Copyright Notice

The document is provided by the contributing author(s) as a means to ensure timely dissemination of scholarly and technical work on a non-commercial basis. This is the author's version of the work. The final version can be found on the publisher's webpage.

This document is made available only for personal use and must abide to copyrights of the publisher. Permission to make digital or hard copies of part or all of these works for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage. This works may not be reposted without the explicit permission of the copyright holder.

Permission to reprint/republish this material for advertising or promotional purposes or for creating new collective works for resale or redistribution to servers or lists, or to reuse any copyrighted component of this work in other works must be obtained from the corresponding copyright holders. It is understood that all persons copying this information will adhere to the terms and constraints invoked by each copyright holder.

Inderscience papers: © Inderscience Enterprises Ltd. This is the authors' version of the paper provided only for non-commercial purposes. The final publication is available at http://www.inderscience.com/

A survey of safety-oriented model-driven and formal development approaches

Yuliya Prokhorova* and Elena Troubitsyna

TUCS – Turku Centre for Computer Science,
Åbo Akademi University,
Department of Information Technologies,
Joukahaisenkatu 3-5 A, 20520 Turku, Finland
E-mail: Yuliya.Prokhorova@abo.fi
E-mail: Elena.Troubitsyna@abo.fi
*Corresponding author

Abstract: Numerous model-driven and formal approaches have been proposed to facilitate development of complex critical systems. To address safety concerns, these approaches incorporate safety analysis techniques at different stages of the system development process. In this paper, we overview the approaches that have been recently proposed to integrate safety analysis into model-driven and formal development of critical systems. Moreover, we identify several criteria for classifying and comparing these approaches. Our survey aims at guiding industry practitioners as well as identifying promising research directions in the area.

Keywords: survey; safety analysis techniques; model-driven development; MDD; formal methods; critical systems.

Reference to this paper should be made as follows: Prokhorova, Y. and Troubitsyna, E. (xxxx) 'A survey of safety-oriented model-driven and formal development approaches', *Int. J. Critical Computer-Based Systems*, Vol. x, No. x, pp.xxx–xxx.

Biographical notes: Yuliya Prokhorova is a PhD student at Åbo Akademi University and TUCS – Turku Centre for Computer Science. She received her MSc in Computer Systems and Networks from National Aerospace University 'KhAI', Kharkiv, Ukraine in 2008. Her scientific interests include formal modelling and verification of safety-critical and fault-tolerant systems as well as development of safety cases.

Elena Troubitsyna is an Associate Professor at the Department of Information Technologies of Åbo Akademi University. She received her PhD in Computer Science in 2000 on design methods for dependable systems. Her research interests include application of formal methods to development of dependable fault-tolerant systems. She also conducts research on combining formal methods with informal techniques of safety analysis and semi-formal design techniques such as UML. She has worked on applying formal methods to development of industrial fault-tolerant systems within the EU IST projects MATISSE, RODIN, and DEPLOY.

Copyright © 200x Inderscience Enterprises Ltd.

1 Introduction

Development of safety-critical systems is a challenging engineering task. Knight (2002) defines *safety-critical systems* as *'systems whose failure could result in loss of life, significant property damage, or damage to the environment'*. The examples of such systems are found in different application domains – transportation, avionics, space, medicine, etc. Significant research work is dedicated to development, verification and validation of these systems. In particular, over the recent years, model-based development approaches have received significant research attention. In our survey, we consider two mainstream approaches in model-based engineering: *model-driven development (MDD)* and *formal development*.

The *MDD* paradigm (Mellor et al., 2003) received a notable uptake over the last decade. Selic (2003) points out that models help the developers understand a complex problem and its potential solutions through abstraction. The developed automated technologies allow the practitioners to generate complete programmes from the given models and then automatically verify the models by executing them. Usually the models have a graphical representation. To facilitate their verification, a number of approaches formalising the graphical models have been proposed. Such approaches can be considered as semi-formal. The overall experience in utilising *model-driven engineering* (*MDE*) in industrial practice was recently surveyed by Hutchinson et al. (2011).

The *formal approaches* rely on the formal specification languages, i.e., the languages with rigorously defined semantics, to model the behaviour and properties of the system. Though formal approaches to the development of safety-critical systems are not as widely adopted in industry as MDD approaches, their use is mandatory for highly critical applications. The application of different formal methods to the development of safety-critical systems was considered by Barroca and McDermid (1992). The advantages and limitations of formal approaches to software development are discussed in Gaudel (1991) and experience in the practical use of formal methods is summarised in Woodcock et al. (2009). In this paper, we surveyed the approaches that explicitly integrate safety analysis into the modelling and verification process. The aim of this integration is to ensure an adequate representation of safety requirements in the system model as well as validate the modelling results from the system safety point of view.

There is a number of comparative studies and surveys related to the topic of our research. However, our survey differs from the existing ones because we cover not only the visual MDD approaches based on semi-formal languages such as UML/SysML (Giese and Henkler, 2006; Bernardi et al., 2012) but also the approaches based on formal languages, e.g., ITL, B, Event-B.

Since the considered research area develops quickly, we need to reassess the state-of-the-art in this field. Our survey aims at analysing the most prominent approaches proposed during the last decade and identifying open issues in the area. To search for the relevant papers on the topic of our studies, we have used scientific paper databases such as IEEE *Xplore*, SpringerLink, ACM Digital Library, and Science Direct. Additionally, we have conducted a manual search of conference and workshop proceedings. We have reviewed over 200 papers and, as a result, chosen 14 model-driven and seven formal safety-oriented approaches for the detailed consideration.

In this paper, we classify the approaches according to seven criteria such as the application domain, the modelling languages used, the covered phases of the

development life cycle, the adopted safety analysis techniques, the nature of safety analysis (qualitative or quantitative), and the availability of automated tool support. To facilitate selection of a suitable technique, we also define what are the required inputs and outputs for each approach.

We believe that the survey could help industrial practitioners to identify the most suitable approaches to fulfil their design objectives. Moreover, it also could help researchers to spot interesting research directions that have not yet received a proper treatment.

The paper is organised as follows. In Section 2, we define the evaluation criteria, which are later used to classify the surveyed approaches, and motivate their choice. In Section 3, we present the MDD approaches to be considered in the survey and give the results of their evaluation according to the defined criteria. Section 4 presents the formal approaches and their evaluation according to the defined criteria. In Section 5, we discuss the approaches that are relevant to the development of dependable systems in general, rather than specifically focused on safety. Finally, Section 6 highlights the open issues and gives some concluding remarks.

2 Evaluation criteria

To evaluate the selected model-driven and formal development approaches, we introduce seven evaluation criteria. The set of criteria has been derived from the published surveys, e.g., Bernardi et al. (2012), and complemented with our own criteria identified during the survey-related literature review.

- *Domain-specific criterion* – classifies the considered approaches according to their application domain. In general, *domain-pecific modelling* (*DSM*) aims at helping the developers to solve a particular set of problems. It allows for reusing the previously derived concepts that are common for the domain. One of the goals of DSM is increasing the productivity by facilitating verification and validation. Moreover, DSM is often supported by the automatic code generation. Therefore, the domain-specific criterion is an important aspect that influences the choice of the approach to be applied.
- *Language-specific criterion* – identifies the modelling languages used in an approach. The model-based development is supported by a number of either semi-formal, e.g., AADL, UML, or formal, e.g., ITL, B, Event-B, modelling languages. Naturally, the developers would prefer the modelling languages they use in their everyday practice. Hence, in our survey we classify the approaches according to the language-specific criterion as well.
- *Life cycle stages criterion* – relates each approach to the stages of the system and software development life cycle at which it can be applied. To standardise the system development, two models – *systems development life cycle* (*SDLC*) and its subset *software development life cycle* (*SWDLC*) – were proposed (SDLC, 2012). Their primary objectives are to ensure that high quality systems are delivered, to provide management control over the projects, and to increase productivity of the developers.

In our survey, we consider the following development stages: *requirements*, *architecture*, *design*, *implementation*, *verification*, and *validation*.

- *Safety analysis techniques criterion* – classifies the considered approaches with respect to the safety analysis techniques they adopt. There is no best 'universal' safety analysis technique because different tasks require different approaches. To make the choice easier, we present the analysis of all the surveyed approaches according to this criterion.
- *Quantitative/qualitative aspects of analysis criterion* – shows whether an approach permits the quantitative evaluation of safety. Often system engineers need to evaluate the overall risk levels (i.e., risk severity and risk probability) as well as to determine the areas with high probabilities of failure. For these purposes, an approach that incorporates some quantitative analysis technique must be chosen.
- *Availability of automation/tool support criterion* – highlights whether an approach is tool supported. The availability of automation and tool support allows us to reduce the development time and improve confidence in the system design.
- *Inputs and outputs criterion* – describes the inputs and outputs of each approach. By introducing this criterion, we aim at summarising the considered approach as a model or data transformation facility.

3 Evaluation of the model-driven approaches

3.1 MDD approaches

Despite the significant progress observed in MDE, Straeten et al. (2009) identify several groups of questions that still require further attention. They concern such important topics as model quality, run-time models, requirements modelling, standards and benchmarks, modelling languages, domain-specific modelling, empirical analysis, model verification and validation, process support, industrial adoption, formal foundations, scalability issues, and finally, model consistency and co-evolution.

Another classification of MDE challenges is given by France and Rumpe (2007). They distinguish the following groups of challenges: modelling language challenges, separation of concerns challenges, and model manipulation and management challenges. Cabot and Yu (2008) also point out that MDE methods focus on the functional requirements of the system, however, consider neither the organisational context of the system nor its non-functional requirements (usability, reliability, safety, etc.).

In our survey, we review the safety-oriented MDD approaches that aim at bridging the gap between the safety analysis techniques used at different system life cycle stages and the MDD techniques. The list of the selected MDD approaches with the relevant citations and abbreviations is given in Table 1.

Table 1 Considered model-driven approaches

Approach	*Citation*	*Abbreviation*
Integrating FTA into AADL models	Sun et al. (2007)	FTA-AADL
Automatic generation of fault trees from AADL models	Joshi et al. (2007)	AADL-AutoFT
Constructing fault trees from AADL models	Li et al. (2011)	AADL-FT
Automated analysis of extended	Bozzano et al. (2009)	COMPASS
AADL models	Bozzano et al. (2010a,b)	
Integrating the 3+1 SysML with safety engineering	Thramboulidis (2010) Thramboulidis and Buda (2010) Thramboulidis and Scholz (2010)	3+1 SysML-SE
Integrating reliability analysis in the design processes using model-based system engineering techniques	David et al. (2008, 2009) David et al. (2010) Cressent et al. (2011)	MeDISIS
Model-based safety analysis	Belmonte and Soubiran (2012)	MBSE
UML-based FMECA	Guiochet and Baron (2003) Guiochet et al. (2004)	UML-FMECA
Integration of safety analysis in model-driven software development	DeMiguel et al. (2008)	ISAMDSD
UML-based severity analysis methodology	Hassan et al. (2005)	UML-SAM
Software FMEA for UML-based software	Hecht, Xuegao and Hecht (2004)	UML-SFMEA
Model-driven automated software FMEA	Snooke (2004) Snooke and Price (2011)	AutoSFMEA
UML-based method for risk analysis	Martin-Guillerez et al. (2010) Guiochet et al. (2010)	HAZOP-UML
From AADL models to stochastic petri nets	Rugina et al. (2007) Rugina et al. (2008)	AADL-GSPN

Next we present a comparative analysis of the chosen MDD approaches with respect to the defined criteria.

3.2 Results of evaluation

3.2.1 Domain-specific criterion

The majority of the surveyed MDD approaches are used for the development of safety-critical and embedded systems in general (Table 2). For example, AADL-AutoFT, AADL-FT, and AutoSFMEA are among such generic approaches. However, some of the approaches are more specialised. For instance, FTA-AADL is proposed for the product-line engineering. Such approaches as ISAMDSD, UML-SAM and AutoSFMEA are software-oriented. UML-FMECA and HAZOP-UML have been used for the development of automated systems, in particular, human-robotic systems. The specialisation of MeDISIS and UML-SFMEA is real-time embedded systems. The rest of the considered approaches are domain-specific. COMPASS is a space-sector-oriented

approach, while 3+1 SysML-SE has been applied for the industrial engineering of mechatronic systems. Finally, the approach MBSE is dedicated to the railway domain.

Table 2 Overview of approaches and criteria – domain-specific

Approaches	*Application domain*	*Specialisation*
FTA-AADL	General	Safety-critical product-lines
AADL-AutoFT	General	Safety-critical systems
AADL-FT	General	Safety-critical systems
COMPASS	Aerospace	Safety-critical systems
3+1 SysML-SE	Industrial engineering	Mechatronic systems
MeDISIS	General	Real-time embedded systems
MBSE	Railways	Signaling systems
UML-FMECA	Medical engineering	Human-robotic systems
ISAMDSD	General	Software-related
UML-SAM	General	Safety-critical software
UML-SFMEA	General	Embedded software for real time systems
AutoSFMEA	General	Embedded software
HAZOP-UML	General	Human-robotic systems
AADL-GSPN	General	Real-time safety-critical systems

3.2.2 Language-specific criterion

Currently, two kinds of modelling languages are used in the considered approaches: *architecture analysis and design language* (*AADL*) (AADL, 2012; Feiler and Gluch, 2012) and *unified modelling language* (*UML*) (OMG UML, 2012). Each of these two languages has its own extensions. In this survey, we consider several approaches that are based on such extensions as *system-level integrated modelling* (*SLIM*) language (Bozzano et al., 2010b) for AADL and *system modelling language* (*SysML*) (OMG SysML, 2012) for UML. SLIM is a subset of AADL for modelling the heterogeneous systems that include software and hardware components and their interactions. A SLIM specification includes a description of the nominal (functional) behaviour, a description of the erroneous (dysfunctional) behaviour, and fault injections, i.e., a description how the erroneous behaviour influences the nominal behaviour. Similarly, SysML extends UML to facilitate system engineering needs. It provides extensions of UML to incorporate requirements in the system engineering process as well as to support the specification, analysis, design, verification and validation of complex systems. Hence, all approaches can be divided into two categories: AADL-based and UML/SysML-based. As shown in Table 3, both modelling languages are widely used. Among the considered approaches, six are AADL-based and nine are UML/SysML-based. The approach called MeDISIS belongs to both categories. Moreover, David et al. (2010) extend MeDISIS with the method that allows construction of an *AltaRica data flow* (*AltaRica DF*) model (Point and Rauzy, 1999) from the results obtained during functional and risk analyses. The MBSE approach also supports transformation of SysML models into AltaRica models. The approach AADL-GSPN permits the model transformation from an AADL model into a *generalised stochastic petri net* (*GSPN*) model (Chiola et al., 1993). Due to iterative character of model

transformations, a GSPN model as well as the corresponding AADL architecture and its error model can be validated progressively and corrected, if required.

Table 3 Overview of approaches and criteria – language-specific

Approaches	*Languages*					
	AADL	*SLIM*	*UML*	*SysML*	*AltaRica*	*GSPN*
FTA-AADL	X	-	-	-	-	-
AADL-AutoFT	X	-	-	-	-	-
AADL-FT	X	-	-	-	-	-
COMPASS	X	X	-	-	-	-
3+1 SysML-SE	-	-	-	X	-	-
MeDISIS	X	-	-	X	X	-
MBSE	-	-	-	X	X	-
UML-FMECA	-	-	X	-	-	-
ISAMDSD	-	-	X	-	-	-
UML-SAM	-	-	X	-	-	-
UML-SFMEA	-	-	X	-	-	-
AutoSFMEA	-	-	X	-	-	-
HAZOP-UML	-	-	X	-	-	-
AADL-GSPN	X	-	-	-	-	X

Table 4 Overview of approaches and criteria – life cycle stages

Approaches	*Life cycle stages*				
	Requirements	*Architecture*	*Design*	*Implementation*	*V&V*
FTA-AADL	-	X	-	-	-
AADL-AutoFT	-	X	-	-	-
AADL-FT	-	X	-	-	-
COMPASS	X	X	X	-	X
3+1 SysML-SE	X	-	-	-	-
MeDISIS	-	-	X	-	-
MBSE	X	-	X	-	-
UML-FMECA	X	-	-	-	-
ISAMDSD	-	X	X	-	-
UML-SAM	-	X	-	-	-
UML-SFMEA	-	-	X	X	X
AutoSFMEA	-	-	X	-	-
HAZOP-UML	X	-	-	-	-
AADL-GSPN	-	X	-	-	-

3.2.3 Life cycle stages criterion

The analysis results presented in Table 4 allow us to conclude that the majority of the proposed approaches focus on such development stages as *requirements* – five approaches, *architecture* – seven approaches and *design* – six approaches, while only one of them covers the *implementation* phase (UML-SFMEA), and two – the *verification* and *validation* (*V&V*) phases (COMPASS and UML-SFMEA).

3.2.4 Safety analysis techniques criterion

The overview of the approaches based on the safety analysis techniques criterion is given in Table 5.

Table 5 Overview of approaches and criteria – safety analysis techniques

Approaches	*Safety analysis techniques*							
	Static FT	*FTA*	*FMEA*	*FMECA*	*SFMEA*	*PHA*	*FFA*	*HAZOP*
FTA-AADL	-	X	-	-	-	-	-	-
AADL-AutoFT	X	-	-	-	-	-	-	-
AADL-FT	X	-	-	-	-	-	-	-
COMPASS	-	X	X	-	-	-	-	-
3+1 SysML-SE	-	-	-	-	-	X	-	-
MeDISIS	-	-	X	-	-	-	-	-
MBSE	-	X	X	-	-	X	-	-
UML-FMECA	-	-	-	X	-	-	-	-
ISAMDSD	-	X	-	X	-	-	-	-
UML-SAM	-	X	X	-	-	-	X	-
UML-SFMEA	-	-	-	-	X	-	-	-
AutoSFMEA	-	-	-	-	X	-	-	-
HAZOP-UML	-	-	-	-	-	-	-	X
AADL-GSPN	-	-	-	-	-	-	-	-

- *FTA-AADL.* The approach (Sun et al., 2007) takes into consideration the extension of *fault tree analysis* (*FTA*) (Storey, 1996), *product line FTA* (*PLFTA*). By combining PLFTA and AADL, the authors extend the scope of early safety analysis into the architectural stage of development.
- *AADL-AutoFT.* Joshi et al. (2007) aim at providing the safety engineers with the automatically generated *static fault trees* (*static FTs*). They point out that the result of safety analysis carried out manually by different safety engineers can be different for the same system. To avoid this ambiguity, they automate the static FT derivation.
- *AADL-FT.* The approach (Li et al., 2011) also utilises static FTs. In this approach, the use of AADL and its error model annex allows the safety engineers to construct fault trees consistently and effectively.

- *COMPASS.* Bozzano et al. (2009) aim at supporting the software engineers with a trustworthy modelling and analysis framework that allows them to automatically derive *dynamic fault trees* (*dynamic FTs*), *failure modes and effects analysis* (*FMEA*) (Storey, 1996), assessment of *fault detection, isolation and recovery* (*FDIR*) and observability of requirements.
- *3+1 SysML-SE.* Thramboulidis and Scholz (2010) adopt *preliminary hazard analysis* (*PHA*) (Storey, 1996). They apply PHA for the mechatronic systems to identify the list of hazards and their causes. The results are used for the risk analysis executed for each hazard. The final results of the analysis consist of the identification of the severity and probability of each hazard as well as the estimation of the associated risk.
- *MeDISIS.* The approach (David et al., 2008) proposes a solution for an automatic synthesis of FMEA from UML/SysML models. Here FMEA is used to find the dysfunctional, i.e., erroneous, behaviour of the system. This step is crucial because it identifies the failure modes that will be qualified and quantified at the next two steps, involving construction of a model integrating the functional and dysfunctional behaviours with Altarica DF as well as the analysis and quantification of the dysfunctional behaviour and the impact on requirements and timing constraints with AADL.
- *MBSE.* Belmonte and Soubiran (2012) propose an approach that combines different safety analysis techniques. They identify accidental scenarios using PHA and build a hierarchy of FMEA tables, where each FMEA corresponds to some function of the system. Hence, the safety and system models are built in parallel. Next, the authors perform model transformations to obtain an AltaRica model. Finally, they generate a set of accident sequences and a number of fault trees for sub-systems.
- *UML-FMECA.* The approach (Guiochet and Baron, 2003) demonstrates how the use of risk analysis techniques such as *failure modes, effects and criticality analysis* (*FMECA*) can be coupled with the object-oriented system modelling.
- *ISAMDSD.* DeMiguel et al. (2008) incorporate such safety analysis techniques as FTA and FMECA into the process of safety-aware software architecture modelling. The results of the analysis are used to detect inconsistencies in software architectures and safety requirements.
- *UML-SAM.* To address the problem of severity assessment, this approach (Hassan et al., 2005) adopts three different safety analysis techniques: *functional failure analysis* (*FFA*) (Papadopoulos and McDermid, 1999), FMEA and FTA. The identification of system hazards is done by performing FFA. The components/connectors failure modes are identified by applying FMEA. Finally, to construct a detailed cause and effect model, the authors propose to use FTA. Here FTA combines the results of FFA and FMEA analyses.
- *UML-SFMEA.* The approach proposed by Hecht, Xuegao and Hecht (2004) aims at automating major steps of a *software failure modes and effects analysis* (*SFMEA*). According to the authors, SFMEA can be used for the verification and validation of embedded software for real-time systems.

- *AutoSFMEA*. In Snooke and Price (2011), the reliability estimation of software is done by conducting SFMEA. It considers all potential faults in the system software (e.g., faulty inputs or software bugs), studies the consequences of faults and proposes actions to reduce the risks. Snooke (2004) relies on SFMEA to determine the impact of hypothetical faults on the system. SFMEA reduces the gap between the abstract metrics that assess overall design strategy and software testing that can only verify a small part of the system behaviour.
- *HAZOP-UML*. In Martin-Guillerez et al. (2010), the authors concentrate on the UML modelling of interactions between humans and a robotic system. The risk analysis on this model is performed with the guideword-based method – *hazard and operability* (*HAZOP*) method (Storey, 1996).
- *AADL-GSPN*. This approach focuses on generating dependability evaluation models from MDE approaches as well as structuring the dependencies between the system components in a systematic way. This is done to avoid errors in the resulting model of the system and thus facilitate its validation.

To summarise, the most commonly used safety analysis techniques are FTA and FMEA and their variations, i.e., static FT, FMECA, Software FMEA, etc. This is not surprising, since the considered approaches mainly focus on early stages of the system development, where such techniques bring the most significant benefits.

3.2.5 Quantitative/qualitative aspects of analysis criterion

The comparison of the approaches according to the quantitative/qualitative aspects of analysis criterion is given in Table 6. The approach AADL-GSPN is based on quantitative analysis. The quantitative measures are obtained from processing GSPN models. The approaches such as COMPASS, ISAMDSD, and UML-SAM can be considered both as quantitative and qualitative, because they rely on both quantitative and qualitative safety analysis techniques – FTA and FMEA/FMECA techniques. The other approaches are qualitative.

Table 6 Overview of approaches and criteria – quantitative/qualitative aspects of analysis

Approaches	*Quantitative*	*Qualitative*
FTA-AADL	X	-
AADL-AutoFT	X	-
AADL-FT	X	-
COMPASS	X	X
3+1 SysML-SE	X	-
MeDISIS	-	X
MBSE	-	X
UML-FMECA	-	X
ISAMDSD	X	X
UML-SAM	X	X
UML-SFMEA	-	X
AutoSFMEA	-	X
HAZOP-UML	-	X
AADL-GSPN	X	-

Table 7 Overview of approaches and criteria – availability of automation/tool support

Approaches	*Automation/tool support*
FTA-AADL	PLFT Eclipse plug-in
AADL-AutoFT	Plug-in for the OSATE framework to generate static FT and the CAFTA tool to analyse FT
AADL-FT	-
COMPASS	Integrated platform COMPASS
3+1 SysML-SE	-
MeDISIS	Automated FMEA generation
MBSE	Semi-formal tool for risk assessment
UML-FMECA	-
ISAMDSD	Safety modelling framework (SMF)
UML-SAM	-
UML-SFMEA	-
AutoSFMEA	-
HAZOP-UML	CASE tool
AADL-GSPN	ADAPT tool

3.2.6 Availability of automation/tool support criterion

The evaluation of the considered approaches according to the availability of an automated tool support is given in Table 7.

- *FTA-AADL.* Sun et al. (2007) have developed the product-line fault tree (PLFT) plug-in that is used to assist in making decisions regarding the system redundancy and safety features. The tool helps to generate a product-line fault tree for a specific product-line member from the previously constructed product-line fault tree. The fault tree created by the plug-in uses the information from the AADL error models to automatically perform basic analyses for each potential product-line member.
- *AADL-AutoFT.* Joshi et al. (2007) have developed a plug-in for the OSATE framework that allows for automated generation of static FTs. The plug-in produces the data that can be used by the commercial tool CAFTA to perform the fault tree analysis. According to the approach, to allow for fault tree generation, the information about the error model of the system is collected in the form of a *directed graph* (*DG*). The fault tree generation algorithm is a recursive algorithm that transforms a DG into a fault tree.
- *COMPASS.* In Bozzano et al. (2009, 2010b), the authors describe an integrated platform that combines extensions of the existing tools. It is built upon a symbolic model checker, probabilistic model checker and a requirements analysis tool. The input of the toolset is a SLIM model, the outputs are traces, (probabilistic) fault trees and FMEA tables as well as the diagnosability and performability measures.
- *MeDISIS.* The algorithm for automatic synthesis of FMEA is presented in David et al. (2008). The authors use sequence diagrams to create a FMEA table. The algorithm identifies the type of the component under consideration. Then it

searches for the relevant failure mode of the components in the database. The detailed overview of the algorithm is given in David et al. (2010). The algorithm includes the following steps: identifying system components, impacts on requirements, components relationships, impacts on system properties and performance, collecting failure modes in the database, and constructing the resulting FMEA table.

- *MBSE.* The approach is supported by a semi-formal modelling tool for risk assessment. Moreover, the authors give a detailed formalisation of the model transformation. They aim at creating a tool to derive an AltaRica model from both safety and system models. The tool could be used in cooperation with the available tools for fault tree generation.
- *ISAMDSD.* DeMiguel et al. (2008) use the safety modelling framework (SMF) that has a set of tools for the analysis and development of safety-aware UML architectures. It includes metamodels, UML profiles, model transformations and evaluation of metrics. The SMF tool chain consists of the following steps: coupling SMF and UML modelling tools, transition from UML models to *safety-aware architectures* (*SAA*), transition from SAA to safety analysis models (FTA/FMECA) and coupling with safety analysis tools.
- *HAZOP-UML.* Martin-Guillerez et al. (2010) have developed the CASE tool that maintains consistency between UML models and HAZOP tables as well as provides the document generation and management features. It is built as an Eclipse plug-in that allows the developers to construct UML use cases and sequence diagrams. By applying the guideword templates, it automatically generates the HAZOP tables. The resulting HAZOP tables can be filled in by the developers using the automatically generated lists: the list of guidewords, the list of deviations, the recommendation list and the corresponding hazards.
- *AADL-GSPN.* Rugina et al. (2008) propose a tool called ADAPT – AADL architectural models to stochastic petri nets through model transformation. The tool provides the developers with the capability to perform the evaluation of various dependability measures (i.e., reliability and availability) from the respective AADL models. ADAPT is essentially a set of Eclipse plug-ins that are based on model transformation rules.

3.2.7 Inputs and outputs criterion

We summarise the analysis of each approach by highlighting its inputs and outputs as shown in Table 8. In the remaining part of the paper, the notation (1), (2), etc. is used to emphasise that an approach consists of several steps. Each step may produce a result that can be considered as an independent output or as an intermediate input to the next step of the approach.

Table 8 Overview of approaches and criteria – inputs and outputs

Approaches	*Inputs*	*Outputs*
FTA-AADL	AADL PLFT	FT
AADL-AutoFT	AADL model	Static FT
AADL-FT (1)	AADL	Database
AADL-FT (2)	Database	Static FT
COMPASS	SLIM model	FMEA, FTA, FDIR, performability
3+1 SysML-SE (1)	Use cases + PHA	Hazards, causes
3+1 SysML-SE (2)	Hazards, causes	Severity/probability
MeDISIS (1)	SysML model	FMEA (DBD)
MeDISIS (2)	SysML model + DBD	AltaRica DF model
MeDISIS (3)	SysML model + DBD	AADL model
MeDISIS (4)	SysML model	Simulink model
MBSE (1)	PHA/ FMEA	SysML/DSML model
MBSE (2)	SysML/DSML model	AltaRica model + FTA, accident sequences
UML-FMECA	UML concept of message	FMECA
ISAMDSD	UML profile + SAA	FMECA, FTA
UML-SAM	UML + FFA, FMEA, FTA	Severity
UML-SFMEA	Use cases	SFMEA
AutoSFMEA	Use cases	SFMEA
HAZOP-UML	UML + HAZOP	Hazards, causes
AADL-GSPN	AADL model	GSPN model

- *FTA-AADL.* This approach allows the practitioners to automatically shorten and adapt a fault tree (FT) for a specific product from a previously constructed PLFT.
- *AADL-AutoFT.* The authors support the AADL-AutoFT approach with the possibility to automatically generate static FTs from the AADL models.
- *AADL-FT.* This method allows for constructing fault trees from AADL models. By the application of the approach, several results can be obtained. Firstly, it extracts the fault information from the given AADL models and stores it in a database structure (1). Secondly, it constructs a fault tree from the extracted fault information stored in the database (2).
- *COMPASS.* The approach takes as the input a SLIM model and a set of property patterns and produces a number of valuable outputs. Firstly, the supporting toolset generates traces, (probabilistic) fault trees and FMEA tables. Secondly, the approach itself allows for analysing diagnosability and FDIR. Finally, it supports the performance evaluation of the given SLIM model.
- *3+1 SysML-SE.* To obtain a list of hazards and a list of causes per each hazard, the authors adopt the concept of essential use cases in combination with PHA. The risk analysis, which is a part of PHA, is performed for each hazard (1). The output of this activity is the identification of the severity and the probability for each hazard as well as the estimation of the associated risk (2).
- *MeDISIS.* This method allows the developers to obtain several different outputs. Firstly, it supports automatic generation of FMEA tables from SysML models and

storage of the FMEA results in the *dysfunctional behaviour database* (*DBD*) (1). Secondly, it allows for the construction of an AltaRica DF model from the results of the functional and dysfunctional behaviour analysis (SysML + DBD) (2). Thirdly, it allows the transformation of the given SysML models into AADL models. The obtained models incorporate the dysfunctional behaviour (3). Finally, it supports the generation of Simulink models from SysML models (4).

- *MBSE*. The approach describes a model to model translation, where the input models are a SysML model and a *domain specific modelling language* (*DSML*) model, while the output model is an AltaRica model (2). The SysML model represents the functional part of the system model, while the dysfunctional part is modelled using a specially developed language DSML. To derive a SysML model, the authors use PHA. To obtain a DSML model, they rely on FMEA of each function of the system (1). Additionally, the approach allows for generation of fault trees from the given AltaRica models.
- *UML-FMECA*. This approach focuses on the application of FMECA for the UML concept of *message* and its failure mode analysis. The resulting output is a FMECA table for each message failure mode.
- *ISAMDSD*. The authors use a metamodel, which represents safety concepts of the component-based SAA, for the internal representation of architectures. The input of the ISAMDSD approach is the architectural models obtained from the given UML models. The output is a set of tables (FMECA), trees (FMECA, FTA) and diagrams (FTA).
- *UML-SAM*. The UML-based severity assessment is performed combining three different hazard analysis techniques: FFA, FMEA and FTA. The authors use UML models to automate the process of estimating severity.
- *UML-SFMEA*. To generate SFMEA, UML use case models form the input to the UML-SFMEA approach. The obtained SFMEA is used to highlight the conditions that cause the highest severity failures as well as the parts of a programme where the detection and recovery procedures are missing or covered incompletely.
- *AutoSFMEA*. According to this approach, the resulting code-level software FMEA is generated from the UML use case diagrams (partial functional models).
- *HAZOP-UML*. HAZOP attributes and guidewords are utilised for generic interpretation of UML use cases and sequence diagrams. The output of the approach is a list of identified possible hazards and their causes.
- *AADL-GSPN*. This approach allows the developers to automatically obtain a GSPN model from an AADL model.

4 Evaluation of the formal approaches

4.1 Formal development approaches

The use of formal methods in development of safety-critical systems is guided by the functional safety standard IEC 61508 (2010). IEC 61508 sets four *safety integrity levels*

(*SILs*), where SIL 1 is the lowest and SIL 4 is the highest one. The higher the SIL, the more formal and structured design methods should be used. The MDD approaches can be suitable up to SIL 3, while the use of formal methods is recommended for SIL 2 and SIL 3, and highly recommended for SIL 4 (Smith and Simpson, 2010).

Barroca and McDermid (1992) suggest that formal methods can be used in two ways. On the one hand, they can be used for development of system specifications. On the other hand, the obtained formal specifications can be used as a basis for verifying the correctness of software.

In our survey, we consider the formal approaches that have been directly linked with safety analysis techniques and aimed at facilitating the safety-critical systems development, verification and validation. The list of the selected formal approaches with the relevant citations and abbreviations is given in Table 9.

Table 9 Considered formal approaches

Approach	*Citation*	*Abbreviation*
Computer-aided PHA, FTA and FMEA for automotive embedded systems	Mader et al. (2011)	CASAAES
Patterns for representing FMEA in formal specification of control systems	Lopatkin et al. (2011a, 2011b)	FMEA-patterns
Verifying formal specifications using fault tree analysis	Liu (2000)	VFSFTA
Formal safety analysis in transportation control	Thums and Schellhorn (2002)	FSATC
Combining formal methods and safety analysis	Ortmeier et al. (2004)	ForMoSA
AltaRica	Bieber et al. (2002) Chaudemar et al. (2009)	AltaRica
Mode logic for layered control systems	Prokhorova et al. (2011)	MLLCS

To perform an analysis of the surveyed formal approaches, we have chosen the same evaluation criteria as for the model-driven approaches: *domain-specific, language-specific, life cycle stages, safety analysis techniques, quantitative/qualitative aspects of analysis, availability of automation/tool support, inputs and outputs*.

4.2 Results of evaluation

4.2.1 Domain-specific criterion

Similar to the MDD approaches, the majority of the surveyed formal approaches are generic. They are used to develop correct, complete and safe formal models of control systems related to different application domains. The evaluation results of the approaches according to the domain-specific criterion are presented in Table 10. Among the considered approaches, three are domain-specific. The approach CASAAES has been developed for the automotive domain, FSATC has been used for the control systems in transportation, and AltaRica has been specifically developed for the avionics. Moreover, let us point out that the VFSFTA approach differs from the rest of the approaches because it focuses not on deriving but rather on verifying formal specifications.

Table 10 Overview of approaches and criteria – domain-specific

Approaches	*Application domain*	*Specialisation*
CASAAES	Automotive	Embedded systems
FMEA-patterns	General	Safety-critical control systems
VFSFTA	General	Formal specifications
FSATC	Transportation	Control systems
ForMoSA	General	Critical embedded systems
AltaRica	Avionics	Complex systems
MLLCS	General	Mode-rich layered control systems

4.2.2 Language-specific criterion

The considered formal approaches utilise different modelling languages (Table 11). The CASAAES approach is based on the domain-specific language EAST-ADL (ATESST2, 2010). The VFSFTA approach uses *structured object-oriented formal language* (*SOFL*) for verification. Two approaches (FSATC and ForMoSA) rely on *interval temporal logic* (*ITL*). The AltaRica approach is based on the AltaRica formal language (Point and Rauzy, 1999). There exists also a combination of this language and Event-B formalism (Event-B, 2012) presented in Chaudemar et al. (2009). The approaches FMEA-patterns and MLLCS also use Event-B. Such a variety of formal languages does not allow us to single out the most popular language or divide the approaches into groups according commonly used languages. Only two formalisms that have been used by more than one approach are ITL and Event-B. However, this is still insufficient evidence for us to state that these two formalisms are the most widely adopted in the safety-oriented model-based development of critical systems.

Table 11 Overview of approaches and criteria – language-specific

Approaches	*Languages*				
	EAST-ADL	*SOFL*	*ITL*	*AltaRica*	*Event-B*
CASAAES	X	-	-	-	-
FMEA-patterns	-	-	-	-	X
VFSFTA	-	X	-	-	-
FSATC	-	-	X	-	-
ForMoSA	-	-	X	-	-
AltaRica	-	-	-	X	X
MLLCS	-	-	-	-	X

4.2.3 Life cycle stages criterion

Table 12 shows at which stages of the system life cycle the chosen formal approaches can be employed. In particular, they are used for derivation and formalisation of safety requirements (four approaches), architecting (three approaches) and designing (five approaches) of system models as well as verification and validation of these models (six approaches). Moreover, the results of the analysis show that, in contrast to the MDD approaches, the formal approaches are equally often used at the early and later stages of the life cycle. However, the selected formal approaches are not used at the *implementation* stage.

Table 12 Overview of approaches and criteria – life cycle stages

Approaches	*Life cycle stages*				
	Requirements	*Architecture*	*Design*	*Implementation*	*V&V*
CASAAES	X	-	-	-	-
FMEA-patterns	X	X	X	-	X
VFSFTA	-	-	-	-	X
FSATC	X	-	X	-	X
ForMoSA	-	-	X	-	X
AltaRica	-	X	X	-	X
MLLCS	X	X	X	-	X

4.2.4 Safety analysis techniques criterion

The results of analysis given in Table 13 show that the majority of approaches utilise the FTA technique. Three approaches (CASAAES, FSATC and AltaRica) combine several techniques. The CASAAES approach applies PHA, FTA and FMEA. PHA is used to identify and classify hazards and to define the top-level safety requirements, while FTA and FMEA are used for verification of the system architecture. The CASAAES approach allows for establishing the consistency between the results of PHA and FTA /FMEA. In the FSATC approach, the authors apply PHA to create a list of hazards, which is later used as the basis for conducting FTA. *Functional hazard analysis* (*FHA*) (Wilkinson and Kelly, 1998) is adopted by the AltaRica approach to achieve the safe objectives required by the domain-specific standards. AltaRica also permits the generation of fault trees to describe the failure situations leading to an unexpected event. Two approaches, FMEA-patterns and MLLCS, focus on the FMEA technique. In the FMEA-patterns approach, the FMEA results are formalised via a set of generic patterns, i.e., the authors establish the correspondence between fields of a FMEA table and different types of patterns. The authors of the MLLCS approach tailor the FMEA table structure in order to augment the mode transition logic (which is one of the main objectives of the approach). They propose to conduct FMEA of each operational mode of a system to identify mode transitions required to implement fault tolerance.

Table 13 Overview of approaches and criteria – safety analysis techniques

Approaches	*Safety analyses*			
	FTA	*FMEA*	*PHA*	*FHA*
CASAAES	X	X	X	-
FMEA-patterns	-	X	-	-
VFSFTA	X	-	-	-
FSATC	X	-	X	-
ForMoSA	X	-	-	-
AltaRica	X	-	-	X
MLLCS	-	X	-	-

In comparison to the MDD approaches, the formal ones utilise even fewer safety analysis techniques. For example, none of the surveyed formal approaches uses HAZOP or FMECA. One of a potentially advantageous use of FMECA, would allow the designers to augment formal models with evaluation of the probability of a component failure and severity of consequences. The lack of such techniques leads to the limitations considered in the next paragraph.

4.2.5 Quantitative/qualitative aspects of analysis criterion

The classification of the approaches based on the quantitative/qualitative aspects criterion is given in Table 14. Though the majority of formal approaches utilise FTA, they cannot be classified as quantitative. In these approaches, FTA is used to examine how the component faults and failures contribute to the top hazard occurrence. The only approach that employs the quantitative FTA for assessing failure probabilities is ForMoSA. A quantitative approach to linking formal Event-B models and fault trees has been also proposed by Tarasyuk et al. (2011).

Table 14 Overview of approaches and criteria – quantitative/qualitative aspects of analysis

Approaches	*Quantitative*	*Qualitative*
CASAAES	-	X
FMEA-patterns	-	X
VFSFTA	-	X
FSATC	-	X
ForMoSA	X	X
AltaRica	-	X
MLLCS	-	X

4.2.6 Availability of automation/tool support criterion

The role of formal methods and their automated tool support was defined by Heitmeyer (2005). However, only three out of seven considered approaches are well automated (Table 15). The CASAAES approach provides a plug-in for the open source tool Papyrus

that supports property checking, model correction, fault tree generation and FMEA table generation. The incorporation of FMEA results in formal specifications in the FMEA-patterns approach is automated by a plug-in for the Rodin platform – a modelling framework for the Event-B formalism. It adds simple user interface features for model instantiation, extends the user input facility with suggesting necessary Event-B elements, and provides a library of Event-B and FMEA-specific transformations. The AltaRica approach is supported by the AltaRica fault tree generator that generates a fault tree from an AltaRica model. The generated fault tree is then analysed by the ARALIA fault tree analyser to formally define unexpected combinations of failures.

Table 15 Overview of approaches and criteria – availability of automation/tool support

Approaches	*Automation/tool support*
CASAAES	Plug-in for the tool Papyrus
FMEA-patterns	Plug-in for the Rodin platform
VFSFTA	-
FSATC	-
ForMoSA	-
AltaRica	AltaRica fault tree generator and ARALIA fault tree analyser
MLLCS	-

4.2.7 Inputs and outputs criterion

We give an evaluation of the selected formal approaches according to this criterion in Table 16. Some approaches produce valuable intermediate results listed in the order they are produced.

- *CASAAES*. The toolset supporting the approach takes as an input an EAST-ADL model and generates qualitative FTA and FMEA as the output. This approach ensures consistency of PHA results, fault trees and FMEA tables.
- *FMEA-patterns*. The safety and fault tolerance requirements derived from FMEA are implemented as formal Event-B models by the means of interactive pattern instantiation and automatic model transformation.
- *VFSFTA*. The approach allows for verification of a formal specification using FTA. At the first step, a fault tree reflecting the relationship between the top event, i.e., the target task to be verified, and the leaf events, i.e., events that lead to the top event, is derived from the specification. All the leaf events are verified (1). At the second step, the decision on whether the top event can occur is made (2).
- *FSATC*. The input of this approach is an informal model. The approach supports the development of formal models and formalised fault trees for all the identified hazards. It is used for proving the correctness and completeness of the constructed fault trees from a formal model. The output of the approach is a correct and safe formal model with the verified safety properties.
- *ForMoSA*. Similarly to FSATC, this approach takes as an input an informal model and produces the resulting formal specification, validated via the generated proof obligations (POs) of FTA.

- *AltaRica.* According to this approach, the resulting system model, which meets safety requirements, is obtained by formalising the safety requirements derived from FHA (1). The safety requirements of the system are checked using fault tree generation and analysis (2).
- *MLLCS.* The approach allows the developers to derive the fault tolerance part of a mode logic using FMEA (1). Moreover, it shows how to formalise the required conditions of mode consistency and how to ensure them while developing a system using a formal language (2).

Table 16 Overview of approaches and criteria – inputs and outputs

Approaches	*Inputs*	*Outputs*
CASAAES	EAST-ADL model	FT, FMEA
FMEA-patterns	Safety requirements derived from FMEA	Formal specification
VFSFTA (1)	Formal specification	FT
VFSFTA (2)	FT	Formal specification
FSATC	Informal model	Correct and safe formal model
ForMoSA	Informal specification	Formal model validated by checking POs of FTA
AltaRica (1)	Safety requirements derived from FHA	AltaRica model that meets safety requirements
AltaRica (2)	AltaRica model	FT
MLLCS (1)	Safety requirements, system modes and forward mode transitions	FMEA
MLLCS (2)	FMEA	Formal specification

5 Discussion

While searching and analysing the literature on the topic of our survey, we have found a significant number of available approaches that utilise safety analysis techniques in the model-based development process. Moreover, during the period of conducting our research several new works have appeared. This shows that the area is rapidly developing. Hence, we believe that reassessing the sate-of-the-art in the field is highly important and can be useful for the research community and industrial practitioners in safety-critical domain and reliability engineering in general.

The latter part of the audience would be also interested in the approaches that more broadly address the reliability and availability requirements in model-based development. We overview the most prominent approaches below. Bondavalli et al. (2001) proposed an approach to integrating UML with a set of validation, verification and evaluation techniques through model transformation. The transformation maps UML diagrams to *timed petri nets*. The work conducted by Huszerl et al. (2002) introduces a method for the quantitative dependability analysis of systems modelled using a particular type of UML diagrams – statechart diagrams. This analysis is based on transformation from UML statechart diagrams to *stochastic reward nets*. The dependability concern is addressed via the notion of state perturbations (i.e., states corresponding to the degraded

system behaviour, transitions to such states, etc.). Bernardi et al. (2010) proposed an approach that aims at improving the elicitation, documentation and analysis of reliability and availability requirements within UML profiles. They adopt the *dependability analysis and modelling* (*DAM*) profile, which is a specialisation of the *modelling and analysis of real time and embedded systems* (*MARTE*) profile, to describe software requirements. To determine and mitigate combinations of system faults and failures, the authors propose to incorporate FTA into the requirements gathering process.

There are also several formal development approaches that do not directly integrate safety analysis but still are relevant for development of dependable systems. The problem of combining MDE with *formal methods* (*FMs*) is presented in Gargantini et al. (2010). The authors mainly focus on the development of a tool-set around *abstract state machines* (*ASMs*) in order to support its practical use in the system development life cycle. The application of MDE to FMs and the application of FMs to MDE are considered as an in-the-loop integration approach. Formal models are incorporated into MDE as domain-specific languages by developing their metamodels. The ASM metamodel is defined as an abstract syntax description of a language for ASMs. The paper (Cichocki and Górski, 2001) shows how to use the *communicating sequential processes* (*CSP*) notation as a formal method supporting the FMEA process and the associated tool FDR to identify component faults and to analyse the consequences those faults can have in the higher-level system components.

Furthermore, while surveying various modern approaches to the safety-oriented model-based development, we cannot ignore the META toolset proposed by RC META (2013). It is an Eclipse-based toolset that consists of several parts. Firstly, it allows for SysML-AADL translation (SysML models are created using enterprise architect and translated to AADL by a plug-in). Secondly, it includes tools for system design and analysis: EDICT with architectural design patterns and OSATE, which is used for constructing and editing system architecture models in AADL. Finally, it supports the structural verification (Lute) and the compositional behaviour verification (AGREE). The META toolset is mature and has extensive industrial use, especially in avionics. Versatility and the integrated nature of the approach would not allow us to classify it in the same way as the surveyed techniques.

6 Conclusions

In this paper, we have presented an overview of 14 safety-oriented model-driven and seven safety-oriented formal approaches to development of critical systems. The overview results reveal that there is no unified model-based approach that would facilitate achieving different levels of safety (i.e., SILs), different domain needs and all stages of the system development process.

Our survey has identified that the initial stages of system development are better supported. This can be explained by more abstract and, hence, succinct models that can be managed well by the safety analysis techniques. However, it also identifies a lack of the approaches that would support safety analysis at the implementation level. The survey has shown that the majority of the approaches are domain-neutral. On the one hand, genericity guarantees better flexibility. On the other hand, it might require additional efforts to tailor the chosen approach to the domain-specific needs.

The comparison between AADL- and UML/SysML-oriented MDD approaches shows that both strands are equally well supported. However, the availability of automated tool support is better for AADL. The survey highlighted the lack of domain-specific applications that would directly link the domain-specific MDD and the safety analysis techniques. The most prominent approaches that have been identified are COMPASS and MeDISIS because they support a wide range of the safety analysis techniques and have mature tool support.

In general, the number of the formal approaches that integrate safety analysis is lower than model-driven ones. Moreover, the formal approaches are associated with a smaller number of the safety analysis techniques. The most commonly used technique is FTA. Formal techniques have less extensive tool support – only three (CASAAES, FMEA-patterns and AltaRica) out of seven have mature tool support. In contrast, the half of surveyed MDD approaches is automated.

The survey has highlighted several issues in the field of safety-oriented model-based development. Firstly, currently none of the approaches integrates safety analysis into all stages of the system development life cycle. Therefore, the complete development process inevitably needs to combine several approaches. Secondly, formal approaches should be more actively extended with the quantitative safety analysis techniques. Finally, the last but not least problem to be solved is the lack of mature tool support that would allow the safety engineers to adopt the proposed approaches in the industrial practice.

Acknowledgements

The authors would like to thank Linas Laibinis for fruitful discussions as well as the anonymous reviewers for the many helpful suggestions.

References

AADL (2012) ‘AADL’ [online] http://www.aadl.info/ (accessed 20 September 2012).

ATESST2 (2010) *ATESST2 Project Consortium: EAST-ADL Domain Model Specification*, Technical Report Version 2.1, Release Candidate 3.

Event-B (2012) ‘Event-B and the Rodin platform’ [online] http://www.Event-B.org/ (accessed 12 October 2012).

IEC61508 (2010) International Electrotechnical Commission, IEC 61508, functional safety of electrical/electronic/programmable electronic safety-related systems.

OMG SysML (2012) ‘OMG Systems Modeling Language’ [online] http://www.omgsysml.org/ (accessed 5 September 2012).

OMG UML (2012) ‘OMG Unified Modeling Language’ [online] http://www.uml.org/ (accessed 5 September 2012).

RC META (2013) ‘The Rockwell Collins META toolset’ [online] https://wiki.sei.cmu.edu/aadl/index.php/RC_META (accessed 5 February 2013).

SDLC (2012) ‘Systems development life cycle: objectives and requirements’ [online] http://www.benderrbt.com/Bender-SDLC.pdf (accessed 20 September 2012).

Barroca, L. and McDermid, J. (1992) ‘Formal methods: use and relevance for the development of safety critical systems’, *The Computer Journal*, Vol. 35, No. 6, pp.579–599.

Belmonte, F. and Soubiran, E. (2012) 'A model based approach for safety analysis', in *Computer Safety, Reliability, and Security (SAFECOMP12)*, Springer Berlin Heidelberg, Vol. 7613 of *LNCS*, pp.50–63.

Bernardi, S., Merseguer, J. and Lutz, R. (2010) 'Reliability and availability requirements engineering within the unified process using a dependability analysis and modeling profile', in *Proc. of the 9th European Dependable Computing Conference (EDCC10)*, pp.95–104.

Bernardi, S., Merseguer, J. and Petriu, D.C. (2012) 'Dependability modeling and analysis of software systems specified with UML', *ACM Comput. Surv.*, Vol. 45, No. 1, pp.2:1–2:48.

Bieber, P., Castel, C. and Seguin, C. (2002) 'Combination of fault tree analysis and model checking for safety assessment of complex system', in *Proc. of the 4th European Dependable Computing Conference on Dependable Computing*, Springer-Verlag, pp.19–31.

Bondavalli, A., Dal Cin, M., Latella, D., Majzik, I., Pataricza, A. and Savoia, G. (2001) 'Dependability analysis in the early phases of UML-based system design', *Journal of Computer Systems Science and Engineering*, Vol. 16, No. 5, pp.265–275.

Bozzano, M., Cimatti, A., Katoen, J-P., Nguyen, V., Noll, T. and Roveri, M. (2009) 'The COMPASS approach: correctness, modelling and performability of aerospace systems', in *Proc. of International Conference on Computer Safety, Reliability and Security (SAFECOMP09)*, pp.173–186.

Bozzano, M., Cavada, R., Cimatti, A., Katoen, J-P., Nguyen, V., Noll, T. and Olive, X. (2010a) 'Formal verification and validation of AADL models', in *Proc. of Embedded Real Time Software and Systems Conference (ERTS10)*, pp.1–9.

Bozzano, M., Cimatti, A., Katoen, J-P., Nguyen, V., Noll, T. and Roveri, M. (2010b) 'Safety, dependability and performance analysis of extended AADL models', *The Computer Journal*, Vol. 54, No. 5, pp.754–775.

Cabot, J. and Yu, E. (2008) 'Improving requirements specifications in model-driven development processes', in *Proc. of International Workshop on Challenges in Model-Driven Software Engineering (ChaMDE08)*, pp.36–40.

Chaudemar, J-C., Bensana, E., Castel, C. and Seguin, C. (2009) 'AltaRica and Event-B models for operational safety analysis: unmanned aerial vehicle case study', in *Proc. of Workshop on Integration of Model-based Formal Methods and Tools*, pp.15–19.

Chiola, G., Marsan, M., Balbo, G. and Conte, G. (1993) 'Generalized stochastic petri nets: a definition at the net level and its implications', *IEEE Transactions on Software Engineering*, Vol. 19, No. 2, pp.89–107.

Cichocki, T. and Górski, J. (2001) 'Formal support for fault modelling and analysis', in U. Voges (Ed.): *Proc. of the 20th International Conference on Computer Safety, Reliability and Security (SAFECOMP01) LNCS 2187*, pp.190–199, Springer-Verlag Berlin Heidelberg.

Cressent, R., Idasiak, V., Kratz, F. and David, P. (2011) 'Mastering safety and reliability in a model based process', in *Proc. of Reliability and Maintainability Symposium (RAMS11)*, pp.1–6.

David, P., Idasiak, V. and Kratz, F. (2008) 'Towards a better interaction between design and dependability analysis: FMEA derived from UML/SysML models', in *Proc. of ESREL 2008 and the 17th SRA-EUROPE Annual Conference*.

David, P., Idasiak, V. and Kratz, F. (2009) 'Automating the synthesis of AltaRica data-flow models from SysML', in *Reliability, Risk, and Safety, Three Volume Set: Theory and Applications; Proc. of European Safety and Reliability Conference (ESREL09)*, pp.105–112.

David, P., Idasiak, V. and Kratz, F. (2010) 'Reliability study of complex physical systems using SysML', *Reliability Engineering and System Safety*, Vol. 95, No. 4, pp.431–450.

DeMiguel, M., Briones, J., Silva, J. and Alonso, A. (2008) 'Integration of safety analysis in model-driven software development', *IET Software*, Vol. 2, No. 3, pp.260–280.

Feiler, P.H. and Gluch, D.P. (2012) *Model-Based Engineering with AADL: An Introduction to the SAE Architecture Analysis and Design Language*, Addison-Wesley Professional, Westford, Massachusetts, USA.

France, R. and Rumpe, B. (2007) 'Model-driven development of complex software: a research roadmap', in *Future of Software Engineering (FOSE07)*, pp.37–54.

Gargantini, A., Riccobene, E. and Scandurra, P. (2010) 'Combining formal methods and MDE techniques for model-driven system design and analysis', *International Journal on Advances in Software*, Vol. 3, Nos. 1, 2, pp.1–18.

Gaudel, M-C. (1991) 'Advantages and limits of formal approaches for ultra-high dependability', in *Proc. of the 6th International Workshop on Software Specification and Design*', pp.237–241.

Giese, H. and Henkler, S. (2006) 'A survey of approaches for the visual model-driven development of next generation software-intensive systems', *J. Vis. Lang. Comput.*, Vol. 17, No. 6, pp.528–550.

Guiochet, J. and Baron, C. (2003) 'UML based FMECA in risk analysis', in *Proc. of the European Simulation and Modelling Conference (ESMc '03)*, pp.1–8.

Guiochet, J., Motet, G., Baron, C. and Boy, G. (2004) 'Toward a human-centered UML for risk analysis. Application to a medical robot', in *Proc. of Human Error, Safety and Systems Development, IFIP 18th World Computer Congress*, pp.177–192.

Guiochet, J., Martin-Guillerez, D. and Powell, D. (2010) 'Experience with model-based user-centered risk assessment for service robots', in *Proc. of the 12th IEEE High Assurance Systems Engineering Symposium (HASE10)*, pp.104–113.

Hassan, A., Goseva-Popstojanova, K. and Ammar, H. (2005) 'UML based severity analysis methodology', in *Proc. of Reliability and Maintainability Symposium (RAMS05)*, pp.158–164.

Hecht, H., Xuegao, A. and Hecht, M. (2004) 'Computer-aided software FMEA for unified modeling language based software', in *Proc. of Reliability and Maintainability Annual Symposium (RAMS04)*, pp.243–248.

Heitmeyer, C. (2005) 'Developing safety-critical systems: the role of formal methods and tools', in T. Cant (Ed.): *Proc. of the 10th Australian Workshop on Safety-Related Programmable Systems (SCS 2005)*, Vol. 55 of *CRPIT*, pp.95–99, ACS, Sydney, Australia.

Huszerl, G., Majzik, I., Pataricza, A., Kosmidis, K. and Cin, M.D. (2002) 'Quantitative analysis of UML statechart models of dependable systems', *The Computer Journal*, Vol. 45, No. 3, pp.260–277.

Hutchinson, J., Rouncefield, M. and Whittle, J. (2011) 'Model-driven engineering practices in industry', in *Proc. of the 33rd International Conference on Software Engineering (ICSE11)*, ACM, New York, NY, USA, pp.633–642.

Joshi, A., Vestal, S. and Binns, P. (2007)' Automatic generation of static fault trees from AADL models', in *Proc. of the IEEE/IFIP Conference on Dependable Systems and Networks' Workshop on Dependable Systems, DSN07-WADS*.

Knight, J. (2002) 'Safety critical systems: challenges and directions', *in Proc. of the 24th International Conference on Software Engineering*, ACM, New York, NY, USA, pp.547–550.

Li, Y., Zhu, Y., Ma, C. and Xu, M. (2011) 'A method for constructing fault trees from AADL models', in J.M. Calero, L.T. Yang, F.G. Mármol, L.J.G. Villalba, A.X. Li and Y. Wang (Eds.): *Automatic and Trusted Computing (ATC11) LNCS 6906*, pp.2430–258, Springer-Verlag Berlin Heidelberg.

Liu, S. (2000) 'Verifying formal specifications using fault tree analysis', in *Proc. of International Symposium on Principle of Software Evolution*, IEEE Computer Society Press, pp.1–10.

Lopatkin, I., Iliasovi, A., Romanovsky, A., Prokhorova, Y. and Troubitsyna, E. (2011a) 'Patterns for representing FMEA in formal specification of control systems', in *Proc. of the 13th International Symposium on High-Assurance Systems Engineering (HASE11)*, IEEE Computer Society, pp.146–151.

Lopatkin, I., Prokhorova, Y., Troubitsyna, E., Iliasov, A. and Romanovsky, A. (2011b) *Patterns for Representing FMEA in Formal Specification of Control Systems*, Technical Report TUCS 1003.

Mader, R., Armengaud, E., Leitner, A., Kreiner, C., Bourrouilh, Q., Grießnig, G., Steger, C. and Weiß, R. (2011) 'Computer-aided PHA, FTA and FMEA for automotive embedded systems', in *Proc. of International Conference on Computer Safety, Reliability and Security (SAFECOMP11)*, pp.113–127.

Martin-Guillerez, D., Guiochet, J., Powell, D. and Zanon, C. (2010) 'A UML-based method for risk analysis of human-robot interaction', in *Proc. of the 2nd International Workshop on Software Engineering for Resilient Systems (SERENE10)*, pp.32–41.

Mellor, S., Clark, A. and Futagami, T. (2003) 'Guest editors' introduction: model-driven development', *IEEE Software*, Vol. 20, No. 5, pp.14–18.

Ortmeier, F., Thums, A., Schellhorn, G. and Reif, W. (2004) *Combining Formal Methods and Safety Analysis – The ForMoSA Approach*, in SoftSpez Final Report, pp.474–493.

Papadopoulos, Y. and McDermid, J. (1999) 'Hierarchically performed hazard origin and propagation studies', in *Computer Safety, Reliability and Security (SAFECOMP99) LNCS 1698*, Springer-Verlag, pp.139–152.

Point, G. and Rauzy, A. (1999) 'AltaRica – constraint automata as a description language', *European Journal on Automation*, Vol. 33, Nos. 8–9, pp.1033–1052.

Prokhorova, Y., Laibinis, L., Troubitsyna, E., Varpaaniemi, K. and Latvala, T. (2011) 'Derivation and formal verification of a mode logic for layered control systems', in *Proc. of the 18th Asia-Pacific Software Engineering Conference (APSEC11)*, IEEE Conference Publishing Services, pp.49–56.

Rugina, A-E., Kanoun, K. and Kaâniche, M. (2007) 'Architecting dependable systems iv', Springer-Verlag, Berlin, Heidelberg, chapter *A System Dependability Modeling Framework Using AADL and GSPNs*, pp.14–38.

Rugina, A-E., Kanoun, K. and Kaâniche, M. (2008) 'The ADAPT tool: from AADL architectural models to stochastic petri nets through model transformation', in *Proc. of the 7th European Dependable Computing Conference (EDCC08)*, pp.85 –90.

Selic, B. (2003) 'The pragmatics of model-driven development', *IEEE Software*, Vol. 20, No. 5, pp.19–25.

Smith, D.J. and Simpson, K.G.L. (2010) *Safety Critical Systems Handbook: A Straightforward Guide to Functional Safety*, IEC 61508 (2010 Edition) and Related Standards, Elsevier.

Snooke, N. and Price, C. (2011) 'Model-driven automated software FMEA', in *Proc. of Reliability and Maintainability Symposium (RAMS11)*, pp.1–6.

Snooke, N.A. (2004) 'Model-based failure modes and effects analysis of software', in *Proc. of DX04*, pp.221–226.

Storey, N. (1996) *Safety-Critical Computer Systems*, Addison-Wesley, Harlow, UK.

Straeten, R., Mens, T. and Baelen, S. (2009)'Models in software engineering', Springer-Verlag, Berlin, Heidelberg, chapter *Challenges in Model-Driven Software Engineering*, pp.35–47.

Sun, H., Hauptman, M. and Lutz, R. (2007) 'Integrating product-line fault tree analysis into AADL models', in *Proc. of the 10th IEEE In High Assurance Systems Engineering Symposium (HASE07)*, pp.15–22.

Tarasyuk, A., Troubitsyna, E. and Laibinis, L. (2011) 'Quantitative verification of system safety in Event-B', in E. Troubitsyna (Ed.): *Software Engineering for Resilient Systems*, Vol. 6968 of *Lecture Notes in Computer Science*, pp.24–39, Springer Berlin Heidelberg.

Thramboulidis, K. and Buda, A. (2010) '3+1 SysML view model for IEC61499 function block control systems', in *Proc. of the 8th IEEE International Conference on Industrial Informatics (INDIN10)*, pp.175–180.

Thramboulidis, K. and Scholz, S. (2010) 'Integrating the 3+1 SysML view model with safety engineering', in *Proc. of IEEE International Conference on Emerging Technologies and Factory Automation (ETFA10)*, pp.1–8.

Thramboulidis, K. (2010) 'The 3+1 SysML view-model in model integrated mechatronics', *Journal of Software Engineering and Applications*, Vol. 3, No. 2, pp.109–118.

Thums, A. and Schellhorn, G. (2002) 'Formal safety analysis in transportation control', in *Proc. of the Workshop on Software Specification for Safety Relevant Transportation Control Tasks*.

Wilkinson, P. and Kelly, T. (1998) 'Functional hazard analysis for highly integrated aerospace systems', in *Proc. of Certification of Ground/Air Systems Seminar*.

Woodcock, J., Larsen, P., Bicarregui, J. and Fitzgerald, J. (2009) 'Formal methods: practice and experience', *ACM Computing Surveys*, Vol. 41, No. 4, pp.1–36.