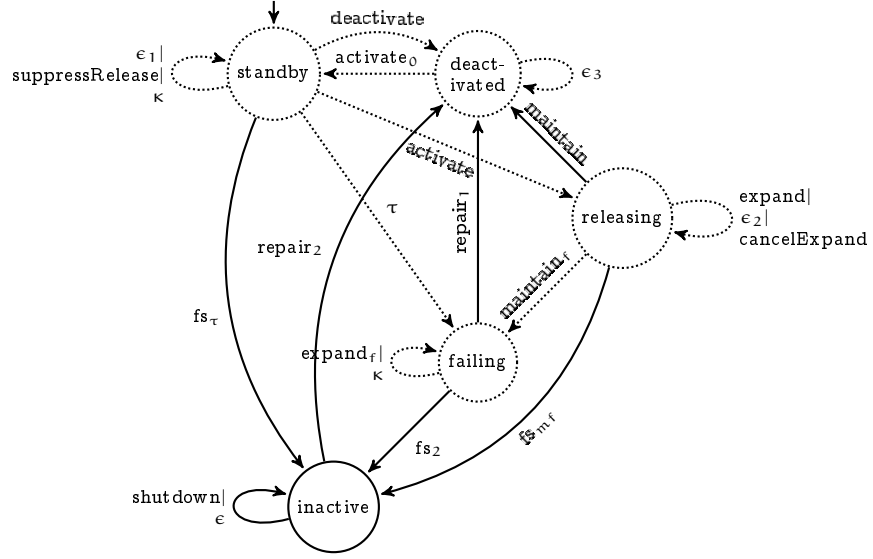


Figure 6.11.: The safety measures  $\text{Airbag}_{S_s}$  (solid) as a fragment of the MTS  $\text{Airbag}_S \oplus \text{Airbag}_{S_f} \oplus \text{Airbag}_{S_s}$  according to  $\mathcal{A}_S$

$\text{Airbag}_{S_s}$  improves the safety measure shown in Example 4.5 by applying the *deactivation* variant: The actions  $\text{repair}_1$  and  $\text{repair}_2$  lead to the mode deactivated which indicates a defective airbag with the need to visit a car repair shop soon. These two actions are triggered by a repaired event issued by  $\mathcal{A}_E$ .

*Preventive measure fulfilling*  $\gamma_{\text{treat.bump}}$  (b):  $\text{Airbag}_S$  is considered correct if it keeps passenger shock below a certain threshold whenever a collision occurs. By  $\text{Airbag}_{S_s}$ , we apply the pattern “maintenance/repair actions.” Here,  $m_u$  instantiates to standby,  $m'_u$  to releasing, and  $ma/re$  to maintain, see Figure 6.11. To overcome occurrences of  $\phi_{\text{bump}}$ , we assume the actions maintain,  $\text{repair}_1$  and  $\text{repair}_2$  to be timely triggered through a repaired event issued by  $\mathcal{A}_E$ . Note that, adapted from Example 2.1, the action  $\text{maintain}_f$  has already been included in Figure 6.9b. This action can be seen as a result of redoing Step 2 for  $\text{Airbag}_S \oplus \text{Airbag}_{S_f} \oplus \text{Airbag}_{S_s}$ .

**Artefact  $\mathcal{M}$  – Safety Measures  $\mathbb{M}_{\text{save}}$**  The safety fragment  $\text{StopBrake}_{S_s}$  is shown in Figure 6.10,  $\text{Airbag}_{S_s}$  in Figure 6.11. Finally,  $\text{Airbag}_{S_s}$  specifies reliable crash, equipment and environment diagnosis. Actions are specified in Table A.14. Based on Defini-



## 6.2. Approval Case: Commercial Road Vehicle

Variables	...	n	...	...	n + j	...	m	→
impact	F	T	T	F	F	F	F	...
deformed	0	2	10	10	10	10	10	...
crashed	F	F	F	T	T	T	T	...
released	F	F	F	F	T	T	T	...

Table 6.10.: A run of  $\mathcal{M}$  which models the safety goal  $\gamma_{\text{safeAirbag}}$

tion 4.1, we add the aspect  $\mathbb{M}_{\text{save}}$  to both agents as follows:

$$\begin{aligned} \text{AccBrakes}_S &= (\text{StopBrakes}_S \oplus \text{StopBrakes}_{Sf} \oplus \text{StopBrakes}_{Ss}) \\ &\otimes \alpha(\text{Brakes}_S \otimes \text{Accelerates}_S) \end{aligned} \quad (6.18)$$

$$\begin{aligned} \mathcal{A}_S &= \text{ActDeacts}_S \otimes (\alpha(\text{DriveMoves}_S) \oplus \alpha(\text{DriveMoves}_{Sf})) \\ &\otimes (\text{Airbag}_S \oplus \text{Airbag}_{Sf} \oplus \text{Airbag}_{Ss}) \end{aligned} \quad (6.19)$$

$$\mathcal{M} = \mathcal{A}_E \otimes \mathcal{A}_S \quad (6.20)$$

**Step 6.2** For sake of simplicity, we omit the identification and resolution of misperceptions of responsibility which may arise from Step 5.2.

**Step 6.3** Table 6.10 shows a run for *safety goal*  $\gamma_{\text{safeAirbag}}$  in Table A.13. Taking an operational situation  $\sigma_\alpha$  to define  $\Sigma_0 = \{\sigma \mid \sigma_\alpha(\sigma)\}$  according to Definition 2.6, this step requires a bounded check of  $\mathcal{M} \models \widehat{\Gamma}$ .

**Discussion**  $\text{Airbag}_S$  is active in the harm state. Hence,  $\text{Airbag}_S$  counts as a passive safety measure (Section 4.6.1) independent of whether it is included in  $\mathbb{M}_{\text{use}}$  in Step 1 or not. Moreover, only the changes applied to  $\text{Airbag}_S$  in the current iteration are subject to validation. Finally,  $\mathcal{A}_S$  constrains the behaviour of  $\text{Airbag}_S$  to operational situations where the airbag must be released and where it is prohibited doing so. Until now, there are no safety measures which technically compensate the actions  $\text{suppressRelease}$  and  $\text{cancelExpand}$  as causal factors for  $\phi_{\text{bump}}$ . A treatment at technological level, not subject of investigation, can reduce the probability of performance of these two actions. The instruction of drivers to avoid child safety seats or bottles in the release area of the airbag would be a safety measure for  $\phi_{\text{harmExp}}$ . Further analysis can result in additional assumptions for safe airbag performance.

$\text{DriveMoves}_S$  as a function and  $\text{AccBrakes}_S$  as a sub-function thereof, correctly performed by  $\mathcal{A}_S$ , can be modified by an electronic stability programme, pre-crash safety or anti-blocking system to treat further mishaps. The sub-function  $\text{Steers}_S$  technically enables the integration of a safety measure to directly intercept interactions between driver, vehicle and environment (Heißing and Ersoy 2007).

This section argues for answers to **RQ1** and **RQ2** (cf. Section 1.1) by reflecting on the case studies. From these observations, strengths and limitations are identified. After providing hints, the discussion closes with an outlook on further work.

### Contents

7.1	Evaluation of the Case Study . . . . .	104
7.2	Improvements on Related Work . . . . .	105
7.3	Some Limitations of the Approach . . . . .	108
7.4	Challenges and Hints . . . . .	109
7.5	Conclusions . . . . .	112
7.6	Further Work . . . . .	114

## 7.1. Evaluation of the Case Study

The case study aims to provide insight into the capabilities of the proposed method. This insight shows to a sufficient extent that the research goal “support safety-oriented requirements validation” (**RG**) can be achieved, and its subordinate questions “How can a technical system be modelled for hazard analysis?” (**RQ1**) and “How can hazards be identified and a safe specification be derived?” (**RQ2**) can be answered.

**Note on Case Representation** Use cases are documented using a simplified version of the template according to Cockburn (2000). The visualisation of hierarchical decomposition is motivated by use case diagrams (e.g. Friedenthal et al. 2008) and models for human-machine interface designs (e.g. Paternò et al. 1997).

**Conclusions from the ATM Case** The *main objective of the pilot case* in Section 6.1 was to show the modelling capabilities of the approach, answering **RQ1**. For **RQ1.1**, the

## 7.2. Improvements on Related Work

pilot case indicates a transition from a system model used for requirements analysis and design to a model for safety analysis. Such a transition promises reuse of existing models and design knowledge. For the ATM, the environment model was enriched to capture complex behaviour of the world. The behavioural model allows actions partially being ordered by modes. Mode channels capture dependencies between concurrent functions. Modes and dependencies can reduce complexity in the search for causal factors. For RQ1.2, the pilot case shows the reasonably abstract capture of control and display elements (i.e. the human-machine interface), and other points of physical interaction (i.e. passive and moving mechanical parts, side effects).

**Conclusions from the CRV Case** The *main objective of the approval case* in Section 6.2 was to show the safety analysis capabilities of the approach, answering RQ1 and RQ2.

For RQ1.2, we can see that the safety-related system boundary can hide sensors and actuators. Nevertheless, the defect model is used to capture sensing and actuation faults, and other system defects.

For RQ2.1, the approval case demonstrated various ways of how to attach a defect model to the specification. For RQ2.2, Section 6.2 shows in several steps how guide words can be used to derive mishaps, hazards and, finally, safety goals and A/G pairs as constraints for safe realisations. For RQ2.3, the approval case argues how to use the system boundary to rewrite safety goals as responsibility relationships among behaviour of the environment and the system. A *safety measure in the system* should reduce hazardous reactions and improve controllability of unexpected stimuli. A *safety measure in the environment* should improve controllability of hazardous reactions and reduce unexpected stimuli. For RQ2.4, the results of hazard analysis provide a stop criterion for stating the completeness of safety measures.

## 7.2. Improvements on Related Work

Below, I outline several strengths of the proposed method in comparison with the related work as discussed in Section 3.4:

**Safety-oriented Requirements Validation** The present work aims at requirements validation including the identification of hazardous defects and their treatment through safety measures. The approach supports *a-priori, predictive and constructive modelling* to elicit hazards from guessing potential mishaps but also *a-posteriori and empirical modelling* of occurred hazards and mishaps (see, e.g. Hopkins 2000). Validation according to Definition 3.1 can use system models without requiring a design or realisation. This is an improvement on many of the works listed on page 38f.

**Behavioural Abstraction** The present approach focuses behavioural modelling and supports composition (cf. Definition 2.13). It uses abstraction, mode transition systems and behavioural properties to capture causal factors, hazards and mishaps. The chosen abstractions aim at the separation of functional concerns, scalability and reuse as

## 7. Discussion

opposed to reflecting system structure. Behavioural abstraction enables investigating defective modes of  $\mathcal{A}_S$  and  $\mathcal{A}_E$ , and isolating causal factors. As described on page 42f, safety analysis based on structural models regards failure modes for each component to analyse hazardous linear causal chains. The present method makes structural approaches helpful whenever the defect model has to be made less conservative or more precise. Anyway, structural and behavioural models should be considered in combination. In summary, the semantics of  $\mathcal{M}$  enables automation in *model validation* and testing of realisations.

**Defect Taxonomy and Representation** The taxonomy proposed in Section 4.2.1 adopts kinds of defects known from the literature<sup>1</sup> to support the evaluation of effectiveness of safety measures. MTSs are expressive enough to describe safety-related defects as characterised in Section 4.2.2. First, the model helps predict the range of failures and their impacts as opposed to localising faults. Second, aside from these kinds of defects, and the actual and ideal system views (Section 2.4), the approach distinguishes the views “actually specified and modelled” and “safely specified and modelled” to define *specification defects*. Hence, Section 4.2 can be seen as a complement to work on fault diagnosis and localisation as discussed in the Sections 2.2.3 and 3.4.1.

In the approach of Mannering et al. (2007), as mentioned on page 44, failures are perceived as hazards whereas in the present approach, hazards and failures can be unrelated. The proposal of Mannering et al. lacks modelling concepts in this respect. Hall and Silva (2008) characterise defects in a way as described in Section 4.2.2: defects and hazards are deviations of an actual specification from a safe specification (cf. page 48). The authors, however, disregard behavioural properties, transition systems and a defect taxonomy whereas the present work provides guidance for hazard treatment.

**Environment Modelling and Non-linear Causative Reasoning** The approach puts emphasis on the creation of an environment model to consider environment defects (e.g. easily violable assumptions, maloperation, misperceptions of responsibility). It leverages the *human operator viewpoint* of systems with repetitive or tedious operational procedures (e.g. consumer products, private cars, commercial road vehicles), or with comprehensive automated diagnosis equipment (e.g. in power and process plants, planes, trains, vessels). Such systems demand the study of *hazardous maloperation* in specified or defective situations beyond the improvement of their reliability. The chosen system boundary, as outlined in Figure 4.1, supports such studies.

The approaches described on page 43 address the notion of behavioural safety, though only few of them regard formal behavioural modelling. I may put further notes on more recent research in this context: Property defects such as misperceptions of responsibility (cf. page 53, Section 4.5 and Table 4.3) have also been reported as fallacies of property specification by Dasgupta (2006). D’Ippolito et al. (2011) neither discuss maloperation to be encoded in the assumptions of a specified controller nor do they

---

<sup>1</sup>See, e.g. Börzsök 2011, Hummer et al. 2012, Leveson 2012, Mehrpouyan 2011, Pock 2012, Stålhane et al. 2012.

provide guidance. Whereas work devoted to the viewpoint of Leveson (2012) discusses safety in an organisational and societal context, the present work aims on a technical basis for discussions about the degree of freedom from hazards. The framework of Section 4.1, part of this basis, applies a rigorous pattern-based validation procedure and probabilistic reasoning to identify and control hazardous causalities.

**Pattern-based Safety Engineering Guidance** As mentioned above, I put emphasis on providing a pattern-based analysis and treatment method (cf. Appendix A.1): Step 4 (Section 5.2.2) adopts the idea of van Lamsweerde (2009) who uses goal graphs to elicit hazards. As an alternative to goal graphs for validation, the present method helps validate properties and MTSs at once by using the treatments discussed in Section 4.6. Moreover, the patterns *fail-safe* and *failed expectations*, as proposed by Wagner et al. (2010), can be used in the present method (cf. Section 4.6.1), and for behavioural modelling and non-linear reasoning.

On the one hand, I have several remarks on more recent research concerned with guidance (cf. Section 3.4.3): Kelly (1998) neglects behavioural modelling as opposed to, for example, Leveson (2012), Abrial (2006) and the present work. As indicated in Section 3.4.3, the framework of Thramboulidis and Scholz (2010) provides guidance similar to the present framework. It stays, however, unclear to which extend behavioural models are used, not least because of a missing formalisation. The concepts of Lund et al. (2011) are related to the present approach as follows: for example, assets can be indirectly referenced by the variables used to describe mishaps, vulnerabilities can be modelled in  $\mathcal{A}_{S|fail}$  (e.g. defects), threats can be modelled in  $\mathcal{A}_{E|fail}$  (e.g. maloperation), incidents form runs fulfilling a mishap, risks correspond to abstract events or hazards, and treatments match safety measures. The procedure discussed by the authors investigates collaboration and approval, and takes into account ISO Std. 31000 (2009), more than the procedure explained in Chapter 5 does. Lund et al., however, barely apply patterns of transition systems and behavioural properties.

On the other hand, this thesis goes further than many works in formal safety or reliability analysis using modular behavioural models (cf. page 41f): For example, McDermid and Pumfrey (1994) and Fenelon et al. (1994) refused to provide a precise semantics of the applied guide words. The approach of Heitmeyer et al. (1998) excludes an environment model, the notions of mishaps and aspects (Definition 2.14), and the use of temporal logic. Heitmeyer et al. do not distinguish modes from other abstract states which can enlarge specifications but avoid the effort of creating an additional abstraction. Moik (1999) and Bitsch et al. (1999) disregard patterns and procedure. As opposed to the approach of Herrmann and Krumm (1999, 2000), the present work has a probabilistic temporal notion of hazard as well as several MTS patterns to follow. Neogi (2002) applies backward reasoning without using MTS models and past formulae. David et al. (2010) neglect the use of behavioural properties and patterns for safety measures. Compared to the present method, Haxthausen et al. (2011, 2014) are missing patterns, a procedure and an environment model to verify safety in a more holistic fashion (cf. Section 3.1).

The comparison in Table A.10 (Appendix A.4) provides evidence that the proposed

## 7. Discussion

method largely coheres with known and practiced safety procedures: For example, ISO 26262 part 1 also distinguishes hazardous, defective and operational states (cf. Figure 4.5). The present approach supports validation (Definition 3.1) where many design decisions are still missing. It addresses ISO 26262 part 3 on hazard analysis and conception of safety measures, and coheres with IEC 61508 in that  $\Gamma$  corresponds to *safety requirements* and  $M_{\text{save}}$  to *safety functions*. Furthermore, the method provides *structured, reusable modelling* aligned with requirements engineering, design and system testing.

**Probabilistic Causative Reasoning** I may comment on some of the related approaches: PCTL formulae according to Johnson (1993) can be evaluated for plain computation trees instead of MARKOV chains. In contrast, the present approach uses probability parameters being part of the action specifications of an MTS which generates MARKOV chains. As described in Section 3.4.3, Lund et al. (2011) present a more elaborate investigation of risk analysis based on the frequency view of probability. In contrast, the present work provides a more constructive view of behavioural specification by using MTS models.

Nissanke and Dammag (2002) present an approach similar to the modelling of the three aspects  $M_{\text{use}}$ ,  $M_{\text{fail}}$  and  $M_{\text{save}}$ , and the abstract transitions shown in Figure 4.5: Their “timed safety clauses” correspond to superimposition of  $M_{\text{save}}$  using indeterminisation and action priorities. Definition 4.1 suggests models where risk classes are derivable from mishaps, hazards and effect probabilities using probabilistic reasoning. MTS models use static action priorities instead of a dynamic determination of these priorities. This can be a restriction of the present method. Nissanke and Dammag, nevertheless, leave out a discussion of modelling guidance, an environment model, maloperation and formal semantics. Their notion of risk is only qualitative and can make model validity difficult to assess.

### 7.3. Some Limitations of the Approach

Next, I indicate *premises and choices* which can constrain method applicability:

1. *Early use*: As late defect removal is costly, the method should be applied in early life cycle stages (e.g. prototype validation, model-in-the-loop analysis). In contrast to Pyle (1991), the present method only considers two viewpoints (cf. page 48). Fewer viewpoints might save effort but increase the risk of oversights during analysis as indicated by Wilson and McDermid (1995).
2. *Abstraction independent of technology*:  $\mathcal{M}$  reduces the control loop to the boundary between  $\mathcal{A}_E$  and  $\mathcal{A}_S$  (Section 4.1); internals of actions and causal factors may be omitted in  $\mathcal{M}$  to cope with a lack of design knowledge. Such knowledge can be obtained later, for example, by FMEA or FTA of an architecture design. As a drawback, coincidental faults may occasionally compensate

themselves such that negative effects would not show up at the system boundary.  $\mathcal{M}$ , however, is suited to guess and capture<sup>2</sup> potential defects (Section 4.2).

3. *Abstraction to qualitative model*: Action effects are based on a uni-granular (i.e. globally synchronous), discrete equidistant time abstraction. Composition, state and action abstraction are constrained to the set of used variables.
4. *Further development and V&V*: By Definition 4.3,  $\mathcal{W}$  is *defective* if it deviates from  $\mathcal{M}$  and  $\mathcal{M}'$ . As long as  $S'$  is unknown, any lack of safety knowledge and observability (e.g. due to wrong abstraction) entails two *threats*:
  - $f^-$  A defect stays hidden (false negative):  $\mathcal{W}$  deviates from  $\mathcal{M}'$  and conforms to an unknowingly defective  $\mathcal{M}$ .
  - $f^+$  An observed deviation does not express a defect (false positive):  $\mathcal{W}$  conforms to  $\mathcal{M}'$  but deviates from an unknowingly defective  $\mathcal{M}$ .

Hidden and inexpressive deviations<sup>2</sup> of  $\mathcal{W}$  from  $S'$  and  $S$  can obstruct conclusions on *hazards* (Gleirscher 2011).  $\mathcal{M}$  may *hide hazards* in general ( $f_{\mathcal{M}}^-$ ) or for a specific system realisation ( $f_{\mathcal{W}}^-$ ), and lead to *wrongly assessed hazards* ( $f^+$ ).

## 7.4. Challenges and Hints

Here, I derive challenges and hints motivated by the limitations in Section 7.3.

**Challenge 1** *Achieve model validity* in  $S$  in spite of four<sup>3</sup> ways of *abstraction* of  $\mathcal{W}$ :

1. Typification of states, modes and events ( $\mathcal{V}, \mathbb{T}$ )
2. Indeterminacy, probabilistic and time abstraction, priorities ( $\mathcal{M}$ )
3. Mode and action abstraction in the mode transition system ( $\mathcal{M}$ )
4. State and action abstraction in the abstract transition system ( $\mathcal{M}_\alpha$ )

**Hints** To avoid threat  $f_{\mathcal{M}}^-$ , education and experience of safety engineers should be improved; additional guide words and MTS patterns should be provided; techniques for requirements elicitation, probability estimation and model checking with refinement should be used. To avoid threat  $f_{\mathcal{W}}^-$ , regard variable-specific instrumentation for measuring the enabling conditions of actions, the actuation of their effects and the embodiment of actions (e.g. control input events stimulated by humans, physical events stimulated by mechanical test beds).

*Model validity* incorporates the goal of  $\mathcal{M}$  to be a *correct abstraction*<sup>4</sup> of  $\mathcal{W}$ , or that  $\mathcal{M}$  and  $\mathcal{W}$  are *bisimilar* (Pappas 2003) or in a *conformance relation* (Rusu

<sup>2</sup>Appendix A.8 further discusses the extent to which certain defects can be observed.

<sup>3</sup>Excluding *interface abstraction* of transition systems according to Broy and Stølen (2001).

<sup>4</sup>See, e.g. Bloomfield et al. 1991, Clarke et al. 2000, Henzinger 2000, Parnas et al. 1990, Pyle 1991, Smith 1995.

## 7. Discussion

et al. 2005). Sargent (1999) discusses the notions of *conceptual model validity* and *data validity* which should be checked for any MTS: such validation would include the *estimation of effect probabilities* which can be carried out using several techniques.<sup>5</sup> This topic is, however, out of scope of the present work. In control theory, Smith and Doyle (1992) and Ljung (1998) connect the validation problem with *identification experiments* relating uncertain models with measured data. According to Sargent’s conclusion, threats to model validity can remain independent of the expressiveness of the chosen modelling language (e.g. mode transitions systems) and temporal logic (e.g. PCTL\*, Section 2.3.2). Model validity is further investigated in Section 7.6, Appendix A.8 and by Trochim and Donnelly (2008).

**Challenge 2** *Achieve compactness, effectiveness and reusability of  $\mathcal{M}$ :*

1. Leverage the expressiveness of the discussed model to keep  $\mathcal{M}$  lean (e.g. permissive behaviour of humans according to Jackson 2001).
2. Use simplifications (e.g. merge or split actions and types).
3. Avoid or handle effect interference in composite actions (e.g. shared output devices such as actuators and displays).
4. Concisely represent MTSs with many dependencies (e.g. de-/activation or disturbance imposed by energy control or “filter” functions).
5. Concisely model actions with strongly heterogeneous time granularity (e.g. concurrency of human and system actions).
6. Leverage reuse (e.g. use of a validated MTS in a different environment as a fragment of another MTS) and variability (e.g. composition or superimposition with optional and alternative choice of MTSs; Apel and Lengauer 2008).

**Hints** As MTSs can hide internals, we may require input from structural models to explain certain behaviour. Indeterminacy for underspecification enhances defect modelling, although deterministic models idealise predictability of a technical system to make fault localisation easier. Indeterminacy can support safety engineers in dealing with a lack of knowledge of the interfaces, the structure, and the initialisation of the system and the environment. For concise and reusable description of functions and defects, *non-deterministic choice* of actions in  $\mathcal{M}$  (see page 20) can take place among several

- enabled *coercive* actions (*explicit*)
- enabled *permissive* actions and  $\kappa$  (*explicit*)
- effects for  $\kappa$  (*implicit*)

---

<sup>5</sup>See, e.g. Börzsök 2011, Gaede 1977, Kumamoto 2007.



- actions enabled because of an incomplete perception of the system boundary (Broy et al. 2009) or isolating<sup>6</sup> an MTS by removing its mode channels (*quasi*).

System realisations are inherently complete and deterministic but possibly hazardous. Hence, each incomplete or non-deterministic  $\mathcal{A}_S$  needs to be made *causal*, to form a *realisable* system specification (Broy and Stølen 2001), and *safe*, to form a *valid* system specification  $\mathcal{A}'_S$ . Although we might be able to construct  $\mathcal{A}'_S$  to be coercive and deterministic,  $\mathcal{A}_E$  may stay permissive and non-deterministic, even violate  $\Gamma'$  or deviate from  $\mathcal{M}'$  or  $\mathcal{M}$ . Section 4.5 addresses this issue by applying A/G style.

*Heterogeneous operating speeds* should be taken care of because the execution of an action always takes one time step: Several intermediate effects during the interval of one time step can be encoded by sequences of values instead of single values (Broy 2010). An effect which requires more than a single time step can be encoded by a partial effect if its variable type is precise, or by an additional waiting mode with an  $\epsilon$  action and a time-triggered control action leading to the mode performing the effect.

**Challenge 3** *Achievement and preservation of behavioural safety* across composition ( $\oplus, \otimes$ ) and evolution ( $\rightsquigarrow$ ) in  $\mathcal{S}$ .

**Hints** The composition and evolution operators support *scalability* and *evolvability*. These abilities require ways to derive a conclusion on behavioural safety from piecewise evaluations of MTSs and behavioural properties. Hence, we have to consider mechanisms for the composition operators which *preserve behavioural safety*, that is, which are *modular* with respect to this property. Such mechanisms can reduce necessary and unwanted functional dependencies and property interrelation.

*Property achievement in  $\mathcal{A}_S|_{\text{save}}$* : For *fault tolerance* as an extension of software correctness, Gärtner (1999) explains program transformations (i.e. fault detection or correction based on a defect model) and corresponding changes of property assertions.

*Property preservation of  $\oplus, \otimes$  in  $\mathcal{M}$  with  $\Gamma$* :  $\mathcal{M}$  can be annotated with A/G pairs. In as many as possible situations, including functional dependencies and effect interference, the operator  $\otimes$  has to support the entailment

$$\{\mathcal{A}_{s_1}\}\mathcal{M}_1\{\text{Gr}_1\} \wedge \{\mathcal{A}_{s_2}\}\mathcal{M}_2\{\text{Gr}_2\} \models \{\mathcal{A}_{s_1} \wedge \mathcal{A}_{s_2}\}\mathcal{M}_1 \otimes \mathcal{M}_2\{\text{Gr}_1 \wedge \text{Gr}_2\}$$

(see, e.g. Broy 1998). Such an entailment is also required for the operator  $\oplus$ , except for a reduced guarantee  $\{\text{Gr}_1 \vee \text{Gr}_2\}$ .

*Property preservation of  $\rightsquigarrow$* : The cases  $\mathbb{T} \not\subseteq \mathbb{T}'$  and  $\mathcal{V} \not\subseteq \mathcal{V}'$  are common and can obstruct comparability of  $\llbracket \mathcal{M} \rrbracket$  and  $\llbracket \mathcal{M}' \rrbracket$ . Regarding the premises 2 and 3 (cf. page 108), correct abstraction, however, is supposed to be maintained by Definition 4.14. Gleirscher et al. (2014) discuss an application of evolution in practice.

<sup>6</sup>An MTS is isolated by excluding variables in  $\mathcal{V}_m$  from its action specifications, resulting indeterminacy can be resolved again by parallel composition (Definition 2.9).

## 7. Discussion

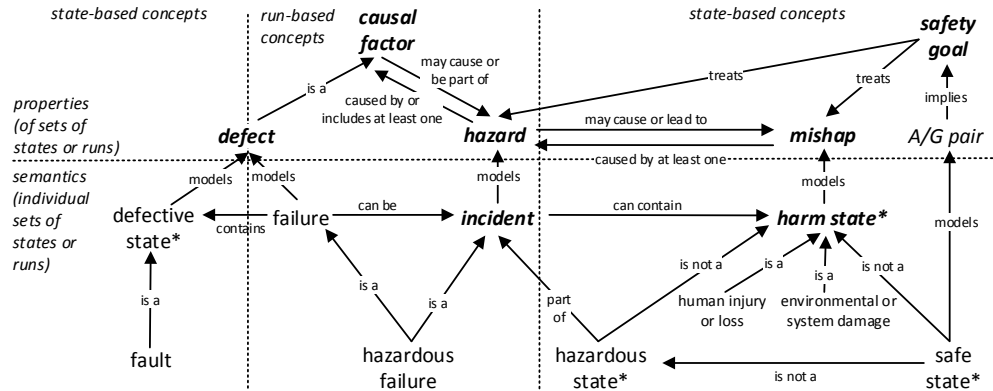


Figure 7.1.: Overview of concepts for behavioural safety, \*... can be an abstract state

## 7.5. Conclusions

*Safety-oriented requirements validation* represents the transition from an “actually specified and modelled”  $S$  to a “safely specified and modelled”  $S'$  ( $S \rightsquigarrow S'$ ). Beyond this transition, a system life cycle has to include an assured realisation ( $S' = \mathcal{W}'$ ). Final V&V of a realisation for safety is, however, omitted here. Anyway, the present method helps collect evidence and build up argumentation for safety of a technical system and its software-intensive control subsystem in early life cycle stages:

*A safe specification  $S'$  implies that each software-intensive control subsystem, implementing the specified safety measures, contributes to hazard freedom and, hence, safety.*

The method guides safety analysis starting from use cases, operational situations and mishaps, and ending in the knowledge of causal factors, hazards and safety goals. The approach uses system modelling to achieve (i) the encoding of domain, requirements and defect knowledge (answer to RQ1), (ii) the assessment of hazards based on this knowledge, and (iii) justified safety measures to be accomplished by subsequent engineering (answer to RQ2).

Figure 7.1 depicts concepts for safety-oriented requirements validation. The framework defined in Section 4.1 helps compare ideal, actual, specified and realised behaviour to conceptualise *causal factors*. The *abstraction* applied in  $S$  enables safety engineers to represent and communicate safety-related domain knowledge in a way largely independent of technology. The world model  $\mathcal{M}$  helps focus on *which* hazards could reasonably and potentially occur prior to *how* assessed hazards can actually occur. This way, the method leverages *reuse* of this knowledge in requirements engineering, functional safety, reliability engineering and system testing. The approach is *a-priori usable* in requirements validation whereas a-posteriori approaches can be biased by strong dependence on technological specificities.

The method combines hazard knowledge with a *defect model* which represents impacts of technology-specific faults on causal factors. Reliability analysis of a system design model, for example, component-level FMEA, can justify or refute this defect model as soon as required.  $S'$  can then help improve safety integrity. The *environment model* enhances defect modelling, hazard identification and the planning of safety measures. Moreover, the identification of *hazardous behaviour* permitted by a specification results in the knowledge of *specification defects*. Here, three cases are of particular interest:

1. *Long-term interactivity*: Indicated by Leveson (2012), it is not only required to trace defects through a known design (e.g. by FMEA, FTA) but also to observe behaviour to understand hazardous operational situations and to identify subtle defects in  $S$ . Behavioural reasoning elucidates whether hazards are caused by past states and *long-term deviations* from safe behaviour of any or both, the system and its environment.
2. *Human action*: The defect model in  $S$  accommodates the *automation paradox* (cf. Section 1.1) by enabling the identification of hazards resulting from intentional or unconscious *misbehaviour of the environment* (e.g. security attack, driver inattention) including *intervention* if failure or maloperation occurs.
3. *Complex functionality*:  $S$  enables the assessment of hazards which arise from concurrent functions and their assumed or inexactly known *hazardous dependencies*. The physical side effects possibly underlying such dependencies can, for example, be modelled via *mode channels* and *effect interferences*.

Knowledge of unexpected and defective behaviour in these cases motivates the *specification of passive and preventive safety measures*. Hence, the approach helps *treat hazardous specification defects* prior to *treating hazardous operational defects*.

$M$  should capture as many relevant situations in the application domain as possible. Hence, *model completeness and validity* are decisive (cf. challenges 1 and 2 in Section 7.4). The MTS patterns for predictive defect modelling, and the guide words for mishap modelling and causal factor search (cf. Section 4.3) constitute prerequisites for a complete and valid  $M$ . In the procedure shown in Chapter 5, these means help identify variables to assert constraints and properties of, for example, side effects and faults (Step 2), harm states (Step 3), and hazardous states (Step 4). The abstract transition system  $M_\alpha$  helps proceed from  $\Phi$  (Step 3) and  $\mathcal{H}$  (Step 4) to  $\Gamma$  (Step 5). Furthermore, the well-formedness conditions on non-deterministic<sup>7</sup> (i.e. distribution unspecified, e.g. unknown) and probabilistic choice (i.e. distribution specified, e.g. equipartition), and prioritisation constrain MTS *modelling decisions*.

A/G-based safety goals help specify practicable safety measures by imposing responsibilities: A/G pairs can reduce difficulties in resolving specification defects and in treating hazards through safety measures for extreme, unexpected or defective behaviour. The value of a guarantee and the strength of the associated assumption, however, justify the interest to fulfil this assumption. In summary, safety goals constrain

<sup>7</sup>Section 7.4 discusses several kinds of *indeterminacy*.

## 7. Discussion

the set of safe models and contribute to finding a safe realisation ( $\llbracket \mathcal{W} \rrbracket \subseteq \llbracket \mathcal{M} \rrbracket \cap \llbracket \widehat{\Pi} \rrbracket$ , cf. Figure 4.7).

Underpinned by the evaluation of the case study, the proposed solution supports safety-oriented requirements validation (RG, cf. Section 1.1). By comparison with related work (Section 3.4.2), Section 7.2 provides further evidence that this solution is an improvement on the state of the art, particularly, on Leveson’s viewpoint (cf. Section 3.1) which has been mostly pursued by this thesis. Hence, I might suggest that this thesis contributes to the field of safety engineering.

## 7.6. Further Work

This section enumerates several topics that can be pursued in the future.

**Development of Verification Techniques** The construction and evolution of MTSs requires the proof of standard algebraic properties of the operators  $\oplus$ ,  $\otimes$  and  $\alpha$  as shown, for instance, by Hoare (1985). State and action abstraction motivates bisimulation proofs, for example, according to Milner (1973) and Pappas (2003). Usually, treatments raise the obligation of refinement proofs as discussed by Broy and Stølen (2001). The Steps 4.3, 4.5, 5.2, 5.4 and 6.3 can be supported by compact inference rules for PCTL\* formulae similar to the rules proposed by Pnueli and Kesten (2002).

**Support of Evolutionary Development by the Procedure (Chapter 5)** Which changes to  $\mathbb{M}_{\text{fail}}$  are required after having transferred  $\mathbb{M}_{\text{save}}$  to  $\mathbb{M}_{\text{use}}$  to conduct a further iteration? Which parts of  $\mathcal{S}$  are affected when changing the system boundary (e.g. because of a changed control concept)? As mentioned, *preservation of behavioural safety* in  $\mathcal{S}$  is required during composition and evolution.  $\mathcal{S}$  is subject to evolution, for example, because of consumer or technology market changes (Gleirscher et al. 2014).

**Automation of the Procedure (Chapter 5)** To which extent can  $\mathbb{M}_{\text{fail}}$  be *generated* from  $\mathbb{M}_{\text{use}}$  (Step 2)? For example, guessing defects (Step 2.4) by deriving defective modes from specified modes, by modifying actions, by adding  $\tau$  or  $\epsilon$  actions. How can mode, action and event guide words be combined with property patterns to *refine hazards* (Step 4)? Adapted from Dwyer et al. (1999), for example, “<unattended> <start> of ‘move’ ” yields “move erroneously precedes userAction” and  $E(\neg(\text{move} \vee \text{userAction}) \text{U} (\text{move} \wedge \neg \text{userAction}))$ . See Dobi et al. (2013) and Letier (2001).

**Practical Evaluation** How well does the method fit practical needs? Does the effort of deriving  $\mathcal{S}'$  exceed the benefits? How can the application of MTS patterns for defects and safety measures be improved? How can the application of guide words for mishaps, causal factors and hazards be improved? How can state and action abstraction be more effectively combined with pattern application?

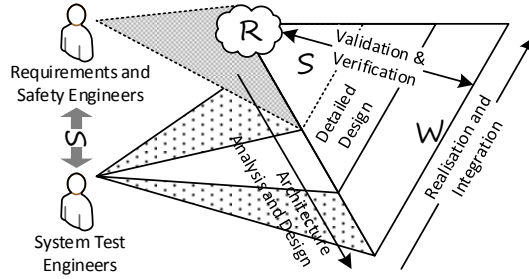


Figure 7.2.: Specification-based testing in V&V according to Figure 2.1

**Integration with Other Approaches** *Dynamic FTA* (Dugan et al. 1992) can be efficiently abstract and conservative by reducing a component’s modes to a *defective* and an *operational* mode. How well can  $\mathcal{A}_{S|_{\text{fail}}}$ , the analysis stage (Section 5.2) and  $\mathcal{A}_{S|_{\text{save}}}$  be mapped to FTA concepts (e.g. dependencies, and/or, k-of-m, priorities, failures, permanent or transient faults, minimal cut sequences, coverage components, cold/warm/hot spares)?

*Specification-based testing* of safety-critical control systems<sup>8</sup> is supported by  $S'$ . This technique can help test engineers assess hazardous defects and strengthen the argument for  $\mathcal{W} = \mathcal{W}'$  (see Figure 7.2): Which test cases can and should be derived from  $S'$ ? To which extent is  $S'$  suitable for stimulation and response monitoring, and for the definition of test stop criteria? Test cases which cover  $\mathbb{M}_{\text{fail}}$  and  $\mathbb{M}_{\text{save}}$  are of particular interest. Dasgupta (2006) discusses the build-up of  $\Gamma$  and A/G pairs to cover  $\mathcal{M}$ . Failed tests can reveal specification defects to be used in *model validation* (cf. challenge 1 in Section 7.4), or operational defects to be included in  $\mathbb{M}_{\text{fail}}$  (cf. Step 2.2b in Section 5.1.2). Appendix A.8 and Gleirscher (2011) describe *hazard-based testing* and a test bed instrumentation.

According to *game theory*,  $\mathcal{A}_E$  and  $\mathcal{A}_S$  can be seen as players: How can knowledge about mishaps and defects be converted into friendly ( $\mathbb{M}_{\text{save}}$ ) and adversarial strategies ( $\mathbb{M}_{\text{fail}}$ ) within the freedom of choice in  $\mathcal{M}$ ? The harder the game for the adversarial parts, the safer the game for the environment to achieve T1 and T2 (cf. page 36). How can A/G pairs be turned into *knowledge* about past and *beliefs* about future, and, hence, into a stochastic game with incomplete knowledge? Game theory has been applied to reachability analysis, reasoning on compatibility and A/G-based refinement of transition systems (de Alfaro and Stølinga 2004), synthesis of fault-tolerant controllers (Cheng et al. 2012, D’Ippolito et al. 2011) and model checking (Peled et al. 1999).

*Legal theory* and safety assurance may benefit from each other, for example, using lawyers’ methods to constructing safe specifications and jurisprudence’s methods to verify system realisations (Logrippo 2014).

<sup>8</sup>See, e.g. Beizer 1995, Das et al. 2012, Gleirscher 2013a, Liggesmeyer 2009, Peleska 1996, Pyle 1991, Rusu et al. 2005.

## Bibliography

- Abdulkhaleq, A. and S. Wagner (2013). Integrating state machine analysis with system-theoretic process analysis. In *Multikonf. Soft. Eng.: ZeMoSS – Zertifizierung und modellgetriebene Entwicklung sicherer Software*, Aachen, Germany. 45
- Abrial, J.-R. (1998). On B. In D. Bert (Ed.), *B*, Volume 1393 of *LNCS*, pp. 1–8. Springer. 16
- Abrial, J.-R. (2006). Train systems. In M. J. Butler, C. B. Jones, A. Romanovsky, and E. Troubitsyna (Eds.), *RODIN Book*, Volume 4157 of *LNCS*, pp. 1–36. Springer. 39, 42, 107
- ADAC (2012). Zahlen, Fakten, Wissen. Aktuelles aus dem Verkehr. Online: [www.adac.de/\\_mmm/pdf/statistik\\_zahlen\\_fakten\\_wissen\\_0512\\_46600.pdf](http://www.adac.de/_mmm/pdf/statistik_zahlen_fakten_wissen_0512_46600.pdf), accessed: 2013-03-23. 2
- Agha, G. and C. Hewitt (1985). Concurrent programming using actors: Exploiting large-scale parallelism. In *Found. of Soft. Tech. and Theor. Comp. Sci.*, Volume 206 of *LNCS*, pp. 19–41. Springer. 14, 15
- ANSI and IEEE (1983). ANSI/IEEE Std. 729: Standard Glossary of Software Engineering Terminology. 16
- Apel, S. and C. Lengauer (2008). Superimposition: A language-independent approach to software composition. In *Software Composition*, Volume 4954 of *LNCS*, pp. 20–35. Springer. 29, 110
- Atlee, J. M. and J. A. McDermid (1995, Mar). Integrating requirements analysis and safety analysis. In *2nd IEEE Int. Symp. Req. Eng.*, York, England, pp. 158–9. IEEE CS. 3, 32
- Baader, F., D. Calvanese, D. L. McGuinness, D. Nardi, and P. F. Patel-Schneider (Eds.) (2003). *The Description Logic Handbook: Theory, Implementation, and Applications*. Cambridge U P. 15

- Baier, C. and J.-P. Katoen (2008, May). *Principles of Model Checking*. MIT Press. [13](#), [15](#), [16](#), [17](#), [27](#), [28](#), [29](#), [66](#)
- Bainbridge, L. (1983, Nov). Ironies of automation. *Automatica* *19*(6), 775–9. [4](#)
- Barroca, L. M. and J. A. McDermid (1992). Formal methods: Use and relevance for the development of safety-critical systems. *Comp. J.* *35*(6), 579–99. [32](#)
- Basili, V., G. Caldiera, and D. H. Rombach (1994). The goal question metric approach. In J. Marciniak (Ed.), *Encyclopedia of Software Engineering*. Wiley. [9](#)
- BAUA (2012). Bundesanstalt für Arbeitsschutz und Arbeitsmedizin – Tödliche Arbeitsunfälle 2001–2010. Online: [www.baua.de/Toedliche-Arbeitsunfaelle](http://www.baua.de/Toedliche-Arbeitsunfaelle), accessed: 2013-02-28. [2](#)
- Beizer, B. (1995). *Black-Box Testing*. Wiley. [3](#), [14](#), [15](#), [115](#)
- Biehl, M., C. DeJiu, and M. Törngren (2010, Apr). Integrating safety analysis into the model-based development tool chain of automotive embedded systems. In *SIGPLAN/SIGBED Conf. Languages, Compilers, and Tools for Embedded Systems*, Stockholm, Sweden, pp. 125–32. ACM. [38](#), [39](#), [40](#), [146](#), [147](#)
- Bitsch, F. (2001, Sep). Safety patterns – the key to formal specification of safety requirements. In U. Voges (Ed.), *Comp. Safety, Reliability and Security*, Volume 2187 of *LNCS*, pp. 176–89. Berlin, Germany: Springer. [41](#)
- Bitsch, F., E. Canver, and A. Moik (1999, Dec). Strukturierte Erstellung von Sicherheitsspezifikationen in UML mit Hilfe der FMEA-Methode. In E. Schnieder (Ed.), *Workshop Formale Techniken für die Eisenbahnsicherung*, Number 436 in Fortschrittsberichte 12: Verkehrs-/Fahrzeugtechnik, Braunschweig, Germany, pp. 225–45. VDI. [41](#), [107](#), [146](#), [147](#)
- Bloomfield, R., P. Froome, and B. Monahan (1991). Formal methods in the production and assessment of safety critical software. *Reliability Eng. & Sys. Safety* *32*(1-2), 51–66. [32](#), [109](#)
- Boehm, B. W. (1991). Software risk management: Principles and practices. *IEEE Software* *8*(1), 32–41. [33](#), [36](#), [45](#)
- Börscök, J. (2011, May). *Funktionale Sicherheit: Grundzüge sicherheitstechnischer Systeme* (3rd ed.). VDE Verlag. [31](#), [34](#), [35](#), [36](#), [37](#), [65](#), [66](#), [106](#), [110](#), [158](#)
- Börger, E. and R. Stärk (2003). *Abstract State Machines: A Method for High-Level System Design and Analysis*. Springer. [29](#)
- Botaschanjan, J. and B. Hummel (2009). Specifying the worst case – orthogonal modelling of hardware errors. In *18th Int. Symp. Software Testing and Analysis*, New York, NY, USA, pp. 273–84. ACM. [17](#), [39](#)
- Bowles, J. and C. Wan (2001). Software failure modes and effects analysis for a small embedded control system. In *Ann. Reliability and Maintainability Symp.*, Philadelphia, PA, USA, pp. 1–6. IEEE. [38](#), [39](#), [41](#), [42](#), [146](#), [147](#)
- Braun, P., J. Phillips, B. Schätz, and S. Wagner (2009, Sep). Model-based safety cases for software-intensive systems. *Electr. Notes Theor. Comput. Sci.* *238*(4), 71–7. Position paper, 1st Int. Worksh. Cert. of Safety-crit. Soft. controlled Sys. [4](#)

## Bibliography

- Breitling, M. (2000). Modeling faults of distributed, reactive systems. In M. Joseph (Ed.), *Formal Techniques in Real-Time and Fault-Tolerant Systems*, Volume 1926 of *LNCS*, pp. 58–69. Springer. 17, 39
- Broy, M. (1998). A functional rephrasing of the assumption/commitment specification style. *Formal Methods in System Design* 13(1), 87–119. 15, 66, 111
- Broy, M. (2005). *Engineering Theories of Software Intensive Systems*, Chapter "Service-oriented Systems Engineering: Specification and Design of Services and Layered Architectures – The JANUS Approach", pp. 47–81. Springer. 14, 15, 29, 73
- Broy, M. (2010). A logical basis for component-oriented software and systems engineering. *The Computer Journal* 53(10), 1758–82. 14, 15, 29, 73, 111
- Broy, M. (2012, May). Functional safety based on a system reference model. In *Austral. Sys. Safety Conf.*, Brisbane, QLD, Australia. Keynote paper. 4, 5, 40
- Broy, M., C. Leuxner, W. Sitou, B. Spanfeller, and S. Winter (2009). Formalizing the notion of adaptive system behavior. In *Symp. App. Comp.*, pp. 1029–33. ACM. 65, 111
- Broy, M., B. Penzenstadler, M. Gleirscher, and J. Eckhardt (2012, Apr). Requirements Engineering. Online: [www4.in.tum.de/lehre/vorlesungen/re/ss12](http://www4.in.tum.de/lehre/vorlesungen/re/ss12), accessed: 2013-02-28. Lecture material, summer term, course no. IN2198, Institut für Informatik, Technische Universität München. 9, 76
- Broy, M. and K. Stølen (2001). *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*. Springer. 14, 15, 16, 26, 29, 30, 41, 66, 67, 109, 111, 114, 141
- Buys, J. and J. Clark (1995). Events and Causal Factors (ECF) Analysis. Technical Report SCIE-DOE-01-TRAC-14-95, Technical Research and Analysis Center, Idaho Falls, USA. SCIENTECH. 35, 146, 147
- Cacciabue, P. C. (2004, Sep). *Guide to Applying Human Factors Methods: Human Error and Accident Management in Safety-Critical Systems*. Springer. 1, 35, 146, 147
- Catino, C. and L. Ungar (1995). A model-based approach to automated hazard identification of chemical plants. *AIChE Journal* 41(3), 97–109. 38, 39, 41, 146, 147
- Chen, D.-J., R. Johansson, H. Lönn, Y. Papadopoulos, A. Sandberg, F. Törner, and M. Törnren (2008). Modelling support for design of safety-critical automotive embedded systems. In M. D. Harrison and M.-A. Sujan (Eds.), *SAFEComp*, Volume 5219 of *LNCS*, pp. 72–85. Springer. 38, 39, 40, 146, 147
- Chen, P. P.-S. (1976). The Entity-Relationship Model – Toward a unified View of Data. *ACM Trans. Database Syst.* 1(1), 9–36. 15
- Cheng, C.-H., M. Geisinger, H. Ruess, C. Buckl, and A. Knoll (2012). Game solving for industrial automation and control. In *IEEE Int. Conf. Robotics and Automation*, pp. 4367–72. 115
- Chillarege, R., I. Bhandari, J. Chaar, M. Halliday, D. Moebus, B. Ray, and M. Wong (1992). Orthogonal defect classification – a concept for in-process measurements. *IEEE Trans. Soft. Eng.* 18(11), 943–56. 39



- Clarke, E. M., O. Grumberg, S. Jha, Y. Lu, and H. Veith (2000). Counterexample guided abstraction refinement. In E. A. Emerson and A. P. Sistla (Eds.), *CAV*, Volume 1855 of *LNCS*, pp. 154–69. Springer. 29, 109
- Cockburn, A. (2000, Oct). *Writing Effective Use Cases*. Crystal Series for Software Development. Amsterdam: Addison-Wesley Longman. 13, 104
- Courtois, P.-J. and D. Parnas (1993, May). Documentation for safety critical software. In *15th ICSE*, pp. 315–23. 14, 29, 32
- Damm, W. and T. Peikenkamp (2004, Jul). Modellbasierte Sicherheitsanalyse in STATE-MATE / Model-based Safety Analysis. Online: [www.informatik.hu-berlin.de/studium/ringvorlesung/ss04](http://www.informatik.hu-berlin.de/studium/ringvorlesung/ss04), accessed: 2013-02-15. Lecture material, lecture series on “Model-based Development” at Humboldt Universität Berlin. 39, 40, 65, 146, 158
- Das, S., A. Banerjee, and P. Dasgupta (2012). Early analysis of critical faults: An approach to test generation from formal specifications. *IEEE Trans. on CAD of Integrated Circuits and Systems* 31(3), 447–51. 17, 115
- Dasgupta, P. (2006, Jul). *A Roadmap for Formal Property Verification* (1st ed.). Springer. 43, 106, 115
- Dasgupta, P. (2012, May). Formal specification – from theory to practice. Presentation slides, talk series at Technische Universität München. 43
- David, P., V. Idasiak, and F. Kratz (2010, Apr). Reliability study of complex physical systems using SysML. *Reliability Eng. and Sys. Safety* 95(4), 431–50. 41, 107, 146, 147
- Davis, A. M. (1988). A comparison of techniques for the specification of external system behavior. *Commun. ACM* 31(9), 1098–115. 15
- de Alfaro, L. and M. Stølinga (2004). Interfaces: A game-theoretic framework for reasoning about component-based systems. *Electr. Notes Theor. Comput. Sci.* 97, 3–23. 115
- Dehlinger, J. and J. B. Dugan (2008, Aug). Dynamic event/fault tree analysis of multi-agent systems using Galileo. In H. Zhu (Ed.), *8th Int. Conf. Quality Software*, Oxford, UK, pp. 429–34. *IEEE CS*. 146, 147
- DeMarco, T. (1979). *Structured Analysis and System Specification*. Yourdon Press. Upper Saddle River, Englewood Cliffs, NJ, USA: Prentice-Hall. 15
- Diaconescu, R. S. (2011). Automatic Pre-Processing of Accident Data for Evaluation of Preventive Safety Systems. Bachelor thesis, Technische Universität München. 4
- D’Ippolito, N., V. A. Braberman, N. Piterman, and S. Uchitel (2011, May). Synthesis of live behaviour models for fallible domains. See Taylor et al. (2011), pp. 211–20. 39, 43, 66, 106, 115, 146, 147
- Dobi, S., M. Gleirscher, M. Spichkova, and P. Struss (2013, Jun). Model-based Hazard Analysis and Risk Assessment. Technical Report TUM-I1333, Technische Universität München. 5, 9, 39, 41, 65, 73, 90, 96, 114, 146, 147
- Dugan, J., S. Bavuso, and M. Boyd (1992). Dynamic fault tree models for fault-tolerant computer systems. *IEEE Trans. Reliability* 41(3), 363–77. 3, 16, 34, 40, 42, 115, 146, 147

## Bibliography

- Dulac, N. (2007). *A Framework for Dynamic Safety and Risk Management Modeling in Complex Engineering Systems*. Ph. D. thesis, Massachusetts Institute of Technology, Cambridge, MA, USA. [43](#)
- Dwyer, M. B., G. S. Avrunin, and J. C. Corbett (1999). Patterns in property specifications for finite-state verification. In *ICSE*, pp. 411–20. [16](#), [41](#), [73](#), [114](#)
- Ehrlenspiel, K. and H. Meerkamm (2013, Mar). *Integrierte Produktentwicklung: Denkabläufe, Methodeneinsatz, Zusammenarbeit* (5th ed.). Hanser. [1](#), [12](#)
- Ericson, C. A. (2005). *Hazard Analysis Techniques for System Safety*. Hoboken, NJ, USA: Wiley. [2](#), [3](#), [31](#), [33](#), [34](#), [35](#), [66](#)
- Esser, M. and P. Struss (2007). Fault model-based test generation for embedded software. In *20th Int. Joint Conf. Artif. Intell.*, pp. 342–7. [17](#), [39](#), [41](#), [146](#), [147](#)
- Evers, C. (2012, Sep). Unterschätzte Risikofaktoren – Ermüdung und Ablenkung als Ursache für schwere Lkw-Unfälle. Online: DVR Bundesanstalt für Straßenwesen (BASt), [www.dvr.de/download/ps081124-25\\_ec\\_pr.pdf](http://www.dvr.de/download/ps081124-25_ec_pr.pdf), accessed: 2013-02-28. [2](#)
- Fantechi, A., S. Gnesi, F. Mazzanti, R. Pugliese, and E. Tronci (1999). A symbolic model checker for ACTL. In D. Hutter, W. Stephan, P. Traverso, and M. Ullmann (Eds.), *Applied Formal Methods (FM-Trends '98)*, Volume 1641 of *LNCS*, pp. 228–42. Springer. [29](#)
- Feather, M. (2004, Nov). Towards a unified approach to the representation of, and reasoning with, probabilistic risk information about software and its system interface. In *15th IEEE Int. Symp. on Softw. Reliability Eng.*, Saint-Malo, France, pp. 391–402. [39](#), [42](#), [146](#), [147](#)
- Feather, M. and L. Markosian (2011, Aug). Building a safety case for a safety-critical NASA space vehicle software system. In *4th IEEE Int. Conf. Space Mission Challenges for Inform. Tech.*, Palo Alto, CA, USA. [44](#)
- Feather, M. S. (1987). Language support for the specification and development of composite systems. *ACM Trans. Program. Lang. Syst.* *9*(2), 198–234. [16](#), [65](#)
- Fenelon, P., J. A. McDermid, M. Nicolson, and D. J. Pumfrey (1994, Mar). Towards integrated safety analysis and design. *SIGAPP Appl. Comput. Rev.* *2*(1), 21–32. [32](#), [36](#), [42](#), [107](#)
- Floyd, R. (1967). Assigning meanings to programs. In J. Schwartz (Ed.), *Math. Aspects of Comp. Sci.*, pp. 19–32. Amer. Math. Soc. [15](#)
- Focus (2012, Feb). 49 Tote bei schwerem Zugunglück in Buenos Aires. Online: [www.focus.de/panorama/welt/zug-rammt-bahnsteig-schweres-zugunglueck-in-buenos-aires\\_aid\\_716720.html](http://www.focus.de/panorama/welt/zug-rammt-bahnsteig-schweres-zugunglueck-in-buenos-aires_aid_716720.html), accessed: 2013-02-15. [2](#)
- Forejt, V., M. Kwiatkowska, G. Norman, and D. Parker (2011, Jun). Automated verification techniques for probabilistic systems. In M. Bernardo and V. Issarny (Eds.), *Formal Methods for Eternal Networked Soft. Sys.*, Volume 6659 of *LNCS*, pp. 53–113. [66](#)
- Forrester, J. W. (1961). *Industrial Dynamics* (1st ed.). Cambridge U P. [13](#), [14](#), [15](#)
- Friedenthal, S., A. Moore, and R. Steiner (2008, Jul). *A Practical Guide to SysML: The Systems Modeling Language*. Morgan Kaufmann. [9](#), [15](#), [41](#), [104](#)

- Gaede, K. (1977). *Zuverlässigkeit – Mathematische Modelle*. München: Hanser. 36, 110
- Gärtner, F. C. (1999). Transformational approaches to the specification and verification of fault-tolerant systems: Formal background and classification. *J. UCS* 5(10), 668–92. 16, 17, 39, 66, 67, 111, 146, 147
- Gaudel, M.-C. and J. Woodcock (Eds.) (1996, Mar). *3rd IFIP WG 14.3 Int. Symp. Formal Methods Europe: Industrial Benefit and Advances in Formal Methods*, Volume 1051 of LNCS, Oxford, UK. Springer. 128
- Gehlen, P. (2010, Jul). *Funktionale Sicherheit von Maschinen und Anlagen: Umsetzung der Europäischen Maschinenrichtlinie in der Praxis* (2nd ed.). Publicis. 1, 33
- Gleirscher, M. (2011, May). Hazard-based selection of test cases. See Taylor et al. (2011). 5, 9, 39, 65, 109, 115, 159, 162
- Gleirscher, M. (2012, May). Ein Kaffeevollautomat – Fallstudie für modellbasierte Spezifikation zur Vorlesung “Requirements Engineering” im Sommersemester 2011. Technical Report I-125, Technische Universität München. 9
- Gleirscher, M. (2013a, Mar). Another taxonomy of testing approaches. Submitted to ISSTA’13 but rejected in the second review stage. 115
- Gleirscher, M. (2013b, Jan). Hazard analysis for technical systems. In *5th Software Quality Days*, Number 133 in LNBIP, Vienna, pp. 104–24. Springer. vii, 5, 9, 65, 91
- Gleirscher, M. and S. Fuhrmann (2012, Sep). SAFER – Rahmenwerk zur Anforderungsentwicklung für Fahrwerkregel- und Fahrerassistenzsysteme. Internal project report, Technische Universität München & BMW AG. Nondisclosure agreement applies. 9, 91, 92
- Gleirscher, M., D. Golubitskiy, M. Irlbeck, and S. Wagner (2014, Sep). Introduction of static quality analysis in small and medium-sized software enterprises: Experiences from technology transfer. *Software Quality Journal* 22(3), 499–542. 12
- Gleirscher, M., D. Rațiu, and B. Schätz (2007, Mar). Incremental integration of heterogeneous systems views. In *1st Int. Conf. Sys. Eng. and Modeling*, pp. 50–9. IEEE. 12
- Gleirscher, M., A. Vogelsang, and S. Fuhrmann (2014, Aug). A model-based approach to innovation management of automotive control systems. In *8th Int. Workshop Soft. Prod. Mgt.*, Karlskrona, Sweden. IEEE digital library. 9, 91, 111, 114
- Goddard, P. (2000). Software FMEA Techniques. In *RAMS: Ann. Reliability and Maintainability Symp.*, pp. 118–23. IEEE. 3, 35
- Godskesen, J. C. (1999, Sep). Fault models for embedded systems. See Pierre and Kropf (1999), pp. 354–9. Extended abstract. 17
- Hall, J. G., D. Mannering, and L. Rapanotti (2007). Arguing safety with problem oriented software engineering. In *10th IEEE Int. Symp. on High Assurance Sys. Eng.*, Dallas, Texas. 44
- Hall, J. G. and A. Silva (2008). A conceptual model for the analysis of mishaps in human-operated safety-critical systems. *Journal of Safety Science* 46, 22–37. 1, 39, 43, 45, 106, 146, 147

## Bibliography

- Harashima, F., M. Tomizuka, and T. Fukuda (1996). Mechatronics – what is it, why and how? *IEEE/ASME Trans. Mechatronics* 1(1), 1–4. [11](#)
- Harel, D. and R. Marelly (2003, Aug). *Come, Let's Play: Scenario-Based Programming Using LSCs and the Play-Engine* (1st ed.). Springer. [15](#)
- Harel, D. and M. Politi (1998, Oct). *Modeling Reactive Systems With Statecharts: The Stateate Approach*. Software Development. US: McGraw-Hill. [15](#), [29](#), [39](#), [42](#)
- Hatley, D. and I. Pirbhai (1987). *Strategies for Real-Time Systems Specification*. Dorset House. [16](#)
- Hauge, A. A. and K. Stølen (2014, May). An analytic evaluation of the SaCS pattern language – including explanations of major design choices. In *6th Int. Conf. Pervasive Patterns and Applications*, Venice, Italy. Preprint. [44](#)
- Haxthausen, A. E., J. Peleska, and S. Kinder (2011). A formal approach for the construction and verification of railway control systems. *Formal Aspects of Comp.* 23(2), 191–219. [3](#), [43](#), [107](#), [146](#), [147](#)
- Haxthausen, A. E., J. Peleska, and R. Pinger (2014). Applied bounded model checking for interlocking system designs. In S. Counsell and M. Núñez (Eds.), *Software Engineering and Formal Methods*, LNCS, pp. 205–20. Springer. [43](#), [107](#), [146](#), [147](#)
- Heißing, B. and M. Ersoy (2007, May). *Fahrwerkhandbuch: Grundlagen, Fahrodynamik, Komponenten, Systeme, Mechatronik, Perspektiven*. ATZ/MTZ-Fachbuch. Vieweg+Teubner. [4](#), [103](#)
- Heitmeyer, C. L., J. Kirby, B. Labaw, R. Bharadwaj, et al. (1998). SCR\*: A toolset for specifying and analyzing software requirements. In A. Hu and e. M.Y. Vardi (Eds.), *10th Int. Conf. on CAV*, Volume 1427 of LNCS, pp. 526–31. Springer. [16](#)
- Heitmeyer, C. L., J. Kirby, B. G. Labaw, M. Archer, and R. Bharadwaj (1998). Using abstraction and model checking to detect safety violations in requirements specifications. *IEEE Trans. Soft. Eng.* 24(11), 927–48. [3](#), [39](#), [42](#), [107](#), [146](#), [147](#)
- Henzinger, T. (2000). The theory of hybrid automata. In *Verification of Digital and Hybrid Systems*, Volume 170 of NATO ASI Series F: Computer and Systems Sciences, pp. 265–92. Springer. [16](#), [26](#), [65](#), [109](#)
- Herrmann, P. and H. Krumm (1999, Oct). Formal hazard analysis of hybrid systems in cTLA. In *18th IEEE Symp. on Reliable Distributed Systems*, Lausanne, pp. 68–77. IEEE CS. [39](#), [42](#), [107](#)
- Herrmann, P. and H. Krumm (2000, Sep). A framework for the hazard analysis of chemical plants. In *11th IEEE Int. Symp. on Computer-Aided Control System Design*, Anchorage, Alaska, USA, pp. 35–41. IEEE CSS: Omnipress. [42](#), [107](#), [146](#), [147](#)
- Hoare, C. A. R. (1985, Apr). *Communicating Sequential Processes* (1st ed.). Int. Series in Comp. Sci. Prentice-Hall. [15](#), [29](#), [30](#), [114](#)
- Hoffmann, D. W. (2013, Dec). *Software-Qualität* (2nd ed.). Berlin: Springer. [3](#), [12](#), [14](#)

- Hollnagel, E. (2004). *Barriers and Accident Prevention*. Ashgate. 43, 146, 147
- Hopcroft, J. E., R. Motwani, and J. D. Ullman (2006, Jun). *Introduction to Automata Theory, Languages and Computation* (3rd ed.). Prentice-Hall. 15
- Hopkins, A. (2000). *Lessons from Longford: The Esso Gas Plant Explosion*. Sydney, Australia: Arcadia: CCH. 35, 105
- Hummer, W., C. Inzinger, P. Leitner, B. Satzger, and S. Dustdar (2012). Deriving a unified fault taxonomy for event-based systems. In *6th ACM Int. Conf. Distributed Event-Based Systems, DEBS'12*, New York, NY, USA, pp. 167–78. ACM. 39, 106
- IAEA (2008). *The International Nuclear and Radiological Event Scale (INES) – User's Manual* (revised ed.). International Atomic Energy Agency. 2
- IEC (2011). Std. 61508: Functional Safety of Electrical/Electronic/Programmable Electronic Safety-related Systems. 1, 33, 37, 65, 158
- IEEE (1990). Std. 610.12: Standard Glossary of Software Engineering Terminology. 16
- IEEE (1998). Std. 830: Recommended Practice for Software Requirements Specifications. 13
- Illes, T. and B. Paech (2007). An analysis of use case based testing approaches based on a defect taxonomy. In *Software Engineering Techniques: Design for Quality*, Volume 227 of *IFIP Conf. Proc.*, pp. 211–22. Springer. 39
- informyou (Ed.) (2012, Sep). *4th Ann. EUROFORUM Symp. on ISO 26262, Leinfelden-Echterdingen, Stuttgart, Germany*. 124, 131
- ISO (2001). Std. 9126: Software Engineering – Product Quality. 13
- ISO (2006). Std. 13849: Safety of machinery – Safety-related parts of control systems. 33
- ISO (2009). Std. 31000: Risk Management – Principles and Guidelines. 33, 107
- ISO (2011). Std. 26262: Road Vehicles – Functional Safety. 1, 34, 38, 65, 100, 158
- Jackson, M. A. (1983). *System Development*. Prentice-Hall. 1, 12, 15
- Jackson, M. A. (2001). *Problem Frames: Analysing & Structuring Software Development Problems*. Addison-Wesley. 14, 16, 45, 110
- Johnsen, S. O., C. Bjørkli, T. Steiro, H. Fartum, H. Haukenes, J. Ramberg, and J. Skriver (2011, Mar). CRIOP: A scenario method for Crisis Intervention and Operability analysis. Technical Report A4312, SINTEF, Trondheim, Norway. 35, 38, 146, 147
- Johnson, C. (1993). A probabilistic logic for the development of safety-critical, interactive systems. *Int. Journal of Man-Machine Studies* 39(2), 333–51. 28, 42, 44, 66, 108, 146, 147
- Jureta, I., J. Mylopoulos, and S. Faulkner (2008). Revisiting the CORE ontology and problem in requirements engineering. In *16th IEEE Int. Conf. Req. Eng.*, Washington, DC, USA, pp. 71–80. IEEE CS. 16

## Bibliography

- Jürgensohn, T. (2007). *Modelling Driver Behaviour in Automotive Environments – Critical Issues in Driver Interactions with Intelligent Transport Systems*, Chapter 16. “Control Theory Models of the Driver”, pp. 277–92. Springer. [65](#)
- Kahn, G. (1974). The Semantics of a simple Language for Parallel Programming. In *IFIP Congress Proc.*, pp. 471–5. North-Holland. [15](#)
- Kaiser, B., P. Liggesmeyer, and O. Mäckel (2003). A new component concept for fault trees. In *8th Austral. Workshop on Safety-Crit. Sys. and Soft.*, Volume 33, pp. 37–46. [42](#)
- Kath, O. and C. Temple (2012, Sep). From Item Definition to Safety Concept – Process Aspects, Methods and Functional Aspects. See [informyou \(2012\)](#). [38](#), [45](#), [141](#), [148](#)
- Kaynar, D., N. Lynch, R. Segala, and F. Vaandrager (2010, Dec). *The Theory of Timed I/O Automata* (2nd ed.). Synthesis Lectures on Distributed Computing Theory. Morgan & Claypool. [15](#)
- Kelly, T. P. (1998, Sep). *Arguing Safety – A Systematic Approach to Safety Case Management*. Ph. D. thesis, Dept. of Comp. Sci., University of York, UK. [15](#), [36](#), [38](#), [44](#), [45](#), [107](#)
- Kifer, M., G. Lausen, and J. Wu (1995). Logical foundations of object-oriented and frame-based languages. *Journal of the ACM* *42*(4), 741–843. [15](#)
- Kumamoto, H. (2007). *Satisfying safety goals by probabilistic risk assessment*. Reliability Engineering. Springer. [35](#), [36](#), [110](#)
- Lamport, L. (2002, Jun). *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley. [14](#), [16](#), [29](#), [33](#), [40](#), [42](#), [66](#)
- Layman, L., V. R. Basili, M. V. Zelkowitz, and K. L. Fisher (2011, May). A case study of measuring process risk for early insights into software safety. See [Taylor et al. \(2011\)](#), pp. 623–32. [3](#), [33](#)
- Lee, E. A. and S. A. Seshia (2011). *Introduction to Embedded Systems, A Cyber-Physical Systems Approach*. Berkeley U P. [11](#), [16](#)
- Letier, E. (2001). *Reasoning about Agents in Goal-oriented Requirements Engineering*. Thèse de Doctorat en Sciences Appliquées, Université Catholique de Louvain. [114](#)
- Leucker, M. and C. Schallhart (2009). A brief account of runtime verification. *Journal of Logic and Algebraic Programming* *78*(5), 293–303. [14](#)
- Leveson, N. (2011). The use of safety cases in certification and regulation. *Journal of System Safety* *47*(6), e-Edition. [44](#)
- Leveson, N., J. Reese, and M. Heimdahl (1998). SpecTRM: A CAD system for digital automation. In *17th Digital Avionics Systems Conf.*, Volume 1, pp. B52–1. AIAA/IEEE/SAE. [16](#), [40](#), [43](#)
- Leveson, N. G. (1986). Software safety: why, what, and how. *Computing Surveys* *18*(2), 125–63. [31](#)

- Leveson, N. G. (1995, May). *Safeware: System Safety and Computers*. Amsterdam: Addison-Wesley. 31, 32, 33, 65, 66
- Leveson, N. G. (2012, Jan). *Engineering a Safer World: Systems Thinking Applied to Safety*. Engineering Systems. MIT Press. 1, 3, 4, 31, 32, 35, 36, 39, 43, 65, 106, 107, 113, 114, 141, 146, 147, 148
- Leveson, N. G. and P. R. Harvey (1983a, Sep). Analyzing software safety. *IEEE Trans. Soft. Eng.* 9(5), 569–79. 32, 35
- Leveson, N. G. and P. R. Harvey (1983b). Software fault tree analysis. *J. Syst. and Soft.* 3(2), 173–81. 32
- Leveson, N. G., L. D. Pinnel, S. D. Sandys, S. Koga, and J. D. Reese (1997). Analyzing Software Specifications for Mode Confusion Potential. In *Workshop on Human Error and System Development*, pp. 132–46. 65, 158
- Leveson, N. G. and J. L. Stolzy (1987, Mar). Safety analysis using P<sub>ETRI</sub> nets. *IEEE Trans. Soft. Eng.* 13(3), 386–97. 32, 38, 39, 40, 146, 147
- Li, P. (2014, Apr). Spezifikation des automatischen Testprozesses und Werkzeugentwicklung für sicherheitskritische Antriebsfunktionen in Elektro- und Hybridfahrzeugen. Master's thesis, Technische Universität München. 95
- Liggesmeyer, P. (2009, Jun). *Software-Qualität: Testen, Analysieren und Verifizieren von Software* (2nd ed.). Spektrum. 12, 14, 16, 17, 34, 36, 115, 162
- Lindholm, C., J. P. Notander, and M. Höst (2012). A case study on software risk analysis in medical device development. In S. Biff, D. Winkler, and J. Bergsmann (Eds.), *SWQD*, Volume 94 of *LNBIP*, pp. 143–58. Springer. 38, 45
- Ljung, L. (1998). System identification. In A. Procházka, J. Uhlir, P. Rayner, and N. Kingsbury (Eds.), *Signal Analysis and Prediction*, Applied and Numerical Harmonic Analysis, pp. 163–73. Birkhäuser Boston. 14, 110
- Lochmann, K. and M. Gleirscher (2009, Nov). Adaptive Cruise Control (ACC) – Case Study in UML. Online: [www4.in.tum.de/lehre/vorlesungen/re](http://www4.in.tum.de/lehre/vorlesungen/re), accessed: 2013-02-15. Lecture material, requirements engineering, winter term 2009/2010, Institut für Informatik, Technische Universität München. 9, 93
- Logrippo, L. (2014, May). From sumerian codes to computer code: A formal logic perspective on legal theory and information technology in a historical context. Online: [www.site.uottawa.ca/~luigi/papers/LegalLogicBlog.htm](http://www.site.uottawa.ca/~luigi/papers/LegalLogicBlog.htm), accessed: 2014-05-07. 115
- Luhmann, N. (2006). *Soziale Systeme. Grundriss einer allgemeinen Theorie* (12th ed.). Frankfurt am Main: Suhrkamp. 13
- Luksch, A. (2012). *Gefährdungsbeurteilung richtig machen – Schnelleinstieg in eine zentrale Aufgabe des Arbeitsschutzes*. ecomed Sicherheit. Hüthig Jehle Rehm. 1, 31, 35, 37, 66, 67
- Lund, M. S., B. Solhaug, and K. Stølen (2011, Sep). *Model-Driven Risk Analysis: The CORAS Approach* (1st ed.). Springer. 33, 44, 107, 108, 146, 147

## Bibliography

- Lunze, J. (2010, Jul). *Regelungstechnik 1: Systemtheoretische Grundlagen, Analyse und Entwurf einschleifiger Regelungen* (8th ed.). Lehrbuch. Springer. 1, 12, 13, 15, 65
- Lutz, R. (1993). Analyzing software requirements errors in safety-critical, embedded systems. In *IEEE Int. Symp. Req. Eng.*, pp. 126–33. IEEE. 2, 3
- Lutz, R. (2000). Software engineering for safety: A roadmap. In *Conf. Future of Soft. Eng.*, pp. 213–26. ACM. 2
- Magee, J. and J. Kramer (2006, Jul). *Concurrency: State Models and Java Programs* (2nd ed.). Wiley. 29
- Manna, Z. and A. Pnueli (1995, Aug). *Temporal Verification of Reactive Systems: Safety* (1st ed.). Springer. 14, 28, 33
- Mannering, D., J. G. Hall, and L. Rapanotti (2007). Safety process improvement: Early analysis and justification. In *2nd IET Conf. on System Safety*. 45, 106
- Mariani, L. (2003). A fault taxonomy for component-based software. *Electr. Notes Theor. Comput. Sci.* 82(6), 55–65. 39
- Martinus, M. (2004, Dec). *Funktionale Sicherheit von mechatronischen Systemen bei mobilen Arbeitsmaschinen*. Dissertation, Technische Universität München. VDI Fortschrittsberichte Reihe 12 Verkehrstechnik/Fahrzeugtechnik Nr. 586. 37, 38, 66
- Mayer, W. and M. Stumptner (2007). Model-based debugging – state of the art and future challenges. *Electr. Notes Theor. Comput. Sci.* 174(4), 61–82. 17
- McDermid, J. A. (1986, Jun). Safety and software. *Electronics and Power* 32(6), 440. 32
- McDermid, J. A. (1991). Issues in developing software for safety critical systems. *Reliability Eng. & Sys. Safety* 32(1-2), 1–24. 3, 32, 33
- McDermid, J. A. (2001). Software Safety: Where's the Evidence? In *6th Austral. Worksh. Indust. Experience with Safety Crit. Sys. and Soft.*, Volume 3, Brisbane, Australia, pp. 1–6. 3, 4, 32, 37
- McDermid, J. A. (2002). Software hazard and safety analysis. In *Formal Techniques in Real-Time and Fault-Tol. Sys.*, Volume 2469 of *LNCS*, pp. 23–34. Springer. 1, 3, 4, 32, 35, 39, 65
- McDermid, J. A. and D. Pumfrey (1994). A development of hazard analysis to aid software design. In *9th Ann. Conf. Comp. Assurance: Safety, Reliability, Fault Tolerance, Concurrency and Real Time, Security*, pp. 17–25. IEEE. 32, 34, 38, 39, 42, 107, 146, 147
- McDermid, J. A. and D. J. Pumfrey (2001). Software safety: Why is there no consensus? In *Int. Sys. Safety Conf.*, Huntsville. System Safety Society. Online: <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.4.2608>, accessed: 2014-05-28. 32
- Mealy, G. H. (1955, Sep). A method to synthesizing sequential circuits. *Bell Systems Technical Journal* 34(5), 1045–79. 15, 39, 40
- Mehrpouyan, H. (2011). Model-based hazard analysis of undesirable environmental and components interaction. Master's thesis, Linköpings Universitet. 38, 39, 41, 106, 146, 147



- Mhenni, F., J. Choley, A. Riviere, N. Nguyen, and H. Kadima (2012, Nov). SysML and safety analysis for mechatronic systems. In *13th Int. Workshop on Mechatronics, 9th France-Japan & 7th Europe-Asia Congress on and Research and Education in Mechatronics*, Paris, pp. 417–24. IEEE. [41](#), [146](#), [147](#)
- Milner, R. (1973). Processes: A mathematical model of computing agents. In *Coll. Math. Logic.*, Bristol, England. North-Holland. [15](#), [114](#)
- Moik, A. (1999, May). Strukturierte Erstellung von formalen Sicherheitsmodellen für Automatisierungssysteme mit Sicherheitsverantwortung. In *Workshop Sicherheit und Zuverlässigkeit software-basierter Systeme*, Bad Honnef, Germany. GI. [41](#), [107](#)
- Moore, E. (1956). *Automata Studies*, Chapter “Gedanken-Experiments on Sequential Machines”, pp. 129–53. Princeton U P. [15](#), [39](#)
- Nader, R. (1965). *Unsafe at Any Speed: The Designed-in Dangers of the American Automobile*. Grossman. [1](#), [10](#), [35](#)
- Neogi, N. A. (2002, Feb). *Hazard Elimination using Backwards Reachability Techniques in Discrete and Hybrid Models*. Ph. D. thesis, Massachusetts Institute of Technology, Dept. of Aeronautics and Astronautics. [39](#), [40](#), [107](#), [146](#), [147](#)
- Neumann, P. G. (1995). *Computer-related Risks*. NY, USA: Addison-Wesley. [1](#), [32](#), [65](#)
- NHTSA (2008, Jul). National Highway Traffic Safety Administration (NHTSA) – National Motor Vehicle Crash Causation Survey. Report to Congress DOT HS 811 059, U.S. Department of Transportation (DoT). [2](#)
- Nissanke, N. and H. Dammag (2002). Design for safety in Safecharts with risk ordering of states. *Safety Science* *40*(9), 753–63. [39](#), [42](#), [108](#), [146](#), [147](#)
- Ostroff, J. (1997, May). A visual toolset for the design of real-time discrete-event systems. *IEEE Trans. Control Systems Technology* *5*(3), 320–37. [15](#), [16](#)
- Papadopoulos, Y., J. A. McDermid, R. Sasse, and G. Heiner (2001, Mar). Analysis and synthesis of the behaviour of complex programmable electronic systems in conditions of failure. *Reliability Engineering and System Safety* *71*(3), 229–47. [38](#), [40](#)
- Pappas, G. J. (2003). Bisimilar linear systems. *Automatica* *39*(12), 2035–47. [109](#), [114](#)
- Parnas, D. and J. Madey (1995, Oct). Functional Documentation for Computer Systems. *Science of Computer Programming* *25*, 41–61. [14](#), [15](#), [16](#), [29](#)
- Parnas, D. L., A. J. van Schouwen, and S. P. Kwan (1990, Jun). Evaluation of safety-critical software. *Commun. ACM* *33*(6), 636–48. [3](#), [31](#), [32](#), [33](#), [65](#), [109](#)
- Pasareanu, C. S., R. Pelánek, and W. Visser (2007). Predicate abstraction with under-approximation refinement. *Logical Methods in Comp. Sci.* *3*(1), 1–22. [29](#)
- Paternò, F., C. Mancini, and S. Meniconi (1997). ConcurTaskTrees: A diagrammatic notation for specifying task models. In *IFIP TC13 Int. Conf. on Human-Computer Interaction*, pp. 362–9. [104](#)

## Bibliography

- Paynter, H. M. (1960). *Analysis and Design of Engineering Systems*. MIT Press. [13](#), [14](#), [15](#)
- Peikenkamp, T., A. Cavallo, L. Valacca, E. Böde, M. Pretzer, and E. M. Hahn (2006). Towards a unified model-based safety assessment. In J. Górski (Ed.), *SAFECOMP*, Volume 4166 of *LNCS*, pp. 275–88. Springer. [17](#), [39](#), [42](#), [146](#), [147](#)
- Peled, D., M. Y. Vardi, and M. Yannakakis (1999). Black box checking. In J. Wu, S. T. Chanson, and Q. Gao (Eds.), *FORTE*, Volume 156 of *IFIP Conf. Proc.*, pp. 225–40. Kluwer. [66](#), [115](#)
- Peleska, J. (1996, Mar). Test automation for safety-critical systems: Industrial application and future developments. See [Gaudel and Woodcock \(1996\)](#), pp. 39–59. [66](#), [115](#)
- Perchonok, K. (1972, Jul). Accident cause analysis. Technical Report ZM-5010-V-3, Cornell Aeronautical Lab., Transportation Research Dep., Buffalo, NY, USA. [35](#), [147](#)
- Perrow, C. (1984). *Normal Accidents: Living with High-Risk Technologies*. NY, USA: Basic Books. [1](#), [32](#)
- Petersen, J. (1981). *PETRI-Net Theory and the Modeling of Systems*. Englewood Cliffs, NJ: Prentice-Hall. [15](#)
- Petrenko, A. (2000). Fault model-driven test derivation from finite state models: Annotated bibliography. In *4th Summer School on Modeling and Verification of Parallel Processes*, Volume 2067 of *LNCS*, pp. 196–205. Springer. [17](#)
- Pierre, L. and T. Kropf (Eds.) (1999, Sep). *10th IFIP WG 10.5 Advanced Research Working Conf. on Correct Hardware Design and Verification Methods*, Volume 1703 of *LNCS*, Bad Herrenalb, Germany. Springer. [121](#), [132](#)
- Pister, M. (2008). *Integration formaler Fehlereinflussanalyse in die Funktionsentwicklung bei der Automobilindustrie*. Dissertation, Technische Universität München, Germany. [17](#), [39](#), [41](#), [146](#), [147](#)
- Platzter, A. (2010). *Logical Analysis of Hybrid Systems: Proving Theorems for Complex Dynamics*. Heidelberg: Springer. [16](#)
- Pnueli, A. and Y. Kesten (2002). A deductive proof system for CTL\*. In L. Brim, M. Kretinsky, A. Kucera, and P. Jancar (Eds.), *Concurrency Theory*, Volume 2421 of *LNCS*, pp. 24–40. Berlin Heidelberg: Springer. [17](#), [28](#), [114](#)
- Pock, M. (2012). *A Hierarchical Modelling and Evaluation Technique for Safety Critical Systems*. Dissertation, Technische Universität München. [38](#), [41](#), [65](#), [106](#), [146](#), [147](#), [158](#)
- Probst, S. (1996, May). *Chemical Process Safety and Operability Analysis using Symbolic Model Checking*. Ph. D. thesis, Carnegie Mellon University, Pittsburgh, PA, USA. [39](#), [42](#), [146](#), [147](#)
- Pugliese, R. and E. Tronci (1996, Mar). Automatic verification of a hydroelectric power plant. See [Gaudel and Woodcock \(1996\)](#), pp. 425–44. [29](#)
- Pyle, I. C. (1991). *Developing Safety Systems: A Guide using ADA*. Prentice-Hall. [3](#), [32](#), [33](#), [44](#), [65](#), [66](#), [67](#), [108](#), [109](#), [115](#)

- Rasmussen, J. (1997). Risk management in a dynamic society: A modelling problem. *Safety Science* 27(2-3), 183–213. [1](#), [4](#), [32](#), [43](#), [66](#), [146](#), [147](#)
- Rasmussen, J. (1999, Jul). The concept of human error: Is it useful for the design of safe systems? *Safety Science Monitor* 3(special ed.), 1–3. [4](#)
- Rauzy, A. (2002, Oct). Mode automata and their compilation into fault trees. *Reliability Eng. and Sys. Safety* 78(1), 1–12. [30](#), [39](#), [40](#), [42](#), [146](#), [147](#)
- Reese, J. and N. Leveson (1997). Software deviation analysis. In *ICSE*, pp. 250–60. ACM. [17](#), [41](#)
- Reiter, R. (1987). A theory of diagnosis from first principles. *Artificial Intelligence* 32(1), 57–95. [13](#), [16](#)
- Reiter, R. (2001). *Knowledge in Action: Logical Foundations for Specifying and Implementing Dynamical Systems*. MIT Press. [13](#), [15](#), [29](#), [162](#)
- Roth, M. and P. Liggesmeyer (2013). Qualitative Analyse der funktionalen Sicherheit software-intensiver Systeme mittels Zustands/Ereignis-Fehlerbäumen. In W. A. Halang (Ed.), *Funktionale Sicherheit*, Informatik aktuell, pp. 117–26. Berlin Heidelberg: Springer. [39](#), [40](#), [146](#), [147](#)
- RTCA (2001, Oct). SC-190: DO-248B, Final Annual Report for Clarification of DO-178B “Software Considerations in Airborne Systems and Equipment Certification”. [34](#)
- Rushby, J. (1994). Critical system properties: Survey and taxonomy. *Reliability Engineering & System Safety* 43(2), 189–219. [14](#)
- Rushby, J. (2010, Feb). Formalism in safety cases. In *18th Safety-Critical Systems Symp.: Making Systems Safer*, Bristol, UK. Springer. [44](#)
- Rusu, V., H. Marchand, and T. Jérón (2005). Automatic verification and conformance testing for validating safety properties of reactive systems. In J. Fitzgerald, I. J. Hayes, and A. Tarlecki (Eds.), *FM*, Volume 3582 of *LNCS*, pp. 189–204. Springer. [66](#), [109](#), [115](#)
- Sampath, M., R. Sengupta, S. Lafortune, K. Sinnamohideen, and D. C. Teneketzi (1996). Failure diagnosis using discrete-event models. *IEEE Trans. Control Sys. Tech.* 4(2), 105–24. [15](#), [16](#), [29](#)
- Sargent, R. G. (1999). Validation and verification of simulation models. In *Winter Simulation Conf.*, pp. 39–48. IEEE. [14](#), [110](#)
- Sayre, K., J. Kenner, and P. Jones (2001). Safety models: An analytical tool for risk analysis of medical device systems. In *14th IEEE Symp. on Computer-Based Medical Systems*, Maryland, USA. [39](#), [43](#), [146](#), [147](#)
- Schätz, B. (2008, Jun). Modular functional descriptions. *ENTCS* 215(0), 23–38. 4th Int. Workshop Formal Aspects of Component Software. [73](#)
- Schätz, B., A. Fleischmann, E. Geisberger, and M. Pister (2005). Model-based Requirements Engineering with AutoRAID. In A. B. Cremers, R. Manthey, P. Martini, and V. Steinhage (Eds.), *Informatik: Workshop “Modellbasierte Qualitätssicherung”*, LNI, pp. 511–6. GI: Springer, Bonner Köllen. [3](#)

## Bibliography

- Schneider, P., E. Huck, and P. Schwarz (2001, Aug). A modeling approach for mechatronic systems – modeling and simulation of an elevator system. In *11th Int. Symp. Theo. Electrical Eng.*, Linz, Austria. 14, 16
- Schulz, O. and J. Peleska (2010). Reliability analysis of safety-related communication architectures. In E. Schoitsch (Ed.), *Comp. Safety, Reliability, and Security*, Volume 6351 of *LNCS*, pp. 1–14. Berlin, Germany: Springer. 32, 73
- Secchi, C., M. Bonfè, and C. Fantuzzi (2007, Jan). On the use of UML for modeling mechatronic systems. *IEEE Trans. Automation Sci. and Eng.* 4(1), 105–13. 15, 65
- Shappell, S. and D. Wiegmann (2000). The human factors analysis and classification system – HFACS. Technical Report DOT/FAA/AM-00/7, Office of Aviation Medicine, Civil Aeromedical Institute, Oklahoma City, OK, USA. 1
- Shaw, M. (2002). What makes good research in software engineering? *Int. J. Software Tools for Technology Transfer* 4(1), 1–7. 8, 9
- Sinell, H.-J. and H. Meyer (1996). *HACCP in der Praxis: Lebensmittelsicherheit* (1st ed.). Behr's. 31, 35
- Smith, B. C. (1995). *Computers, Ethics and Social Value*, Chapter “Limits of Correctness in Computers”, pp. 456–69. NJ, USA: Prentice-Hall. 3, 109
- Smith, R. and J. Doyle (1992, Jul). Model validation: a connection between robust control and identification. *IEEE Trans. Automatic Control* 37(7), 942–52. 110
- Snooke, N. and C. Price (2011). Model-driven Automated Software FMEA. In *Ann. Reliability and Maintainability Symp.*, pp. 1–6. IEEE. 41, 42, 65, 146, 147
- Spiegel (2012). Störfälle in deutschen AKW. Online: [www.spiegel.de/wirtschaft/soziales/stoerfaelle-in-deutschen-akw-4000-mal-alarm-a-750889.html](http://www.spiegel.de/wirtschaft/soziales/stoerfaelle-in-deutschen-akw-4000-mal-alarm-a-750889.html), accessed: 2013-02-15. 2
- Spivey, J. M. (2008, Jan). *Understanding Z: A Specification Language and its Formal Semantics* (1st ed.). Cambridge U P. 16
- Stachowiak, H. (1973). *Allgemeine Modelltheorie*. Wien: Springer. 14
- Stålhane, T., O. Daramola, and V. Katta (2012, Jul). Patterns in safety analysis. In *4th Int. Conf. Pervasive Patterns and Applications*, Nice, France, pp. 7–10. IARIA/Thinkmind. 38, 41, 66, 106, 146, 147
- Stringfellow, M. V. (2010). *Accident Analysis and Hazard Analysis for Human and Organizational Factors*. Ph. D. thesis, Massachusetts Institute of Technology, USA. 43
- Struss, P. (2003). *Handbuch der Künstlichen Intelligenz* (4th ed.), Chapter “Modellbasierte Systeme und Qualitative Modellierung”, pp. 431–90. München: Oldenbourg. 12, 13, 15
- Struss, P. and A. Fraracci (2011). FMEA of a Braking System – A Kingdom for a Qualitative Valve Model. In *25th Int. Workshop on Qualitative Reasoning*, Barcelona, Spain. 16, 17, 39, 41, 146, 147

- Stursberg, O., H. Graf, S. Engell, and H. Schmidt-Traub (1998, Aug). A concept for safety analyses of chemical plants based on discrete models with an adapted degree of abstraction. In *4th Int. Workshop on Discrete Event Systems*, Cagliari. 35, 39, 41, 146, 147
- Svedung, I. and J. Rasmussen (2002). Graphic representation of accident scenarios: Mapping system structure and the causation of accidents. *Safety Science* 40(5), 397–417. 35, 38, 146, 147
- Taylor, R. N., H. Gall, and N. Medvidovic (Eds.) (2011, May). *33rd Int. Conf. on Software Engineering*, Honolulu, HI, USA. ACM. 119, 121, 124
- Thielscher, M. (2011). A unifying action calculus. *Artificial Intelligence* 175(1), 120–41. 15
- Thramboulidis, K. and S. Scholz (2010, Sep). Integrating the 3+1 SysML view model with safety engineering. In *IEEE Conf. Emerging Technologies and Factory Automation*, Bilbao, pp. 1–8. IEEE Press. 44, 107
- Trochim, W. M. and J. P. Donnelly (2008). *Research Methods Knowledge Base* (3rd ed.), Chapter “Observation and Measurement”, pp. 56–97. Atomic Dog/Cengage Learning. 8, 14, 110
- Umeda, Y., H. Takeda, T. Tomiyama, and H. Yoshikawa (1990). Function, behaviour, and structure. *Applications of Artificial Intelligence in Engineering* 1, 177–93. 14
- Utting, M., A. Pretschner, and B. Legeard (2012). A taxonomy of model-based testing approaches. *Software Testing, Verification and Reliability* 22(5), 297–312. 3
- van Glabbeek, R. J. (2001). *Handbook of Process Algebra*, Chapter 1. “The Linear Time - Branching Time Spectrum I: The Semantics of Concrete, Sequential Processes”, pp. 3–99. Elsevier. 15
- van Lamsweerde, A. (2009). *Requirements Engineering: From System Goals to UML Models to Software Specifications*. Wiley. 1, 3, 12, 13, 14, 15, 16, 29, 33, 36, 44, 45, 66, 67, 107
- VDI (2004, Jun). VDI 2206: Entwicklungsmethodik für mechatronische Systeme (design methodology for mechatronic systems). 37
- Venkatasubramanian, V., J. Zhao, and S. Viswanathan (2000). Intelligent systems for HAZOP analysis of complex process plants. *Computers & Chemical Engineering* 24(9), 2291–302. 38, 41, 146, 147
- Voge, H. and V. Bunimov (2012, Sep). Verwendung von Zustandsautomaten für die Erstellung von Gefährdungs- und Risikoanalysen. See *informyou* (2012). Slides. 39, 40, 146, 147
- von Bertalanffy, L. (1957). Allgemeine Systemtheorie. *Deutsche Universitätszeitung* 12, 8–12. 13
- Wagner, S., B. Schätz, S. Puchner, and P. Kock (2010). A case study on safety cases in the automotive domain: Modules, patterns, and models. In *21st Int. Symp. on Software Reliability Engineering*, pp. 269–78. IEEE. 4, 45, 107
- Wallmüller, E. (2011, Aug). *Software Quality Engineering: Ein Leitfaden für bessere Software-Qualität* (3rd ed.). Hanser. 12, 14, 16

## Bibliography

- Wang, P. K. C. (1964). Control of distributed parameter systems. *Advances in Control Systems 1*, 75–172. [13](#), [15](#)
- Waters, A. and J. Ponton (1989). Qualitative simulation and fault propagation in process plants. *Chemical Engineering Research & Design 67*(4), 407–22. [14](#), [38](#), [39](#), [41](#), [146](#), [147](#)
- Watson, G. and M. Leadbetter (1964). Hazard analysis. I. *Biometrika 51*(1-2), 175. [33](#)
- WHO (2012). World Health Organization (WHO) – International Classification for Patient Safety (ICPS). Online: [www.who.int/patientsafety/implementation/taxonomy](http://www.who.int/patientsafety/implementation/taxonomy), accessed: 2013-02-15. [35](#), [38](#), [147](#)
- Wiener, N. (1965). *Cybernetics: or, Control and Communication in the Animal and the Machine* (2nd ed.). Cambridge, Mass: MIT Press. [13](#), [14](#)
- Wilson, S. P. and J. A. McDerimid (1995). Integrated analysis of complex safety critical systems. *Comp. J. 38*(10), 765–76. [32](#), [42](#), [44](#), [108](#)
- Wittgenstein, L. J. J. (1922). *Tractatus logico-philosophicus: Logisch-philosophische Abhandlung* (1st ed.). Kegan Paul, Trench, Trubner. [14](#), [170](#)
- Wonham, W. M. (1976). Towards an abstract internal model principle. *IEEE Trans. on Systems, Man, and Cybernetics 6*(11), 735–40. [15](#)
- Yenigün, H., V. Levin, D. Peled, and P. A. Beerel (1999, Sep). Hazard-freedom checking in speed-independent systems. See [Pierre and Kropf \(1999\)](#), pp. 317–20. [39](#), [42](#), [146](#), [147](#)
- Yin, R. K. (2009, Oct). *Case Study Research: Design and Methods* (4th ed.). Applied Social Research Methods. Los Angeles, CA: Sage. [9](#)
- Zave, P. and M. Jackson (1997). Four dark corners of requirements engineering. *ACM Trans. Soft. Eng. Meth. 6*(1), 1–30. [3](#), [16](#), [43](#)
- Zhang, H., W. Li, and W. Chen (2010, Oct). Model-based hazard analysis method on automotive programmable electronic system. In *3rd Int. Conf. on Biomedical Engineering and Informatics*, Volume 7, Yantai, China, pp. 2658–61. [38](#), [42](#), [146](#), [147](#)

## Library, Evidence and Excursions

This chapter contains further data and excursions to underpin the proposed method.

### A.1. Transition System Patterns and Guide Words

The Tables A.1 to A.4 provide a library of transition system patterns and guide words that can be used in the method discussed in the Chapters 4 and 5.

Table A.1.: Patterns for causal factor search based on modes, actions and events; the upper bound  $k$  constrains a past formula to at most  $k$  consecutive states preceding the state  $\sigma_h$  (see Figure 4.3);  $e, e', e_{se}$  are abstract events (Definition 2.11), \*... includes the variations listed in Footnote 2 on page 51.

Guide Word	Pattern for Past Formula to form a $\chi \in \mathcal{H}$	Notes on Modelling
<b>Event Guide Word</b> (with respect to $\mathcal{M}, e', v_e, k, \dots$ ; helpful for the analysis of stimuli or reactions)		
<i>e not given/returned</i>	$\neg \bar{F}^{\leq k} e$	
<i>e incorrectly given/returned</i>	$\bar{F}^{\leq k} (e \wedge v_e \in T_{\text{incorVal}})$	$v_e$ is a variable modelling the event $e$ and $T_{\text{incorVal}}$ is a set of incorrect values.
<i>stimulus/reaction e</i> <i>(or e happened)</i>	$\bar{F}^{\leq k} e$ or “ <i>e incorrectly given</i> ” with $v_e \in \text{mon}(\mathcal{A}_S)$	
<i>start of e</i>	$\bar{F}^{\leq k} (e \wedge \bar{X} \neg e)$	
<i>stop of e</i>	$\bar{F}^{\leq k} (\neg e \wedge \bar{X} e)$	
<i>too much* e</i>	$\bar{F}^{\leq k} (e \wedge v_e > \text{upperbound})$	
<i>too little* e</i>	$\bar{F}^{\leq k} (e \wedge v_e < \text{lowerbound})$	

*Continued on the following page*

## A. Library, Evidence and Excursions

Guide Word	Pattern for Past Formula to form a $\chi \in \mathcal{H}$	Notes on Modelling
<i>wrong timing of e</i>	$\bar{F}^{\leq k}(e \wedge (e' \vee t \in T_{\text{wrongTime}}))$	$e'$ is another event used for orientation.
<i>wrong order of e: e before e'</i> <i>e after e'</i>	$\bar{F}^{\leq k}e \wedge (\neg \bar{F}^{\leq k}e' \vee (\neg \bar{F}^{\leq k}e \bar{U}^{\leq k}e'))$ $\bar{F}^{\leq k}e \wedge (\neg \bar{F}^{\leq k}e' \vee (\bar{F}^{\leq k}e \bar{U}^{\leq k}e'))$	We have $\sim \in \{<, >\}$ where $<$ indicates “too short” and $>$ “too long”.
<i>e applied ... too short (&lt; d)</i> <i>... too long (&gt; d)</i>	$\bar{F}^{\leq k}(\neg e \wedge \bar{X}(e \bar{U}^{-d}\neg e))$	$T_{\text{wrongTime}}$ is a set of wrong time stamps. $t$ models the global timer and $d$ specifies the minimum or maximum duration of an event with $d < k$ .
<i>e stopped too soon (before e')</i> <i>e started ...</i>	$\bar{F}^{\leq k}(\neg e \wedge e' \wedge \bar{X}((\neg e \wedge \neg e') \bar{U}(e \wedge \neg e')))$ $\bar{F}^{\leq k}(e' \wedge \bar{X}((\neg e' \wedge e) \bar{U}\neg e))$	
<i>e stopped too late (after e')</i> <i>e started ...</i>	$\bar{F}^{\leq k}(\neg e \wedge \bar{X}(e \bar{U}(e' \wedge \bar{X}(\neg e'))))$ $\bar{F}^{\leq k}(e \wedge \bar{X}(\neg e \bar{U}(e' \wedge \bar{X}(\neg e'))))$	
<i>e happening/applied too long</i> <i>e happening even permanently</i>	$\bar{F}^{\leq k}(e \bar{U}^{\geq d}\neg e)$ $\bar{G}e \vee (e \bar{U}^{>k}e')$	
<b>Mode Guide Word (with respect to <math>\mathcal{M}</math> and <math>m'</math>; helpful for the analysis of control actions)</b>		
<i>activation of m</i>	$\bar{F}^{\leq k}m_{\mathcal{M}} = m$	$\mathcal{M}$ is an MTS, $m'$ is assumed to model a hazardous mode.
<i>deactivation of m</i>	$\bar{F}^{\leq k}m_{\mathcal{M}} \neq m$	
<i>change of m</i>	$\bar{F}^{\leq k}(\bar{X}(m_{\mathcal{M}} = m) \wedge m_{\mathcal{M}} = m')$ with $m \neq m'$	
<b>Action Guide Word (with respect to <math>\mathcal{M}</math> and <math>(m, a, m') \in \Delta</math>; for the analysis of functional actions)</b>		
<i>unexpected* performance of a:</i> <i>unexpected* execution of a</i>	$\bar{F}^{\leq k}(\bar{X}(\neg \text{enabl} \vee \neg a.\text{pre}) \wedge (a.\text{post} \vee m_{\mathcal{M}} = m'))$	The transform of the function $\text{post} : \mathcal{E}_{\text{ctr}(\mathcal{M})}$ into a PCTL* past formula can carry one $\bar{X}$ for each assignment operator (Definition 2.15). $\text{enabl}$ is a state constraint using, for example, $m_{\mathcal{M}} = m$ , $\text{pre}$ , $\text{trig}$ and $\text{delay}$ . $\vec{c}$ is a constant vector with $\vec{c} \in \text{type}(\vec{v})$ .
<i>unexpected* suppression of a</i>	$\bar{F}^{\leq k}(\bar{X}(\text{enabl} \wedge a.\text{pre}) \wedge \neg(a.\text{post} \wedge m_{\mathcal{M}} = m'))$	
<i>unwanted/defective a</i> <i>(e.g. a = <math>\kappa</math> for hazardous <math>\kappa</math>-completion)</i>	$\bar{F}^{\leq k}(\exists \vec{c} : \bar{X}(a.\text{pre} \wedge \vec{v} = \vec{c}) \wedge \vec{v} \neq \vec{c} \wedge \neg a.\text{post} \wedge m_{\mathcal{M}} = m')$ The vector $\vec{v}$ of variables is assumed to be controllable by $\mathcal{M}$ , particularly, by (the defective) $a.\text{post}$ . See the <i>side effect</i> guide word below.	
<i>maloperated a</i>	This guide word is a variant of <i>unexpected performance of a</i> such that $\text{enabl}$ encodes correct operation.	
<b>Hybrid Guide Word (i.e. combinations of event, mode and action guide words)</b>		
<i>side effect <math>e_{se}</math> of a</i> <i>(e.g. heat, spilled acid or poison, explosion)</i>	$\bar{F}^{\leq k}(a.\text{post} \wedge e_{se})$ or $\bar{F}^{\leq k}e_{se}$	$e_{se}$ is an abstract event typically produced by $M_{\text{fail}}$ . $\sigma_{\alpha}$ is an operational situation.
<i>maloperated complex action A</i> <i>or</i> <i>maloperated function <math>\mathcal{M}</math></i> <i>(e.g. result of security attack)</i>	This guide word is an extension of <i>maloperated a</i> such that $\text{Malop}_{\text{Ef}} \otimes \mathcal{M}_{\text{S}}$ with $\Sigma_0 = \{\sigma \mid \sigma_{\alpha}(\sigma)\}$ encodes complex maloperation.	



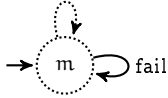
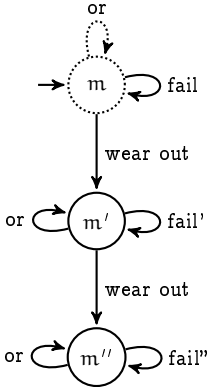
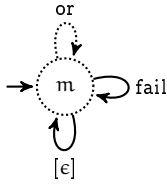
A.1. Transition System Patterns and Guide Words

Table A.2.: Patterns for operational defects to form  $\mathcal{M}_{fail}$ ; † ... occurrences of criteria and facets 1, 2a,d, 3 and 6 of the defect taxonomy; specified (m) and defective ( $m_d$ ) mode, (or)inary action in  $\mathcal{M}_{use}$ , fail for any defective action in  $\mathcal{M}_{fail}$ , ... irrelevant content, \*... wildcard, [a]... action a is part of a pattern variant

Defect Class†	MTS Pattern	Action Specifications and Variants
Systematic defects	<p>where <math>\mathcal{M}_{use}</math> is drawn in dotted lines</p>	<p>Generic part (for a <i>functional action</i> or):  <math>or = (pre, 0, \top, 2, \pi_{or}, post_{or})</math>  <math>fail_1 = (pre, 0, trig_{fail_1}, 1, \pi_{fail_1}, post_{fail_1})</math>  <i>Permanent</i> variant:  <math>fail_2 = (pre_{fail_2}, 0, \top, 1, \_, post_{fail_2})</math>  <math>fail^{-1} = (\perp, \_, \_, \_, \_, \_)</math>  <i>Transient</i> variant:  <math>fail_2 = (pre_{fail_2}, 0, \top, prio, \pi_{fail_2}, post_{fail_2})</math>  <math>fail^{-1} = (\top, 0, trig_{fail^{-1}}, prio, \pi_{fail^{-1}}, post_{fail^{-1}})</math>                      where <math>pre_{fail_2}</math> might be as generic as <math>\top</math>.  <i>Non-deterministic</i> variant:  <math>\pi_{or} = \pi_{fail_1} = 1, trig_{fail^{-1}} = \neg trig_{fail_1}</math>  <i>Probabilistic</i> variant:  <math>\pi_{or} = 1 - \pi_{fail_1}, \pi_{fail_2} = 1 - \pi_{fail^{-1}}, trig_{fail^{-1}} = \top</math>                      This and the following patterns have counterparts for control actions [or] to other modes <math>m'</math> which are analogically applicable and, thus, discussed in low detail. The same holds for probabilistic variants with <math>n</math> alternatives <math>fail_{1.1}</math> to <math>fail_{1.n}</math>.</p>
Random defects	See systematic defects.	<p>Generic part:  <math>or = (pre, 0, \top, 2, \pi_{or}, post_{or})</math>  <math>fail_1 = (pre, 0, \top, 2, \pi_{fail_1}, post_{fail_1})</math>                      For <i>permanent</i> or <i>transient</i> and <i>non-deterministic</i> or <i>probabilistic</i> variants, see <i>systematic defects</i>.</p>
Semi-systematic defects	See systematic defects.	<p><math>fail_1 = fail^{-1} = \tau</math>,  <math>pre_{fail_2}</math> may only refer to variables in <math>\mathcal{V}_e</math>.                      For the other actions, see <i>systematic defects</i>.</p>
Systematic defects (alternative)		<p><math>or = (pre, 0, \top, 2, \_, post_{or})</math>  <math>fail = (pre_{fail}, 0, \top, 1, \_, post_{fail})</math>                      with <math>pre \wedge pre_{fail} \not\leq \perp</math>. Variants are disregarded.</p>
Random, transient defects (alternative)		<p><math>or = (pre, 0, \top, prio, \pi_{or}, post_{or})</math>  <math>fail = (pre, 0, \top, prio, \pi_{fail}, post_{fail})</math>                      For <i>non-deterministic</i>, <i>probabilistic</i> variants, see <i>systematic defects</i>. The action <math>\epsilon</math> is optional.</p>
Common cause defects (1:n)	<p><math>\mathcal{M}_{ccm} = (\otimes_i \mathcal{M}_i) \otimes \mathcal{M}_{def}</math>                      where <math>\mathcal{M}_i =</math></p>	<p>For each dependent function <math>\mathcal{M}_i</math>:  <math>or = (pre \wedge m_{\mathcal{M}_{def}} \neq m_d, 0, \top, prio, \_, post_{or})</math>  <math>fail = (pre \wedge m_{\mathcal{M}_{def}} = m_d, 0, \top, prio, \_, post_{fail})</math>                      with mode channels used in each pre and the defective function <math>\mathcal{M}_{def}</math> applies the pattern <math>\mathcal{M}_{gen}</math> from above.</p>

Continued on the following page

A. Library, Evidence and Excursions

Defect Class†	MTS Pattern	Action Specifications and Variants
Common mode defects (1:n)	See <i>common cause defects</i> .	For each function $\mathcal{M}_i$ , $\text{post}_{\text{fail}}$ encodes similar effects (e.g. loss of power).
Single point defects (1:1)	$\mathcal{M}_{\text{smp}} = (\otimes_i \mathcal{M}_{\text{def}_i}) \otimes \mathcal{M}_{\text{fail}}$ where $\mathcal{M}_{\text{fail}} =$	See <i>common cause defects</i> with $n = 1$ .
Multiple point defects (n:1)		For the failing function $\mathcal{M}_{\text{fail}}$ : $\text{or} = (\text{pre} \wedge \neg \text{pre}_{\text{smp}}, 0, \top, \text{prio}, \_, \text{post}_{\text{or}})$ $\text{fail} = (\text{pre} \wedge \text{pre}_{\text{smp}}, 0, \top, \text{prio}, \_, \text{post}_{\text{fail}})$ with mode channels used in an extended precondition $\text{pre}_{\text{smp}} \equiv \bigwedge_i m_{\mathcal{M}_{\text{def}_i}} = m_d$ and each defective function $\mathcal{M}_{\text{def}_i}$ applies the pattern $\mathcal{M}_{\text{gen}}$ from above.
Cascading defects (m:n)	$\mathcal{M}_{\text{casc}} = (\otimes_i \mathcal{M}_{\text{ccm}_i}) \otimes (\otimes_i \mathcal{M}_{\text{smp}_i})$	This pattern combines the patterns for <i>common cause</i> , <i>common mode</i> , <i>single point</i> and <i>multiple point</i> defects.
Wear out or fatigue (e.g. material ageing, single-purpose mechanisms as in airbags, tiring operators)		<i>Two-stage variant:</i> $\text{or} = (\text{pre}, 0, \top, 2, \pi_{\text{or}}, \text{post}_{\text{or}})$ $\text{fail} = (\text{pre}, 0, \top, 2, \pi_{\text{fail}}, \text{post}_{\text{fail}})$ $\text{fail}'$ and $\text{fail}''$ are similar except for $\pi$ which usually increases. $m'$ and $m''$ indicate worn stages of the mode $m$ . <i>Wear-out-by-time variant:</i> $\text{wearout} = (\top, w, \top, 1, 1, \_)$ $w$ is a time interval to approximate wear out progress. <i>Wear-out-by-condition variant:</i> $\text{wearout} = (\text{pre}_{\text{wo}}, 0, \text{trig}_{\text{wo}}, 1, 1, \_)$ usually with $\text{trig}_{\text{wo}} \equiv \top$ . <i>n-stage variants</i> , and the combination of <i>by-time</i> and <i>by-condition</i> are possible.
Currently unacceptable (delayed) execution or suppression		(Non-deterministic) variant for <i>execution</i> : $\text{or} = (\text{pre}, 0, \top, 2, \_, \text{post}_{\text{or}})$ $\text{fail} = (\text{pre} \vee \text{pre}_{\text{rel}}, 0, \top, 2, \_, \text{post}_{\text{or}})$ without $\epsilon$ , available as a <i>probabilistic</i> variant and for control actions or. <i>Systematically delayed variant:</i> $\epsilon = (\text{pre} \wedge t_{\mathcal{M}} < d, 0, \top, 1.5, 1, \text{NOP})$ $\text{fail} = (\text{pre} \wedge t_{\mathcal{M}} \geq d, 0, \top, 1.5, 1, \text{post}_{\text{or}})$ Note that using $t_{\mathcal{M}}$ is a work-around for the next pattern. Variant for <i>suppression</i> : $\text{or} = (\text{pre}, 0, \top, 2, \_, \text{post}_{\text{or}})$ $\text{fail} = (\text{pre} \wedge \text{pre}_{\text{nar}}, 0, \top, 1, \_, \text{NOP})$ with a relaxed constraint $\text{pre}_{\text{rel}}$ and a narrowed constraint $\text{pre}_{\text{nar}}$ .

Continued on the following page

A.1. Transition System Patterns and Guide Words

Defect Class†	MTS Pattern	Action Specifications and Variants
Delayed execution (alternative)		$or = (pre, 0, \top, 2, \pi_{or}, post_{or})$ $fail_1 = (pre, 0, \top, prio, \pi_{fail_1}, NOP)$ with $prio = 1, \pi_* = 1$ and $m_d$ as the initial mode to form a systematic variant, or $prio = 2$ and $\pi_*$ set individually to form non-deterministic and probabilistic variants as described above for <i>systematic defects</i> . $\epsilon = (\top, 0, \top, 2, \_, NOP)$ $fail_2 = (\top, d, trig_f, 1, \_, NOP)$ $or' = (\top, 0, \top, 2, \_, post_{or'})$ $\tau = (\top, 0, trig_\tau, 1, \_, NOP)$ with an exact ( $trig_f = \top$ ) or minimum ( $trig_f = \perp$ ) delay $d$ for an arbitrary ( $trig_\tau = \perp$ ) or limited ( $trig_\tau \neq \perp$ ) duration.

Table A.4.: Patterns to treat transition system defects and form  $M_{save}$  and  $M_{use}$ ; specified (m), defective ( $m_d$ ) and safe ( $m_s$ ) mode, (re)pair action in  $M_{use}$ , ... irrelevant content

MTS Pattern (solid fragments in $M_{save}$ )	Action Specifications and Variants
Completion via superimposition 	For a <i>functional action</i> or: $or = (pre, 0, \top, prio, \_, post)$ Single-stage, individual variant: $or_c = (\neg pre, 0, \top, prio, \_, post_c)$ such that $\neg pre$ covers <i>unexpected stimuli</i> . Variant with $\epsilon$ -completion: $post_c = NOP$ Control actions can be completed analogically.

Aim: Reduce  $\kappa$ -completion through weakly underspecified or fully specified modes (cf. page 20).

Patterns for treatment of hazardous defects (e.g. permanent systematic or random defect, spurious trip, failure on demand):	
Indeterminisation via superimposition 	For (the partial coverage of) a <i>control action</i> $a$ : $or = (pre, 0, \top, 2, \_, post)$ $a = (pre_a, 0, trig_a, 1, \_, post_a)$ The action $a$ can be operational, defective (i.e. fail) or hazardous. $or_{i1} = (pre_{i1}, 0, trig_{i1}, prio, \_, post_{i1})$ with $pre \wedge pre_a \neq \perp$ and $prio \leq 1$ ( $prio < 1$ can be used to resolve indeterminacy). $recover = (pre_{rec}, 0, trig_{rec}, 1, \_, post_{rec})$ recover can represent repair or maintenance. Defective or hazardous <i>functional actions</i> (see $fail_3$ in pattern <i>fail-safe actions</i> below) can be indeterminised analogically using $or_{i1}$ or an action $or_{i2}$ : $or_{i2} = (pre_{i2}, 0, \top, 1, \_, post_{i2})$

Continued on the following page

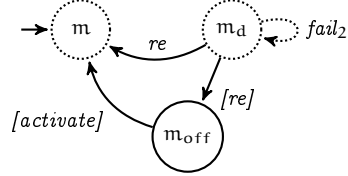
---

MTS Pattern (solid fragments in  $\mathbb{M}_{\text{save}}$ )      Action Specifications and Variants

---

Aim: Turn hazardous defects into detected/perceived defects with sufficient diagnostic coverage (i.e.  $\text{pre}_a \rightarrow \text{pre}_{i1}$  and  $\text{trig}_a \rightarrow \text{trig}_{i1}$  are valid) by either the system or the environment (see criterion 4 in Section 4.2.1).

Repair action  
(usually passive)



From  $\mathbb{M}_{\text{fail}}$  fragment:

$\text{fail}_2 = (\top, 0, \top, 1, \_, \text{post}_{\text{fail}})$

$\mathbb{M}_{\text{save}}$  fragment:

$\text{re} = (\text{pre}_{\text{re}}, 0, \text{trig}_{\text{re}}, \text{prio}, \_, \text{post}_{\text{re}})$

with  $\text{prio} \leq 1$  ( $\text{prio} = 1$  retains indeterminacy).

The action  $\text{re}$  is a variant of the recover action starting from a defective mode and interacting with a tactic  $\mathcal{M}_{\text{repair}}^E$  derived from the associated use case.

Variant with *deactivation*:

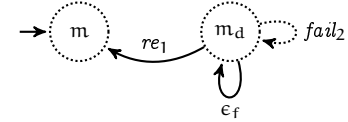
$m$  is revisited via the mode  $m_{\text{off}}$  which deactivates the function.

---

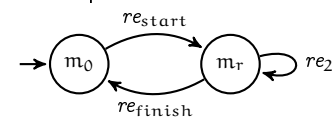
Repair MTS

(usually passive)

$\mathcal{M} = \mathcal{M}_{\text{def}} \otimes \mathcal{M}_{\text{repair}}^S$  where  $\mathcal{M}_{\text{def}} =$



and  $\mathcal{M}_{\text{repair}}^S =$



$\mathbb{M}_{\text{save}}$  fragment of  $\mathcal{M}_{\text{def}}$ :

$\text{re}_1 = (\text{pre}_{\text{re}_1}, 0, \text{pre}_{\text{re}_1}, 0.9, \_, \text{post}_{\text{re}_1})$

$\epsilon_f = (m_{\mathcal{M}_{\text{repair}}^S} = m_r, 0, \top, 0.9, \_, \text{NOP})$

with a state constraint  $\text{pre}_{\text{re}_1}$  such that

$\forall \sigma \in \Sigma \exists \sigma' \in \Sigma : \sigma' |_{\text{tr}(\mathcal{M})} = \text{post}_{\text{re}_1}(\sigma) \wedge \text{pre}_{\text{re}_1}(\sigma')$

$\mathbb{M}_{\text{save}}$  element  $\mathcal{M}_{\text{repair}}^S$ :

$\text{re}_{\text{start}} = (m_{\mathcal{M}_{\text{def}}} = m_d, 0, \text{trig}_{\text{rs}}, \text{prio}, \_, \text{post}_{\text{rs}})$

ideally with  $\text{trig}_{\text{rs}} \equiv \top$  for immediate repair.

$\text{re}_2 = (\text{pre}_{\text{re}_2}, 0, \top, 1, \_, \text{post}_{\text{re}_2})$

interacts with  $\mathcal{M}_{\text{repair}}^E$

$\text{re}_{\text{finish}} = (\text{pre}_{\text{rf}}, \text{delay}, \text{trig}_{\text{rf}}, 0.9, \_, \text{post}_{\text{rf}})$

$m_0$  signifies an inactive mode,  $m_r$  a repair mode.

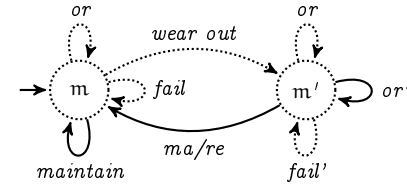
For a *deactivation* variant, see *repair action*.

---

Aim: Repair patterns can be applied as treatments for *wear out* defects (Table A.2) with  $m'$  and  $m''$  instead of  $m_d$  (see below).

Maintenance/repair actions

(preventive or passive)



Single-stage variant for a *function action* or:

$\text{or}' = (\text{pre} \wedge \text{obs}_{\text{wo}}, 0, \top, 1, \_, \text{post}_{\text{or}'})$  such that

$\text{obs}_{\text{wo}}$  can encode material tests and overrule  $\text{or}$ .

$\text{ma}/\text{re} = (\text{pre}_{\text{mr}}, 0, \text{trig}_{\text{mr}}, 0.9, \_, \text{post}_{\text{mr}})$

Preventive self-maintenance variant:

$\text{maintain} = (\text{pre} \wedge \text{obs}_{\text{ma}}, 0, \top, 1, \_, \text{post}_{\text{ma}})$  such

that  $\text{obs}_{\text{ma}}$  can encode maintenance intervals and

overrule  $\text{or}$ .

$n$ -stage variants and control actions or work ana-

logically. For a *deactivation* variant, see *repair action*.

---

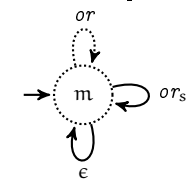
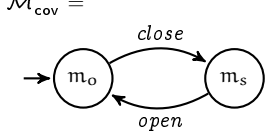
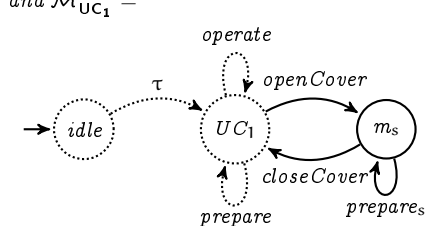
Continued on the following page

## A.1. Transition System Patterns and Guide Words

MTS Pattern (solid fragments in $\mathcal{M}_{\text{save}}$ )	Action Specifications and Variants
<p><i>Fail-safe actions (usually preventive)</i>  <math>\mathcal{M}_1 = \mathcal{M} \oplus \mathcal{M}_{\text{fs}} =</math></p>	<p>Generic part of <math>\mathcal{M}_1</math> for a functional action or:  <math>\text{fail}_3 = (\text{pre}_{\text{fail}_3}, 0, \top, 1, \_, \text{post}_{\text{fail}_3})</math>            For the actions or, <math>\text{fail}_1</math>, <math>\text{fail}_2</math> and <math>\text{fail}^{-1}</math>, see Table A.2.            For the optional re, see the above repair patterns.  <math>\text{fs}_1 = \text{or}_{i1}</math> (see indeterminisation)  <math>\text{fs}_2 = (\top, 0, \top, \text{prio}, \pi_{\text{fs}_2}, \text{post}_{\text{fs}_2})</math>            includes completion or indeterminisation.  <math>\pi_{\text{fs}_2} &lt; 1</math> encodes uncertainty of passive treatment.  <i>Fail-operational variant:</i>  <math>\text{fs}_3 = \text{or}' = (\text{pre}_{\text{or}'}, 0, \top, 1, \_, \text{post}_{\text{or}'})</math>            The action <math>\text{fs}_4</math> can be left out.  <i>Fail-silent variant:</i>  <math>\text{fs}_3 = \text{shutdown} = (\text{pre}_{\text{sd}}, 0, \top, 1, \_, \text{post}_{\text{sd}})</math>  <math>\text{fs}_4 = \epsilon = (\neg \text{pre}_{\text{sd}}, 0, \top, 1, \_, \text{NOP})</math>            shows the completion pattern applied to fail-safe mode <math>m_{\text{fs}}</math>. This pattern works analogically if or is a control action.</p>
<p><i>Fail-operational MTS (usually preventive)</i>  <math>\mathcal{M}_{\text{fo}} = \mathcal{M}_1 \otimes \mathcal{M}_{\text{alt}}</math> where <math>\mathcal{M}_{\text{alt}} =</math></p>	<p>Consider <math>\mathcal{M}_1</math> from above in the fail-silent variant of <math>\mathcal{M}_{\text{fs}}</math>.  <math>\mathcal{M}_{\text{alt}}</math>:  <math>\epsilon = (m_{\mathcal{M}_1} \neq m_{\text{fs}}, 0, \top, 1, 1, \text{NOP})</math>  <math>\text{fo}_1 = (m_{\mathcal{M}_1} = m_{\text{fs}}, 0, \top, 1, 1, \text{post}_{\text{fo}_1})</math> performs the "handover".  <math>\text{fo}_2 = \text{or}' = (\text{pre}_{\text{or}'}, 0, \top, 1, 1, \text{post}_{\text{or}'})</math>  <math>\text{deact} = (m_{\mathcal{M}_1} = m, 0, \text{trig}_{\text{deact}}, 1, 1, \text{post}_{\text{deact}})</math>            where <math>\text{trig}_{\text{deact}} = \top</math> might be convenient. <math>m'</math> indicates a degraded operational mode.</p>
<p><b>Aim:</b> Improve safety by providing MTS patterns for the design patterns FO and FS (cf. page 38).</p>	

*Continued on the following page*

## A. Library, Evidence and Excursions

MTS Pattern ( <i>solid fragments in <math>\mathbb{M}_{\text{save}}</math></i> )	Action Specifications and Variants
Patterns for treatment of <i>hazardous behaviour</i> (e.g. maloperation due to fatigue or unconsciousness):	
<p>Access control mechanism (<i>preventive</i>), for example, a safety cover: <math>\mathcal{M}_1^S = \mathcal{M}_{UC_1}^S \otimes \mathcal{M}_{cov}</math> with <math>\mathcal{M}_{UC_1}^S =</math></p>  <p><math>\mathcal{M}_{cov} =</math></p>  <p>and <math>\mathcal{M}_{UC_1}^E =</math></p> 	<p><math>\mathcal{M}_{UC_1}^S</math> with a functional action or: or = (pre, 0, T, 2, _, post) We use <i>indeterminisation</i> (see above) and <i>priorities</i> to obtain or_s = (pre ∧ m<sub>Mcov</sub> = m_s, 0, T, 1, _, post_s) epsilon = (m<sub>Mcov</sub> = m_o, 0, T, 1, _, NOP) <math>\mathcal{M}_{cov}</math>: close = (pre_c, 0, trig_c, 1, 1, NOP) open = (pre_o, 0, trig_o, 1, 1, NOP) m_o is the mode with granted access (i.e. safety cover opened). <math>\mathcal{M}_{UC_1}^E</math>: prepare = (_, 0, T, 1, 1, pre) prepare_s = (_, 0, T, 1, 1, pre_op) openCover = (_, 0, _, 1, 1, pre_o ∧ trig_o) closeCover = (_, 0, _, 1, 1, pre_c ∧ trig_c) operate = (pre_op, 0, T, 1, 1, pre) The pattern for a control action or works analogously and is, therefore, omitted.</p>
Aim: Assure behavioural safety based on the world model $\mathcal{M} = \mathcal{A}_E \otimes \mathcal{A}_S = \mathcal{M}_1^S \otimes \mathcal{M}_{UC_1}^E$ .	
<p>Attenuation mechanism (<i>passive</i>) (e.g. a car airbag to alleviate collision or obstacle detection to alleviate clamping)</p>	<p>This pattern is described in further detail in Example 4.5 and Section 6.1.</p>
<p>Operator vigilance control (<i>preventive</i>) (e.g. dead man's switch in trains)</p>	<p>This pattern can consist of an additional function <math>\mathcal{M}_{\text{checkAvail}}^S</math> combined with an application of the <i>fail-silent</i> variant of the <i>fail-safe actions</i> pattern (see above).</p>

## A.2. Procedure and Data on Interviews of Safety Practitioners

**Procedure for the Interviews** Table A.5 depicts four *hypotheses* derived from my own research experience and a consolidated *catalogue of initiating questions* applied throughout the interviews to collect evidence or rebuttals. The interviewees have been found using a multi-stage contact approach across various organisations and technical domains mainly within the automotive and commercial road vehicle industries. The safety practitioners were chosen according to their knowledge and expertise in machine and functional safety, hazard analysis and risk assessment (HARA), FMEA

### A.3. Data on the Systematic Map of Related Work

and system reliability assessment. Precedence was given to the practitioners with the longest experience. Table A.6 provides information about the nine interviewees. For their preparation, the selected safety practitioners were provided with information about the background and the kind of the interview, and a preliminary list of topics and questions. Each of the nine interviews has been conducted in German language between June and August 2012. For each interview, the questions from Table A.5 are adapted and issued occasionally depending on previous answers. Answers and observations, which indicate a confirmation or refutation of a hypothesis, are extracted and translated into English. Finally, the interpreted results are used to strengthen the motivation for the present work.

**Results and Interpretation** Table A.7 summarises confirming and refuting answers to the questions underlying the hypotheses 1 to 4. The results are interpreted in Section 1.1.

### A.3. Data on the Systematic Map of Related Work

The Tables A.8 and A.9, and Figure 3.1 represent a *systematic map* of related approaches according to their *abstraction* for building a system model (Section 3.4.1), their *reasoning* about causal factors and effects (Section 3.2), and their underlying *formalism* (Section 2.2.1).

### A.4. Data on the Comparison with Other Procedures

Table A.10 shows a stepwise comparison of the method described in Chapter 5 with the approaches of Leveson (2012) and Kath and Temple (2012).

### A.5. Data on the Automated Teller Machine Case

The Tables A.11 and A.12 contain data for understanding the behavioural properties of the automated teller machine.

**Note on Representation** For the actions described in the Tables A.12 and A.14, Broy and Stølen (2001) show a more concise tabular representation.

### A.6. Data on the Commercial Road Vehicle Case

The Tables A.13 and A.14 contain data for understanding the behavioural properties of the commercial road vehicle.

### A. Library, Evidence and Excursions

State Guide Word	Physical Phenomena and Types to form variables in $\mathcal{V}$	Pattern to form a state constraint $\phi \in \Phi$
<i>contaminate</i>	{low, high} <i>density</i> of a hazardous substance in an area entered by persons	$\text{density}_{\text{Area, Substance}} = \text{high} \wedge \text{entered}_{\text{Person, Area}}$
<i>collide, crash</i> (three variants)	{0, short, medium, long} <i>distance</i> , {low, high} <i>relative speed</i> and {narrow, wide} <i>relative speed angle</i> of manned objects moving in an area; <i>Remark</i> : more precise but more expensive in state observation	$\text{crashed}_{\text{Obj}_s} \equiv \text{distance}_{\text{Obj}_s} = 0 \wedge \text{relativespeed}_{\text{Obj}_s} = \text{high} \wedge \text{relativespeedangle}_{\text{Obj}_s} = \text{narrow}$
	<i>Remark</i> : more directly based on object properties ( <i>position, speed</i> ), less precise capture of the spectrum of situations	$\text{crashed}_{\text{Obj}_s} \equiv  \text{position}_{\text{Obj}_1} - \text{position}_{\text{Obj}_2}  < \text{short} \wedge  \text{speed}_{\text{Obj}_1} - \text{speed}_{\text{Obj}_2}  > \text{low}$
	{low, strong} <i>impact</i> and <i>deformation</i> of a manned object; <i>Remark</i> : suggests state observation done by the system, more precise in timing, less precise capture of the spectrum of situations	$\text{crashed}_{\text{Obj}} \equiv \text{impact}_{\text{Obj}} = \text{strong} \wedge \text{deformed}_{\text{Obj}} \wedge \text{entered}_{\text{Person, Obj}}$
<i>distract</i>	a person <i>distracted</i> by an event in a situation within an area	$\text{someEventInArea} \wedge \text{someSituation} \wedge \text{entered}_{\text{Person, Area}}$
<i>shoot</i>	a person <i>shot</i> by an object flying fast through an area	$\text{flyingFastThroughObject, Area} \wedge \text{entered}_{\text{Person, Area}}$
<i>hit</i>	a person <i>hit</i> by an object moving fast at a certain place	$\text{movingFastAtObject, Place} \wedge \text{adjacentTo}_{\text{Person, Place}}$
<i>bump (opposite of hit)</i>	a moving person <i>bumping</i> into a fixed object at a certain place	$\text{movingFastAtPerson, Place} \wedge \text{adjacentTo}_{\text{Object, Place}}$
<i>fall</i>	<i>fall</i> of a person at a certain place	$\text{fellAt}_{\text{Person, Place}}$
<i>clamp, squeeze</i>	{wide, narrow, closed} <i>position</i> of certain system parts occupied by a person's body parts (e.g. hands)	$\text{position}_{\text{MachineParts}} \neq \text{wide} \wedge \text{occupiedBy}_{\text{MachineParts, BodyParts}}$
<i>shear</i>	{opened, closed} <i>position</i> of a machine's parts occupied by a person's body parts	$\text{position}_{\text{MachineParts}} = \text{closed} \wedge \text{occupiedBy}_{\text{MachineParts, BodyParts}}$
<i>burn</i>	{low, high} <i>temperature</i> of a contact surface touched by a person's body parts	$\text{temperature}_{\text{ContactSurface}} = \text{high} \wedge \text{touchedBy}_{\text{ContactSurface, BodyParts}}$
<i>electrically shock</i>	{none, low, high} <i>voltage</i> of a contact surface touched by a person's body parts	$\text{voltage}_{\text{ContactSurface}} = \text{high} \wedge \text{touchedBy}_{\text{ContactSurface, BodyParts}}$

Table A.3.: Patterns for state constraints to model  $\Phi$



A.6. Data on the Commercial Road Vehicle Case

No.	Hypothesis	Core Initiating Questions
1	The <i>societal demand</i> for system safety increases steadily.	a. How did safety needs and practices change in your application/system domain over the last few years?
2	Systems and safety engineering <i>processes</i> lack integration; <i>roles</i> of safety and reliability engineers are misaligned; the <i>responsibility</i> for safety assurance is mainly split by technology domains.	a. How would you describe your role, tasks and competences in your organisation, for example, your relationship to requirements, system test and design engineers or suppliers? b. Which are the boundaries defining your task of safety assurance? c. How do your results contribute to a conclusion on the safety of a system? d. Did you ever recognise, for example, hazards, systematic faults or specification defects unacceptably late?
3	Commonalities and differences of application domains are known but <i>standards</i> are neither harmonised nor mature.	a. Which kinds of hazards and defects are essential and decisive for your application/system domain? b. Which deficiencies do you perceive in the standards you follow?
4	Safety analysis practice lacks <i>guidance</i> : interdisciplinary <i>system models</i> can aid and reduce <i>incompleteness</i> in safety engineering but are hardly used.	a. Which deficiencies do you perceive in your methods and tools? b. How would you improve your procedure, for example, to consider maloperation or to support reuse, volatile or changing requirements and design decisions as well as functional and technological variety?

Table A.5.: Catalogue of questions underlying the hypotheses and guiding the interviews

A. Library, Evidence and Excursions

IV	Date	Applications / Technology Domains	Expertise / Educational Background
1 J	5.6.12	Car driver assistance, agricultural machines / automotive microelectronics	Functional safety consulting, audits and coaching / informatics, software engineering
2 S	14.6.12	Car driving dynamics / automotive microelectronics	Functional safety, hazard and risk analysis / informatics
3 J	14.6.12	Commercial road vehicles: driver assistance, chassis control, airbags / control electronics	Electronics reliability assessment / automotive engineering
4 S	15.6.12	Automotive control, avionics / control electronics	Functional safety process consulting, HARA conduct, tool development and qualification / informatics, software eng.
5 J	20.6.12	Commercial road vehicles and others / control electronics	Safety engineering consulting and assessments / electrical and control eng.
6 E	25.6.12	Road vehicles, plants and machinery / control electronics	Functional safety assessments and consulting / informatics
7 S	2.7.12	Road and railway traffic control / control microelectronics	Safety methods research for computer-aided chip design / electrical engineering
8 E	9.7.12	Automotive, flight and turbine control / control electronics	Functional safety assessments / electrical engineering
9 E	8.8.12	Automotive control, avionics / electronics and software	Functional safety assessments and consulting, standardisation / unknown*

Table A.6.: Overview of interviews (IV) and safety practitioners; J... junior safety engineer ( $\leq 2$  years of dedicated safety practice), S... senior safety engineer ( $\leq 6$  yrs.), E... safety expert ( $> 6$  yrs.), \*... disregarded during interview

H (C)onfirming and (R)efuting Answers (from the interview transcripts)	
1	C: (IV5) Increasing demand for safety measures in end-user products. (IV1) Functional safety is finding its way into the automotive industry because of product liability and warranty issues. (IV2,4,6) Producers' pressure (QM) and responsibility to avoid increasing or even lost liability claims raises the demand for safety engineers. (IV7) High automation requires strong safety, reliability, fault tolerance and robustness and, thus, higher cost. (IV3) We face such cost conflicts with suppliers.
2	C: (IV9) A vehicle design is decomposed into safe subsystems, the system as a whole is then neglected in analysis. (IV3) Reliability assessments are highly specific to technical parts. (IV4) Process change (e.g. standard or method adoption) is hard to achieve. The lack of methods makes independency proofs and SIL decomposition difficult for novel functions. (IV6) More strongly separate HARA from functional safety to improve interdisciplinary assurance. Underspecified roles/tasks of safety engineering teams for system-subsystem settings hinder cross-technology assurance.
	R: (IV2,4,9) Suppliers, technology-specific teams and a central team for functional safety are coordinated to perform, for example, HARA, RCA and management of safety requirements. (IV3) FMEA delegation and reviews for supplied parts work fine.
3	C: (IV1) Applied standards do not entail safety, e.g. ISO 26262 can be infeasible due to limited observability of operational situations; no guidance for <i>sensor abstractions</i> . (IV2) ISO 26262 neglects <i>driver maloperation</i> . (IV4,7) Due to fast technology evolution, standards and methods lack <i>clear-cut guidance</i> and <i>harmonisation</i> , e.g. AutoSAR or DO-182 with ISO 26262 recommendations for SW, SW/HW interfaces and decomposition. (IV4) Safety practices in avionics and truck industry are intensive and mature. Automotive supply chains still need to improve, e.g. by adopting ISO 26262. (IV8) Automotive industry focuses single point <i>failures</i> whereas aeronautics also handles multiple point failures or latent faults. (IV5) Methods from aeronautics hardly fit road vehicle needs because controllability assumptions about pilots are stronger than those about usual drivers. (IV6) Machinery guidelines for HARA are more precise than ISO 26262 part 3, e.g. "access and working areas" are better understood than "driving situations." Hence, mere adoption of risk graphs across several domains is complicated.
	R: (IV9) Applying the ISO 26262 part 3 (HARA) works fine.
4	C: (IV3) Reliability models are highly technology specific. (IV9) In the automotive industry, system models are less used than individual empirical methods and expert judgements. (IV4,5,6) Systematic HARA modelling and reuse is neither practiced nor standardised; cf. H2-C-IV6. Demand for models as reusable documentation exists. (IV8) Usual defect taxonomy is only partially captured by the regarded models. Interdisciplinary modelling and collaboration is neglected. (IV2) Unification of driving situation registers and damage severity classes needed across collaborating business units within a supply chain. (IV4) Underuse of common driving situation registers.
	R: (IV3,4,6) Customised spreadsheets and tools mostly serve the needs. (IV8) Physical simulations are expensive but effective. (IV7) Interface design models are used for documentation and integration. (IV2) Used HARA and FMEA methods are satisfying. Interdisciplinary abstractions are unqualified for severity estimations. (IV9) Models of the complete vehicle are neither required nor possible; cf. H3-R-IV9. (IV4) Assumptions on driver maloperation are documented in the guidelines.

Table A.7.: Confirmations and refutations, H... hypothesis, IV... interview

A. Library, Evidence and Excursions

Abstraction (Sections 3.2 and 3.4.1)	Main Direction of Causative Reasoning (Sections 3.2 and 3.4.2)		
	deductive	linear causal chain inductive	non-linear chain (interactive †)
implicit (not prescribed, informal)	static FTA, RCA (Cacciabue 2004)	FMEA, ETA, LOPA, HACCP, HAZOP, ECF (Buys and Clark 1995), CRIOP (Johnsen et al. 2011), Stålhane et al. (2012), HERMES, ICPS	Hollnagel (2004), [CRIOP, HAZOP, HERMES, ICPS]‡
structural (only glass-box)	top-down: Biehl et al. (2010), Chen et al. (2008), Svedung and Rasmussen (2002)	bottom-up: Bowles and Wan (2001), Catino and Ungar (1995), Mehrpouyan (2011), Pock (2012), Snooke and Price (2011), Waters and Ponton (1989)	none found; kind of model maybe inappropriate
behavioural (only black-box)	backward: Neogi (2002), Voge and Bunimov (2012)	forward: none found; kind of model maybe inappropriate	D'Ippolito et al. (2011), this work
mixed (modular, grey-box)	dynamic FTA (Dehlinger and Dugan 2008, Dugan et al. 1992), Damm and Peikenkamp (2004), Rauzy (2002), Roth and Liggesmeyer (2013)	Bitsch et al. (1999), David et al. (2010), Dobi et al. (2013), Esser and Struss (2007), Mhenni et al. (2012), Pister (2008), Struss and Fraracci (2011), Stursberg et al. (1998), Venkatasubramanian et al. (2000)	Hall and Silva (2008), STAMP (Leveson 2012), Risklog (Johnson 1993), this work
	bidirectional: Gärtner (1999), Haxthausen et al. (2011, 2014), Heitmeyer et al. (1998), Herrmann and Krumm (2000), Leveson and Stolzy (1987), Nis-sanke and Dammag (2002), Peikenkamp et al. (2006), Probst (1996), CORAS (Lund et al. 2011)		

Table A.8.: Overview and classification of literature on approaches to hazard analysis; model subject disregarded; † ... environment is part of the model, ‡ ... possible

A.6. Data on the Commercial Road Vehicle Case

Formalism† (Section 2.2.1)	Focused Area of Causative Reasoning‡ (Sections 3.2 and 3.4.2)	
	causal factor $\hookrightarrow$ hazard	causal factor or hazard* $\hookrightarrow$ mishap
unclassified**	Perchonok (1972), ICPS (WHO 2012)	
implicit (not prescribed, informal, usually qualitative)	ETA, FMEA, static FTA, HACCP, LOPA, RCA (Cacciabue 2004)	AcciMaps (Svedung and Rasmussen 2002), CRIOP (Johnsen et al. 2011), ECF (Buys and Clark 1995), FMECA, HAZOP, HERMES, FRAM (Hollnagel 2004), Rasmussen (1997), Sayre et al. (2001)
non-timed constraint or flow analysis based on variables, relations or Boolean logic	qualitative: Biehl et al. (2010), Catino and Ungar (1995), Stålhane et al. (2012), Struss and Fraracci (2011), Zhang et al. (2010); quantitative: Bowles and Wan (2001), Chen et al. (2008), Feather (2004), McDermid and Pumfrey (1994), Snooke and Price (2011); multiple: Pock (2012)	qualitative: Hall and Silva (2008), Waters and Ponton (1989); multiple: Mehrpouyan (2011)
timed or temporal logic assertion checking based on transition systems, MARKOV chains or PÉTRI nets	qualitative: D'Ippolito et al. (2011), Esser and Struss (2007), Probst (1996); quantitative: Yenigün et al. (1999); multiple: dynamic FTA (Dehlinger and Dugan 2008, Dugan et al. 1992), Bitsch et al. (1999), David et al. (2010), Gärtner (1999), Haxthausen et al. (2011, 2014), Heitmeyer et al. (1998), Herrmann and Krumm (2000), Johnson (1993), Mhenni et al. (2012), Neogi (2002), Nissanke and Dammag (2002), Peikenkamp et al. (2006), Pister (2008), Rauzy (2002), Roth and Liggesmeyer (2013), Stursberg et al. (1998), Voge and Bunimov (2012)	qualitative: Dobi et al. (2013), Venkatasubramanian et al. (2000), <b>this work</b> ; multiple: Leveson and Stolzy (1987), STAMP (Leveson 2012), CORAS (Lund et al. 2011)

Table A.9.: Overview and classification of literature regarding the formalism and reasoning distance used; †... for causal factor, hazard, mishap; ‡... qualitative, quantitative and multiple abstractions distinguished, direction of reasoning ignored (see Table A.8); \*... only few approaches focus short causal chains between hazards and mishaps; \*\*... information unavailable or insufficient

A. Library, Evidence and Excursions

This Work (see Chapter 5)	STAMP (SpecTRM) (Leveson 2012)	ISO 26262 applied (Kath and Temple 2012)
1. Specify functionality and property assertions	1. Identify mission goals, requirements and constraints	1. Item definition: functions
2. Derive defect model	N/A (due to a lack of concepts, only considered informally)	1. Item definition: malfunctions
3. Identify potential mishaps	2. Define system accidents or unacceptable losses	
4. Assess causal factors and specify hazards	N/A (due to triple comparison)	2. HARA: definition and classification of safety goals
	3. Define mission hazards	N/A (due to the neglect of driver maloperation)
	8. Identify hazardous control actions and their causal factors, derive requirements to treat these actions	
5. Specify safety goals and A/G pairs, assign probability bounds	N/A (due to triple comparison)	2. HARA: definition and classification of safety goals
	4. Define mission-level safety-related constraints	3. Functional safety concept: requirements identification, functional architecture design, ASIL assignment
6.2. Resolve responsibility misperceptions	5. Identify environmental constraints and assumptions, customer-derived system design and programmatic constraints	
6.1 Plan and design safety measures	6. Perform a functional decomposition	
N/A (due to the behavioural view)	7. Design high-level system control structure	4. Definition of technical safety concept, hardware/software design and integration, estimation of component failure rates
6.3 Check validity of safety measures	9. Define system component specifications until hazards are eliminated, mitigated or controlled	5. Safety validation: architecture verification, reliability assessments
	10. Perform validation tests	
N/A (due to focus on the concept phase)	11. Generate designs and software code	N/A (due to exclusion from their case study)

Table A.10.: A comparison of three safety engineering methods; N/A ... unavailable

Table A.11.: Property assertions  $\Gamma$ ; informal description from  $\mathcal{R}_p$ , reduced to safety-related assertions, see Figure 6.5

Id.	Short Description	Description	Step(s)
<i>G9</i>	Safe interaction	The customer shall never be injured by an action of the ATM. The ATM reduces risks of injuries of the bank customer, even if used improperly.	prereq, 1.5, 5.1
<i>M7</i>	Squeezed hands	The customer's hands or fingers get squeezed in the cash slot when interacting with the ATM.	3.2
<i>M13</i>	Electric shock	Hazardous residual current runs through the casing while the bank customer touches the ATM surface.	3.2
<i>M15</i>	Cuts or scratches	The bank customer hurts himself by touching the ATM surface.	3.2
<i>H26</i>	Unexpected closure	The cash slot lid randomly closes and opens or it closes too early.	4.2
<i>D24</i>	Physical obstacles	Hands staying in the cash slot count as an obstacle for the cash slot mechanism.	3.3
<i>Gr10</i>		If a hand is resting in the cash slot, a closing lid shall only apply forces less than 70 N. If obstacles are detected in the opened cash slot, the lid shall stay open.	5.1 (AS <sub>s</sub> )
<i>Gr14</i>		The whole casing has to be protected against residual current.	5.1
<i>Gr16</i>		The ATM surface has to be free of sharp edges or static clamping zones.	5.1
<i>D8</i>	Card position	The variables (i.e. $\text{position}_{\text{card}}$ and $\text{status}_{\text{cardslot}}$ ) pertaining a unique EC card's position have to be kept consistent.	5.1
<i>As19</i>	Proper use	The bank customer keeps his or her hands in the cash slot for less than 5 s.	5.1
<i>As27</i>	Controllability	The bank customer can pull back his or her hands fast enough if they are still in the cash slot or the lid closes unexpectedly.	6.2

Table A.12.: Action specifications for each comprising MTS of the ATM world according to the Figures 6.1, 6.2 and 6.6; the function `rand` returns some  $n \in \mathbb{N} \setminus \{0\}$

Label	pre	delay	trig	prio	$\pi$	post
Tactic "withdraw cash" $WC_E \in M_{use}$						
<code>insert</code>	<code>position_card = inHands</code>	0	T	3	1	<code>position_card := atSlot, status_cardslot := atSlot</code>
<code>select</code>	<code>content_display = servOptions</code>	0	T	3	1	<code>selection_panel := withdraw</code>
<code>chsAmount</code>	<code>content_display = withdrOptions</code>	0	T	3	1	<code>selection_panel := amount(rand())</code>
<code>grabMoney</code>	<code>lid_cashslot = open</code>	0	T	3	1	<code>position_hands := inCashslot</code>
<code>wait (<math>\epsilon</math>)</code>	T	0	T	3	1	NOP
<code>takeMoney</code>	<code>lid_cashslot = open</code>	0	$\perp$	3	1	<code>position_hands := awayFromATM, position_cash := inHands</code>
$\tau$	T	0	$\perp$	3	1	NOP
Function "Withdraw Cash" $WCs \in M_{use}$						
<code>chgDsp</code>	T	0	<code>selection_panel = withdraw</code>	3	1	<code>content_display := withdrOptions</code>
<code>ejectCrd</code>	<code>mid<sub>S</sub> = authenticated</code>	0	<code>selection_panel = amount(A) <math>\wedge</math> A <math>\in</math> <math>\mathbb{N}</math></code>	3	1	<code>status_cardslot := atSlot, position_card := atSlot, amount_tempstore := A</code>
<code>disburse</code>	<code>status_cardslot = noCard</code>	0	T	3	1	<code>content_display := withdrRequest, content_cashslot := cash(amount_tempstore)</code>
<code>reset</code>	T	0	<code>content_cashslot = empty</code>	3	1	<code>content_display := welcScreen, balance_account := amount_tempstore</code>
<code>drawIn</code>	T	0	<code>content_cashslot <math>\neq</math> empty <math>\wedge</math> twcs <math>\geq</math> 5</code>	3	1	<code>content_display := warnScreen, balance_account := amount_tempstore</code>
Function "Services" $SRVs \in M_{use}$						
<code>display1</code>	T	0	T	3	1	<code>content_display := welcScreen</code>
<code>chgDsp</code>	T	0	<code>status_cardslot = atSlot</code>	3	1	<code>content_display := servOptions, status_cardslot := pulledIn, position_card := inATM</code>
<code>display2</code>	T	0	T	3	1	<code>content_display := servOptions</code>
<code>deny</code>	<code>mid<sub>S</sub> = invalid</code>	0	T	3	1	<code>content_display := warnScreen</code>
Tactic "Identify" $IDE \in M_{use}$						
<code>enterPin</code>	T	0	T	3	1	<code>selection_panel := pin(rand())</code>

*Continued on the following page*



A.6. Data on the Commercial Road Vehicle Case

Label	pre	delay	trig	prio	$\pi$	post
Function "Identify" $ID_S \in M_{Use}$						
initiate	$mWCS = withdrawal$	0	$\perp$	3	1	$content_{display} := pinRequest$
retry	$selection_{panel} \neq pin_{card}$	0	$\equiv pre$	3	1	$content_{display} := pinRepeat$
validate	$selection_{panel} = pin_{card}$	0	$\equiv pre$	3	1	NOP
reset	T	0	$msRV_S = welcome$	3	1	NOP
Tactic "Depose" $DEPE \in M_{Use}$						
selectDep	$content_{display} = servOptions$	0	T	3	1	$selection_{panel} := depose$
insCash	$content_{display} = disposeRequest \wedge lid_{cashslot} \neq closed$	0	$\perp$	3	1	$content_{cashslot} := cash(rand()), position_{hands} := inCashslot$
leaveATM	T	0	T	3	1	$position_{hands} := awayFromATM$
wait ( $\epsilon$ )	T	0	T	3	1	
takeCard	$position_{card} = atSlot$	0	T	3	1	$position_{card} := inHands$
Function "Depose" $DEP_S \in M_{Use}$						
chgDsp	$mID_S = identifiedAsCustomer$	0	$selection_{panel} = depose$	3	1	$content_{display} := disposeRequest$
prepare	$lid_{cashslot} = open \vee content_{cashslot} \neq empty$	0	$tDEP_S \geq 5$	3	1	$content_{display} := procDeposit$
procCash	$mLID_S = closed \wedge content_{cashslot} = cash(Y) \wedge Y \in \mathbb{N}$	0	T	3	1	$content_{cashslot} := empty, balance_{account} += Y, content_{deposit} += cash(Y)$
ejectCrd	$content_{cashslot} = empty$	0	T	3	1	$status_{cardslot} := atSlot, position_{card} := atSlot, content_{display} := welScreen$
Function "Open/close Lid" $LID_S \in M_{Use}$						
openLid <sub>1</sub>	$mWCS = "waiting(removal)" \vee mDEP_S = "waiting(disposal)"$	0	T	3	1	NOP
openLid <sub>2</sub>	$lid_{cashslot} \neq open$	0	T	3	1	$lid_{cashslot} += 1$
finish <sub>1</sub>	$lid_{cashslot} = open$	0	T	3	1	NOP
closeLid <sub>1</sub>	$tWCS \geq 2 \vee msRV_S = welcome \vee mDEP_S = processing$	0	T	3	1	NOP
closeLid <sub>2</sub>	$lid_{cashslot} \neq closed$	0	T	3	1	$lid_{cashslot} -= 1$
finish <sub>2</sub>	$lid_{cashslot} = closed$	0	T	3	1	NOP

Continued on the following page

Label	pre	delay	trig	prio	$\pi$	post
<b>LIDs <math>r \in \mathbb{M}_{\text{fail}}</math> ... Defective Lid</b>						
wait <sub>1</sub> ( $\epsilon$ )	T	0	T	1	.5	NOP
openLid <sub>3</sub> ( $\tau$ )	T	0	T	1	.5	NOP
revert ( $\tau$ )	T	0	$\perp$	1	1	NOP
wait <sub>2</sub> ( $\epsilon$ )	T	0	T	1	.5	NOP
closeLid <sub>3</sub>	T	0	T	1	.5	NOP
halt ( $\tau$ )	T	0	$\perp$	1	1	NOP
<b>ASes <math>\in \mathbb{M}_{\text{save}}</math> ... Avoid Squeeze controlled by Environment</b>						
wait <sub>2</sub> ( $\epsilon$ )	$(\text{lid}_{\text{cashslot}} = \text{closing} \vee \text{tASEs} \geq 4) \wedge \text{position}_{\text{hands}} = \text{inSlot}$	0	T	0.9	0.1	NOP
remove-Hands	$(\text{lid}_{\text{cashslot}} = \text{closing} \vee \text{tASEs} \geq 4) \wedge \text{position}_{\text{hands}} = \text{inSlot}$	0	T	0.9	0.9	$\text{position}_{\text{hands}} := \text{awayFromATM}$
wait <sub>3</sub> ( $\epsilon$ )	$\text{lid}_{\text{cashslot}} \neq \text{closing} \vee \text{position}_{\text{hands}} \neq \text{inSlot}$	0	T	0.9	1	NOP
$\tau$	T	0	$\perp$	0.9	1	NOP
<b>ASes <math>\in \mathbb{M}_{\text{save}}</math> ... Avoid Squeeze for Safe Lid</b>						
closeLid <sub>4</sub>	$\text{twCS} \geq 2 \wedge \neg \text{position}_{\text{hands}} = \text{inCashSlot}$	0	T	0.9	.999	NOP
safeRevert	$\text{position}_{\text{hands}} = \text{inCashSlot} \vee \text{lidpressure}_{\text{cashslot}} \geq 70N$	0	T	0.9	.999	NOP

Table A.13.: Property assertions  $\Gamma$ ; informal description as imported from  $\mathcal{R}_p$  and results of further steps

Id.	Short Description	Description	Step(s)
$G1$	Maintenance by customer	The customer is able to conduct maintenance tasks.	prereq.
$G2$	Tilt driving cab		prereq.
$G3$	Drive through car wash		prereq.
$G4$	Use gas station		prereq.
$G5$	Park at a steep hill	The truck is parked at a steep hill.	prereq., 1.1, 3.3
$G6$	Sleep in driving cab		prereq.
$G7$	Drive through city traffic Stop&Go		prereq., 3.3
$G8$	Drive through crossing		prereq.
$G9$	Drive at dense heterogeneous traffic	Dense traffic with unprotected traffic participants on the lane (e.g. pedestrians, cyclists).	prereq., 3.3
$G10$	Use brakes	The trucker is able to use the brake, for example, at traffic lights and stop signs.	prereq., 1.1
$G11$	Bring up to speed	The trucker conducts an intended driveaway.	prereq.
$G12$	Drive backwards		prereq.
$G13$	Drive through roundabout		prereq.
$G14$	Drive at highways or motorways	Usual drive with or without oncoming traffic, large turns.	prereq.
$G15$	Drive in a platoon		prereq.
$G16$	Enter a highway	Driveway, acceleration lane, merging.	prereq.
$G17$	Exit a highway		prereq.
$G18$	Brake because of a traffic jam		prereq.
$G19$	Drive at dense traffic	Dense traffic at a speed of about 80km/h.	prereq.

*Continued on the following page*

A. Library, Evidence and Excursions

Id.	Short Description	Description	Step(s)
$G_{20}$	Change lane		prereq.
$G_{21}$	Drive or park at $\mu$ -low		prereq.
$G_{22}$	Get salvaged after accident		prereq.
$G_{24}$	Use loading crane		prereq.
$G_{25}$	Use loading crane for heavy weights		prereq.
$G_{26}$	Maintain loading crane		prereq.
$G_{27}$	Use truck	The truck fulfils one of its purposes.	prereq., 1.1
$G_{40}$	Avoid accident		prereq., 1.5, 5.1
$\gamma_{\text{safeAirbag}}$	Protect driver/passenger		prereq., 1.5, 5.1
$\phi_{\text{collide}}$	Collision with other objects		3.2
$\phi_{\text{distract}}$	Distraction of driver		3.2
$\phi_{\text{bump}}$	Bump into interior	Bump of passenger into vehicle interior.	3.2
$\phi_{\text{harmExp}}$	Harmful release	Correct release of the airbag but nevertheless harmful protect.	3.2
$M_{66}$	Engage an accident (unintended)	Unintended maloperation of the vehicle.	3.2
$H_{60}$	Unattended permanent brake		4.5
$\chi'_{\text{UnattMove}}$	Unattended driveway	Unattended driveway or start of movement.	4.5
$H_{62}$	Unattended leaving of lane	Unattended leaving of lane or change of direction.	4.5
$\chi'_{\text{UnexpExp}}$	Unexpected execution of expand	Distraction of driver by unexpected release of airbag.	4.5
$\chi_{\text{supprExp}}$	Unexpected suppression of expand		4.5

Table A.14.: Cutout of the action specifications of the comprising MTSs of the CRV world according to Section 6.2

Label	pre	delay	trig	prio	$\pi$	post
$\alpha(\text{DriveMoves}_r) \in M_{\text{use}} \dots \text{functionality}$						
accelerate	$-49 < \text{position} - \text{speed} < 55 \wedge \text{speed} < 3$	0	T	3	1	position += speed, speed += 1
move	$-49 < \text{position} - \text{speed} < 55$	0	T	3	1	position += speed, speed := speed
retard	$-49 < \text{position} - \text{speed} < 55 \wedge \text{speed} > 0$	0	T	3	1	position += speed, speed -= 1
park	speed = 0	0	T	3	1	speed -= 1
halt	speed = 0	0	T	3	1	speed -= 1
start <sup>+</sup>	speed $\neq$ 0	0	T	3	1	NOP
stop <sup>+</sup>	speed = 0	0	T	3	1	NOP
$\alpha(\text{DriveMoves}_r) \in M_{\text{use}} \dots \text{defective fragment}$						
$\tau^+$	T	0	T	2	1	NOP
$\text{StopBrakes}_r \in M_{\text{use}} \dots \text{functionality}$						
$\epsilon$	speed $\neq$ 0	0	T	3	1	NOP
activate	speed = 0	0	switch <sub>stopbrake</sub>	3	1	NOP
deactivate	$\neg \text{switch}_{\text{stopbrake}}$	0	T	3	1	NOP
brake	T	0	T	3	1	speed := 0
$\text{StopBrakes}_r \in M_{\text{def}} \dots \text{operational defects of StopBrakes}$						
brake <sup>1</sup>	load $\geq$ crit	0	T	1.5	.90	speed -= br(speed, load)
unstableBrake <sup>1</sup>	load $\geq$ crit	0	T	1.5	.05	speed -= unbr(speed, load)
suppressBrake <sup>1</sup>	load $\geq$ crit	0	T	1.5	.05	speed -= subr(speed, load)
$\text{StopBrakes}_s \in M_{\text{save}} \dots \text{preventive safety measure for } (A_s, G_r)_{\text{treat.collide}}$						
detDefBrk	speed $\neq$ 0 $\wedge$ switch <sub>stopbrake</sub>	0	T	1	1	NOP
safeBrake <sup>1</sup>	speed $\neq$ 0	0	T	1	1	speed -= sfr(speed, load)
repair	repaired	1h	repaired	1	1	NOP
$\text{Airbags} \in M_{\text{use}} \dots \text{functionality}$						
activate	energy $\wedge$ crashed	0	crashed	3	1	NOP

Continued on the following page

A. Library, Evidence and Excursions

Label	pre	delay	trig	prio	$\pi$	post
deactivate	$\neg$ energy	0	T	3	1	NOP
activate <sub>0</sub>	energy	0	T	3	1	NOP
$\epsilon_3$	T	0	T	3	1	NOP
<b>Airbags<sub>f</sub> <math>\in \mathbb{M}_{fail}</math> ... operational defects of Airbags</b>						
suppressRelease	energy $\wedge$ crashed	0	crashed	1.5	1	NOP
cancelExpand	gas $\neq$ empty	0	T	1.5	1	gas := empty
$\tau$	T	0	$\perp$	1.5	1	NOP
expand <sub>f</sub>	T	0	$\perp$	1.5	.01	released+ = 1, gas- = 1, repaired := $\perp$
maintain <sub>f</sub>	gas $\neq$ full $\wedge$ repaired	1h	repaired	1.5	.01	gas := defective, repaired := $\perp$
<b>Airbags<sub>s</sub> <math>\in \mathbb{M}_{save}</math> ... driver airbag</b>						
maintain	gas $\neq$ full $\wedge$ repaired	1h	repaired	1.5	.99	gas := full
repair <sub>1</sub>	repaired	0	repaired	1	1	gas := full, released := no
$\epsilon$	$\neg$ active	0	T	1	1	NOP
shutdown	active	0	T	1	1	active := $\perp$ , status <sub>warnLamp</sub> := T, repaired := $\perp$
repair <sub>2</sub>	repaired	1h	repaired	1	1	gas := full, released := no, active := T, status <sub>warnLamp</sub> := $\perp$

<sup>1</sup>The functions br, unbr, subr and sibr model the physical dynamics in the control loop being active in these modes.

## A.7. Application and Evaluation of the Defect Taxonomy

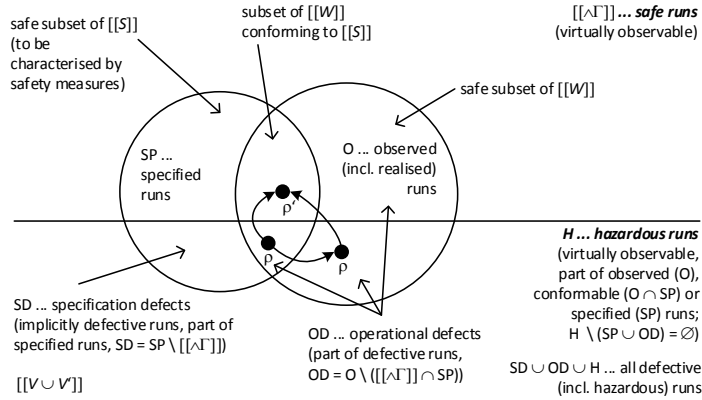


Figure A.1.: Classes of runs investigated by the proposed method;  $\rho, \rho' \dots$  two sample runs

## A.7. Application and Evaluation of the Defect Taxonomy

Table A.15 applies the taxonomy of Section 4.2.1 to align safety-related terminology and the defects investigated in the present work (Section 4.2.2) with the literature.

## A.8. Defects and Model Validity

As discussed in the Sections 1.1 and 7.3, *specification defects* (Definition 4.4) and *wrong abstraction* can make results of V&V inconclusive. Hence, we need to improve *requirements validity* (Chapters 4 and 5) and *model validity* through finding *appropriate abstractions* (cf. challenge 1 on page 109). *Model validation* can be seen as the act of checking that  $\mathcal{M}$  correctly abstracts from  $\mathcal{W}$  and the world in which  $\mathcal{W}$  is going to be realised. This section outlines the influence of model validity on defects.

Following Figure 4.7, Figure A.1 depicts *relationships among classes of runs*. As an alternative to behavioural properties (Definition 2.15), the Tables A.16 to A.20 describe defects as deviating pairs of stimuli and reactions. Nevertheless, property assertions can more concisely and expressively describe complex defects.

**Visibility, Significance and Representation of Deviations** Let  $\mathcal{M}'$  be correct by definition, possibly implicit and approximated by  $\Gamma$ . Three cases, one valid and two threats, can be distinguished:

**op Operational:**  $\mathcal{W}$  is defective, it deviates from  $\mathcal{M}'$  and  $\mathcal{M}$ .

Operational defects (Table A.16a) can be modelled by *defective indeterminacy* (Tables A.16b and A.20c) or fault indicators  $v_f \in \mathcal{V}'$  (Table A.17) to give any  $\mathcal{W}$  the chance to observably deviate. Fault indicators preserve determinism.

**f<sup>-</sup> False negative:**  $\mathcal{W}$  is defective and conforms to an unknowingly defective  $\mathcal{M}$ .

A. Library, Evidence and Excursions

Taxonomy			Terms (References, Defect Category and Kind of Defect)											
			Relevant for this work	L	IEC Std. 61508 (2011), ISO Std. 26262 (2011), Birsöck (2011)	Pock (2012)	DP	specified	plain operational	dependent failures	safety & detectability	perform.	origin	
Criterion	Facet													
1	*	*	s	e	e	s	s	s	s	s	s	s		
	a	s	s	*	*	*	I, SS	r	r	r	r	*		
	b	m	m	*	*	*	*	*	*	*	*	*		
	c	-	-	w	a	a	w	a	w	a	w	a		
	d	p	p	*	*	*	*	*	*	*	*	*		
3	*	*	*	*	1:1	1:1	1:1	1:1	n:1	1:n	1:n	n:n		
	a	-	-	-	-	-	*	n	*	?	?	?		
	b	-	-	-	-	-	-	*	*	*	*	*		
5	s	*	*	*	*	o	o	*	*	*	*	d, r, o		
6	*	*	*	*	*	*	*	*	*	*	*	s, r, o		

Table A.15.: Classification of terms from literature according to Table 4.1. \* ...arbitrary, - ...inapplicable, ? ...inexplicit, (x) ...unusual, L ...Leveson et al. (1997), DP ...Damm and Peikenkamp (2004)



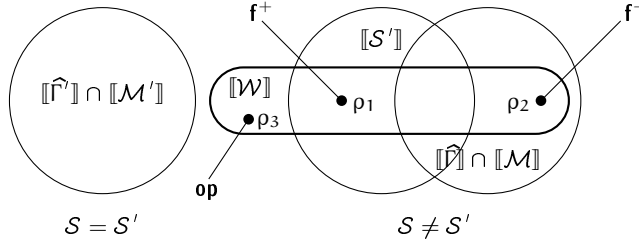


Figure A.2.: A false positive run  $\rho_1$ , a false negative run  $\rho_2$  and an operational defect  $\rho_3$  in case of  $S \neq S'$

Given  $\mathcal{V}_{\text{specified}}$ ,  $\mathcal{V}_{\text{realised}}$  and  $\mathcal{V}_{\text{ideal}} \equiv \mathcal{V}'$  (cf. pages 28 and 48),  $\mathcal{M}$  and  $\mathcal{W}$  may behaviourally conform (Table A.19a) because for the specification based on  $\mathcal{V}_{\text{specified}}$  and any realisation based on  $\mathcal{V}_{\text{realised}}$ , two actually differing runs appear<sup>2</sup> the same (Tables A.18b and A.19b). This lack of observability reduces the detectability of defects and obstructs the definition of safety properties.

$f^+$  *False positive*:  $\mathcal{W}$  conforms to  $\mathcal{M}'$  but deviates from an unknowingly defective  $\mathcal{M}$ .

For a deterministic<sup>3</sup>  $\mathcal{W}$  with  $\mathcal{V}_{\text{realised}} = \mathcal{V}_{\text{ideal}} = \mathcal{V}'$  based on a defective, deterministic  $\mathcal{M}$  with  $\mathcal{V}_{\text{specified}} \subset \mathcal{V}_{\text{realised}}$  (Table A.20a),  $\mathcal{W}$  may appear non-deterministic because of variables disregarded in  $\mathcal{M}$ . This lack of observability can be compensated by *indeterminacy* in  $\mathcal{M}$  (Table A.20b).

As depicted in Figure A.2,  $f^-$  and  $f^+$  denote two threats to validity of the hypothesis “ $\mathcal{W}$  is defective.” By assuming that  $\mathcal{M}'$  specifies correct reactions for  $v_1 = b$ , we can indeterminise  $\mathcal{M}$  to establish conformance with  $\mathcal{M}'$ . Then,  $\mathcal{M}$  captures the choices to consider  $v_1 = b$  in  $\mathcal{W}$  according to  $\mathcal{M}'$ . If  $v_1$  (Table A.20a) is interpreted as a fault indicator (cf. Table A.17)  $\mathcal{M}$  must, however, be retained as in Table A.20b.

**How improper Abstraction affects Defect Detection in V&V** Table A.21 outlines the use of model checking to generate runs for system tests (Gleirscher 2011). The cases 2, 5 and 6 exhibit the threat  $f^-$ , that is, we may fail to detect operational defects. In the cases 5 and 6, we may even be unable to generate runs. Testing requires the check for model validity and whether the execution and monitoring of a test case was done in a correct way.

The threat  $f^+$  may appear rarely during testing if the runs generated from  $\mathcal{M}$  fulfil  $\phi$  and  $\phi$ 's positive or negative *intent*— $\Gamma'$  under-approximates  $\mathcal{M}'$ —is known (cf. Table A.21). Unexpected stimuli treated by safety measures according to Section 4.6 can be ignored to reduce  $f^+$ .

<sup>2</sup>Because of wrongly typed, missing or superfluous variables.

<sup>3</sup>We might assume that  $\mathcal{V}_{\text{ideal}}$  completely captures the world state space including fault indicators. In this respect, we have *quasi-determinism* or “determinism modulo abstraction.”

A. Library, Evidence and Excursions

	stimulus		reaction	
	v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>
$\mathcal{M}'$	a	a	b	a
	b	a	c	d
$\mathcal{M}$	a	a	b	a
	b	a	c	d
$\mathcal{W}$	a	a	b	a
	b	a	f	d

	stimulus		reaction	
	v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>
$\mathcal{M}'$	a	a	b	a
	b	a	c	d
$\mathcal{M}$	a	a	b	a
	b	a	c	d
$\mathcal{W}$	a	a	b	a
	b	a	c	d

(a) (b)

Table A.16.: Samples for (a) case **op**,  $\mathcal{W}$  exhibits a failure (deterministic as opposed to Table A.20c), (b) analysing case **op**,  $\mathcal{M}$  is non-deterministic,  $\mathcal{V}_{\text{specified}} = \mathcal{V}' = \{v_1, v_2, v_3, v_4\}$ ; samples (one per row) hide their actions

	stimulus		fault	reaction	
	v <sub>1</sub>	v <sub>2</sub>	v <sub>f</sub>	v <sub>3</sub>	v <sub>4</sub>
$\mathcal{M}'$	a	a	-	b	a
	b	a	-	c	d
$\mathcal{M}$	a	a	-	b	a
	b	a	-	c	d
$\mathcal{W}$	a	a	?	b	a
	b	a	n	c	d

Table A.17.: Sample for case **op**: a defective mode of  $\mathcal{W}$  activated or deactivated by an additional fault indicator  $v_f \in \mathcal{V}'$ ; - ... disregarded, ? ...  $\mathcal{W}$  needs an oracle

	stimulus		reaction	
	v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>
$\mathcal{M}'$	a	a	b	a
	b	a	c	d
$\mathcal{M}$	a	a	b	a
	b	a	c	d
$\mathcal{W}$	a	a	b	a
	b	a	c	d

	stimulus		reaction	
	v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>
$\mathcal{M}'$	a	a	b	a
	b	a	c	d
$\mathcal{M}$	-	a	b	a
	-	a	b	a

	stimulus		reaction	
	v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>
$\mathcal{M}'$	a	a	b	a
	b	a	c	d
$\mathcal{M}$	-	a	b	a
	-	a	c	d
$\mathcal{W}$	?	a	b	a
	?	a	c	d

(a) (b) (c)

Table A.18.: Samples for (a) model validity, (b) threat  $f^-$  with deterministic  $\mathcal{M}$  and  $\mathcal{W}$  and (c) handling threat  $f^-$  by indeterminacy in  $\mathcal{M}$  to achieve model validity;  $\mathcal{V}_{\text{specified}} = \{v_2, v_3, v_4\}$ ,  $\mathcal{V}' = \{v_1, v_2, v_3, v_4\}$

A.8. Defects and Model Validity

	stimulus		reaction			stimulus		reaction	
	v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>		v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>
$\mathcal{M}'$	a	a	b	a	$\mathcal{M}'$	a	a	b	a
	b	z	d	d		b	a	c	d
$\mathcal{M}$	a	a	b	a	$\mathcal{M}$	-	a	b	a
	b	z	y *	d		-	a	f	a
$\mathcal{W}$	a	a	b	a	$\mathcal{W}$	a	a	b	a
	b	z	y *	d		b	a	f	d

(a)

(b)

Table A.19.: Samples for threat  $f^-$ : (a) violable domain maintenance assumption and violated range maintenance guarantee. z is out of  $\mathcal{M}'$ 's domain, d would be a valid reaction but  $\mathcal{M}$  implies y|\* (y or anything) and  $\mathcal{W}$  may react with y or anything; (b)  $\mathcal{M}$  is defective and non-deterministic,  $\mathcal{W}$  is defective and deterministic

	stimulus		reaction			stimulus		reaction			stimulus		reaction	
	v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>		v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>		v <sub>1</sub>	v <sub>2</sub>	v <sub>3</sub>	v <sub>4</sub>
$\mathcal{M}'$	a	a	b	a	$\mathcal{M}'$	a	a	b	a	$\mathcal{M}'$	a	a	b	a
	b	a	c	d		b	a	c	d		b	a	c	d
$\mathcal{M}$	-	a	b	a	$\mathcal{M}$	-	a	b	a	$\mathcal{M}$	-	a	b	a
	-	a	c	d		-	a	c	d		-	a	c	d
$\mathcal{W}$	a	a	b	a	$\mathcal{W}$	a	a	b	a	$\mathcal{W}$	a	a	b	a
	b	a	c	d		b	a	c	d		b	a	c	d

(a)

(b)

(c)

Table A.20.: Samples for (a) threat  $f^+$  (as opposed to Table A.18b), (b) handling threat  $f^+$  by indeterminacy in  $\mathcal{M}$  and (c)  $\mathcal{W}$  conforming to indeterminacy in  $\mathcal{M}$  (row 5–6) but adding failure as indeterminacy not in  $\mathcal{M}$  (row 6–7, as opposed to Table A.16b);  $\mathcal{V}_{\text{specified}} = \{v_2, v_3, v_4\}$ ,  $\mathcal{V}' = \{v_1, v_2, v_3, v_4\}$

Case	Intent of $\phi$	Satisfying runs ( $\mathcal{M} _{\text{use}}$ )	$\mathcal{M} _{\text{use}}$	Run conformance	$\mathcal{W}$	Result	Threat
1	neg	> 0	d	yes	d	realised spec. defect	
2	neg	> 0	d	no	-	spec. defect	$f^-$
3	pos	> 0	-	yes	-	no defect	
4	pos	> 0	-	no	d	operational defect	
5	neg	none	-	unknown	-	no defect	$f^-$
6	pos	none	d	unknown	-	spec. defect	$f^-$

Table A.21.: Model checking and testing based on  $\mathcal{M}|_{\text{use}}$ ; -...no defect found, d...defect found

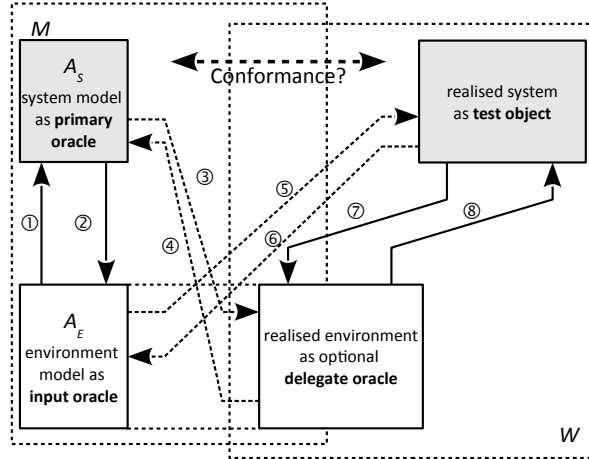


Figure A.3.: The system in  $\mathcal{W}$  as a test object, oracles (primary, input and optional delegate, e.g.  $\mathcal{A}_S$ ,  $\mathcal{A}_E$  and the environment part of  $\mathcal{W}$ ) and channels 1–8 to observe

**A Test Bed for Model Validation** *Which functions and components are required to be implemented?* A test bed should include test drivers and stubs:  $\mathcal{A}_E$  simulating the environment based on a model of the physical world, a monitoring and control component as an interface between  $\mathcal{A}_E$  and the system and an interface for the test engineer to control the test bed. Based on the system boundary, *oracle-based testing* (e.g. Liggesmeyer 2009) can use  $\mathcal{M}$  as *primary and input oracles* and the realised environment as a *delegate oracle*. Figure A.3 depicts channels observable by the test case generator (1–2 for input/output generation, 3–4 for optional capture of delegated interactions with the environment), the test drivers and stubs (5 for stimuli, 6 for response monitoring, 7–8 for usage and diagnosis of delegate oracles at testing-time) and diagnosis at runtime (7–8). Runs (Definition 2.2) and complex actions (Definition 2.10) allow the representation of test cases to be observed at 1–2 and 7–8.

*How can the test bed be implemented?*  $\mathcal{M}$  can be axiomatised using an action language (see Section 2.2.1) or tools for temporal reasoning, for example, PRISM<sup>4</sup>, SPIN<sup>5</sup> or  $\nu$ SMV<sup>6</sup>. Gleirscher (2011) sketches how  $\mathcal{M}$  can be transformed into GOLOG, a PROLOG-based language for action theories described by Reiter (2001).

<sup>4</sup>[www.prismmodelchecker.org](http://www.prismmodelchecker.org)

<sup>5</sup>[spinroot.com](http://spinroot.com)

<sup>6</sup>[nusmv.fbk.eu](http://nusmv.fbk.eu)



## Indices

### B.1. Glossary of Symbols and Notation

#### Mathematical Symbols and Notation

---

$\mathbb{N}_0, \mathbb{R}_0^+$	natural numbers incl. 0, positive real numbers incl. 0
$\mathcal{P}(S), 2^S$	power set of a set $S$
$v : T$	variable $v$ is constrained to type $T$
$(a, b)$	tuple consisting of the elements $a$ and $b$
$[a, b, c]$	list consisting of the elements $a, b$ and $c$
$t.e$	selects the element $e$ from a tuple $t = (\dots, e, \dots)$
$\llbracket . \rrbracket$	the behavioural spectrum of a set of variables, a transition system or a formula
$S _P$	If $S$ is a set of states, runs or tuples and $P$ a set of parameters: $S' := S _P$ assigns to $S'$ the projections of each $s \in S$ such that $s' \in S'$ is reduced to valuations of $p \in P$ . If $S$ is a formula, free variables in $S$ but not in $P$ are hidden by $\exists$ -quantification. If $S$ is an MTS and $P$ an aspect label, $S _P$ denotes the restriction of $S$ to MTSs in $\mathbb{M}_P$ .
$\rightsquigarrow, \rightsquigarrow_*, \rightsquigarrow_r$	single transformation step, multiple steps, step applying rule $r$

---

## Logical Symbols and Notation

$\top, \perp$	logical truth constants for the valuations “true” and “false”
$\Leftrightarrow$	logical equivalence
$\models$	“entails” (logical consequence)
$\vdash$	“writable as” (syntactic inference)
$\triangleq$	“is defined as” for global definitions
$\equiv$	“is congruent with” or “can be substituted by” (term equivalence)
$::=$	“consists of” defining language grammar
$=, ==$	“equality” as a predicate to be fulfilled (semantic equivalence)
$:=$	variable binding or value assignment
$(\forall x, y \exists z) \phi$	quantifications can be written in parentheses
$\forall x : \phi, \exists x. \phi$	reduces outmost parentheses of $\forall x(\phi)$ or $(\exists x)(\phi)$
$\widehat{\Phi}, \bigwedge \Phi$	conjunction of a set $\Phi$ of formulae by $\bigwedge_{\phi \in \Phi} \phi$
$\check{\Phi}, \bigvee \Phi$	disjunction of a set $\Phi$ of formulae by $\bigvee_{\phi \in \Phi} \phi$

## B.2. List of Figures

1.1	A setting for safety-oriented requirements validation . . . . .	6
1.2	An iterative procedure for safety-oriented requirements validation . . . . .	7
2.1	Analytic quality assurance of engineering artefacts . . . . .	12
2.2	Exemplary visualisation of hierarchical decomposition . . . . .	25
3.1	Approaches to hazard analysis of technical systems . . . . .	36
3.2	Kinds of safety measures . . . . .	37
4.1	World model . . . . .	47
4.2	Actual and safe description and representation of a specification . . . . .	48
4.3	A hazard $\chi$ in the behavioural spectrum of $\mathcal{M}$ . . . . .	54
4.4	A behaviour taxonomy for hazard analysis . . . . .	54
4.5	An abstract transition system for hazard analysis . . . . .	58
4.6	A pattern for goal-based hazard treatment . . . . .	64
4.7	Consistency of artefacts aimed by validation . . . . .	65
5.1	Overview of the procedure . . . . .	69

5.2	Overview of stages, steps and sub-steps of the procedure . . . . .	74
6.1	ATM tactics and functions . . . . .	80
6.2	ATM LID <sub>S</sub> defective fragment . . . . .	81
6.3	Hazardous complex actions and a run of the ATM world . . . . .	87
6.4	Hazardous complex actions of the ATM world . . . . .	88
6.5	ATM assertional graph . . . . .	88
6.6	ATM safety fragments . . . . .	89
6.7	CRV $\alpha$ (DriveMove <sub>S</sub> ) fragments . . . . .	93
6.8	CRV StopBrake <sub>S</sub> fragments . . . . .	93
6.9	CRV Airbag <sub>S</sub> fragments . . . . .	94
6.10	CRV StopBrake <sub>S</sub> safety measure . . . . .	101
6.11	CRV Airbag <sub>S</sub> safety measure . . . . .	102
7.1	Overview of concepts for behavioural safety . . . . .	112
7.2	Specification-based testing . . . . .	115
A.1	Classes of runs investigated by the proposed method . . . . .	157
A.2	False positives and false negatives . . . . .	159
A.3	Overview of test bed . . . . .	162

### B.3. List of Tables

1.1	Data on incidents and accidents of technical systems . . . . .	2
2.1	Basic classes of actions . . . . .	21
4.1	Criteria for a defect taxonomy . . . . .	49
4.2	Tabular representation of defective modes . . . . .	52
4.3	Property defects for a hazard . . . . .	63
6.1	ATM use case “withdraw cash” . . . . .	77
6.2	ATM use case “depose cash” . . . . .	78
6.3	Variables for safety analysis of the ATM . . . . .	79
6.4	ATM hazard assessment results . . . . .	84
6.5	CRV use case “use truck” . . . . .	90
6.6	CRV use case “park at steep hill” . . . . .	90
6.7	CRV use case “use brakes” . . . . .	91
6.8	Variables for safety analysis of the CRV . . . . .	92

6.9	CRV hazard assessment results	98
6.10	A run of $\mathcal{M}$ which models the safety goal $\gamma_{safeAirbag}$	103
A.1	Patterns for causal factor search	133
A.2	Patterns for operational defects	135
A.4	Patterns to treat transition system defects	137
A.3	Patterns for state constraints	142
A.5	Catalogue of interview questions	143
A.6	Overview of interviews and safety practitioners	144
A.7	Confirmations and refutations from the interviews	145
A.8	Systematic map of hazard analysis approaches: part 1	146
A.9	Systematic map of hazard analysis approaches: part 2	147
A.10	A comparison of three safety engineering methods	148
A.11	ATM informal property assertions (cutout)	149
A.12	ATM functions and tactics	150
A.13	CRV informal property assertions (cutout)	153
A.14	CRV functions and tactics	155
A.15	Application of defect taxonomy to terms from literature	158
A.16	Samples for case <b>op</b>	160
A.17	Deterministic sample for case <b>op</b>	160
A.18	Samples for threat $f^-$	160
A.19	Samples for threat $f^-$	161
A.20	Samples for threat $f^+$	161
A.21	Model checking and testing	161

## B.4. List of Examples

Example 1.1:	Car Airbag	2
Example 1.2:	Car Airbag	7
Example 2.1:	MTS of a Car Airbag	20
Example 2.2:	MTS of a Car Airbag	23
Example 3.1:	Fail-Safe Patterns	38
Example 4.1:	Operational Defects of a Car Airbag	52
Example 4.2:	Mishap and Hazard Identification for a Car Airbag	55
Example 4.3:	Safety Goals for a Car Airbag and a Car	57
Example 4.4:	Negotiation of Responsibilities for a Car Airbag	60
Example 4.5:	Safety Measures for a Car Airbag	64
Example 4.6:	Maloperation of Electric Light	66



- A**  
A/G pair, *see* assumption/guarantee pair  
abstraction, 3, 14, 105, 109, 112  
    action -, 24  
    state -, 24  
action, 18  
    - effect, 18, 23, 113  
    - trace, 24  
    complex -, 24  
    composite -, 23  
agent, 46  
ALARP, 37, 53, 55, 57  
application domain, 12  
ASIL, 36  
aspect, 25  
assumption, 13, 59  
assumption/guarantee pair, 13, 59  
ATM, 76  
automation paradox, 4, 60, 113
- B**  
behaviour, 14, 17  
behavioural  
    - model, 14, 17  
    qualitative -, 15, 43  
    - spectrum, 18
- C**  
causal  
    - chain, 14, 32  
    - factor, 2, 4, 33, 34, 50, 53, 60, 71  
channel, 14, 18, 47  
computation tree, 17  
control  
    - loop, 6, 108  
    - software, 6, 34, 40, 46  
    - subsystem, 1, 37  
CRV, 90
- D**  
defect, 2, 3, 12, 16, 49, 109  
    - model, 6, 16, 51  
    operational -, 6, 7, 49, 57  
    specification -, 3, 49, 57, 157  
deviation, 48, 157  
driving situation, *see* operational situation
- E**  
environment, 12, 13, 46  
    - model, 6, 46  
ETA, 35  
event, 18
- F**  
failure, 16, 52  
    - mode, *see* defective mode  
fault, 16, 52  
    - indicator, 17, 51, 66, 70  
    - model, *see* defect model  
    - tolerance, 16, 38  
FMEA, 35  
FMECA, 35  
fragment, 23  
FTA, 34  
function, 14, 25, 34, 47  
functionality, *see* function
- G**  
goal, 12, 56  
guarantee, 13, 59

## Index

guide word, 35, 54

### H

HARA, 34, 140  
hazard, 6, 33, 48, 53  
  - analysis, 3, 34, 39, 48  
HAZOP, 35  
hierarchical decomposition, 14, 16, 25, 104

### I

IC, 36, 73, 84, 98  
incident, 33, 40  
indeterminacy, 14, 21, 26, 70, 110, 157  
INES, 2  
interface, 14  
iteration, 70-72

### M

minimal cut sequence, 34, 115  
mishap, 6, 33, 53  
mode, 3, 19, 47  
  - channel, 18, 26, 113  
  - dependency, 26, 51, 111, 115  
  defective -, 47, 52  
  hazardous -, 58  
  safe -, 47  
model checking, 13, 28  
modularity, 14  
MTS, *see* mode transition system

### O

operational situation, 45, 48, 56, 90

### P

parallel composition, 23  
pattern, 36, 38, 61  
  fail-operational -, 38, 61  
  fail-safe -, 38, 61  
  fail-silent -, 38, 61  
PCTL\*, 27  
projection, 18, 163  
property, 1, 13  
  behavioural -, 7, 14, 24, 27, 33, 66, 105  
  domain -, 26, 48, 69, 77  
  interface -, 27, 29  
  structural -, 14

### R

RCA, 35  
realisation, 3, 28  
refinement, 26, 67, 96  
reliability, 3, 36, 40, 45, 70, 113  
requirement, 13  
requirements engineering, 13  
responsibility, 2, 16, 59  
  misperception of -, 53, 62

run, 17

### S

safety, 1, 37  
  - engineering, 31  
  - goal, 48, 57  
    A/G-based -, 59, 62  
  - integrity, 37, 56  
  - measure, 2, 4, 37, 66  
  - requirement, 8, 36, 59  
  behavioural -, 6, 57-59  
  functional -, 3, 37, 56, 66, 112  
satisfaction, 28  
SIL, 36  
specification, 12, 13, 16, 28  
  assumed -, 4, 45  
state, 14, 17  
  - constraint, 18  
  abstract -, 58  
  defective -, 38, 52, 58  
  harm -, 48, 53, 58  
  hazardous -, 38, 53, 58  
  operational -, 48, 58  
  safe -, 38, 58  
superimposition, 23  
system, 1, 11, 12, 37  
  - boundary, 13, 14, 46, 62, 162  
  - life cycle, 12  
  - model, 6, 13  
  - property, 13  
  - view, 16, 28, 44, 48  
systems engineering, 12

### T

tactic, 47  
temporal logic, 16, 27, 110  
totality, 21  
transformation, 14  
transition system, 15  
  abstract -, 24  
  mode -, 6, 18, 24  
  concurrent -, 23, 30  
treatment, 60

### U

underspecification, 16, 21, 26, 59, 73, 110  
use case, 13, 35, 69, 70, 91

### V

V&V, 1, 12, 13  
validation, 38, 69

### W

world, *see* realisation  
  - model, 6, 46



Wovon man nicht sprechen  
kann, darüber muss man  
schweigen.

---

*(Ludwig Wittgenstein 1922)*