Design of Dependable Systems: An Overview of Analysis and Verification Approaches

Jose Ignacio Aizpurua, Eñaut Muxika
Department of Signal Theory and Communications
University of Mondragon
Spain
{*jiaizpurua,emuxika*}*@mondragon.edu*

***Abstract*—Designing a dependable system successfully is a challenging issue that is an ongoing research subject in the literature. Different approaches have been adopted in order to identify, analyse and verify the dependability of a system design. This process is far from obvious and often hampered due to the limitations of the classical dependability analysis techniques and verification approaches. This paper provides an overview of analysis approaches grouped by limitations. The principal points for the characterization of the considered approaches are the capability to handle notions of time, component-wise failure propagations and the use of architectural languages with the aim to extract analysis models from design models. Finally, verification approaches are partially reviewed.**

Keywords-Dependability design; Dependability Analysis; Dependability Verification; Model-Based Analysis.

I. INTRODUCTION

The goal of this paper is to provide a list of sources to those readers who are not familiar to the field of model-based design of dependable systems. Our goal is not to exhaustively evaluate the specific features of these approaches, but to aggregate a comprehensive list of works grouped by their main characteristics.

In computing systems, dependability is defined as "ability of a system to deliver a service that can be justifiably trusted" [1]. Such trustworthiness is based on the assurance of dependability requirements. These requirements are defined through dependability attributes: *Reliability*, *Availability*, *Maintainability*, *Safety (RAMS)*, *confidentiality* and *integrity*. The scope of this overview focuses on RAMS attributes. Consequently, security aspects (confidentiality and integrity) are not addressed.

Reliability is the ability of an item to perform a required function under given conditions for a stated period of time [2]. *Maintainability* is the ability to undergo repairs and modifications to restore to a state in which the system can perform its required actions. *Availability* is the readiness for correct service and *safety* is the absence of catastrophic consequences on the user(s) and the environment.

This survey concentrates on three main phases: dependability analysis, system design and verification. Despite being aware of the relevance of software code for system dependability in each of these phases, we will consider software code as a black box component to limit the extension of this paper (interested readers refer to [3] [4]).

Dependability analysis techniques can be organised by looking at how different system failures are characterized with its corresponding underlying formalisms. On one hand, *event-based* approaches reflect the system functionality and failure behaviour through combination of events. This analysis results in either Fault Tree (FT) like [5] or Failure Mode and Effect Analysis (FMEA) like [6] structures, which emphasizes the reliability and safety attributes. On the other hand, *state-based* approaches map the analysis models into a state-based formalism (e.g., Stochastic Petri Nets (SPN)). Those approaches analyse system changes with respect to time and centre on reliability and availability attributes.

Fault injection and model-checking [7] approaches are mainly adopted for model-based analysis and verification of design decisions. Principally, they are aimed at checking and evaluating dependability requirements using nominal and failure behaviour models. This overview addresses the analysis and verification of system properties using these approaches. There is also another class of verification approaches, which try to ensure the validity of the system by design [8] (i.e., formal verification).

The remainder of this paper is organized as follows: Section II classifies dependability analysis techniques based on the limitations of classical techniques. Section III studies how to adopt these approaches when designing a dependable system. Section IV discusses the characteristics of verification approaches when designing a dependable system. Section V outlines an abstract hybrid design process based on the reviewed analysis, design and verification approaches. Finally, Section VI draws conclusions remarking different challenges for designing dependable systems. Due to space limitations, acronyms are used throughout the work. Interested readers can refer to listed references.

II. REVIEW AND CLASSIFICATION OF DEPENDABILITY ANALYSIS TECHNIQUES

Event-based approaches analyse the failure behaviour of the system by investigating the logic succession of faults.

 ISBN: 978-1-61208-212-7

They identify an event sequence leading to equipment or function failure. Differences are mainly based on representation and analysis structures. Two of the most widely used techniques are: FT Analysis (FTA) [5] and FMEA [6].

Both techniques focus on the identification of events that jeopardize the objectives of the design. However, their logical deductive/inductive orientation (from known effects/causes towards unknown causes/effects respectively for FTA/FMEA) and initial assumptions are different. They are not orthogonal techniques, indeed they are complementary and in some cases they overlap. The extended usage of these approaches for dependability related tasks have lead to the identification of the main limitations. Subsequently, there has been a long list of works aimed at overcoming them:

- L1: FMEA and FTA are static representations of the system, neither time information nor sequence dependencies are taken into account [9].
- L2: Orientation of FTA and FMEA concentrate on the analysis of failure chain information. Consequently, their hierarchy reflects failure influences without considering system functional architecture information [10].
- L3: FMEA and FTA depend on the analyst's skill to reflect the aspects of interest. Failure modes (FM) and undesired events must be foreseen, resulting in a process highly dependent on analyst's knowledge of the system [11].
- L4: Manageability and legibility of FTA and FMEA models is hampered when analysing complex systems. Model size, lack of resources to handle interrelated failures and repeated events, in conjunction with few reusability means, are its main impediments [10] [12].

L1 refers to the capability of the technique to handle temporal notions. This is of paramount importance when analysing fault tolerant systems. L2 emphasizes the interdisciplinary work between dependability analysis and architectural design. Joining both procedures helps obtaining a design, which meets dependability requirements consistently. L3 entails a trade-off solution between the time consuming analysis resulted from understanding the failure behaviour of the system and the acquired experience. A substantial body of works have been oriented towards the automatic generation of analysis models from design models (refer to groups 3, 5 in Table I) addressing limitations L2 and L3. These approaches promote the reuse of design models showing the effects of design changes in the analysis results. However, note that the correctness of the analysis lies in the accuracy of the failure annotations. Finally, L4 underlines the capability of the model to handle the component-wise nature of embedded systems. This permits obtaining a model that better adheres to the real problem and avoids confusing results.

Many authors have developed new alternatives or extended existing ones. Three groups are identified in order to gather the approaches and limitations strategically. Firstly, L1 is addressed in the subsection *dynamic solutions for static-logic approaches*. Secondly, L2 and L4 are covered in *compositional failure analysis approaches*. Finally, specifically focusing on L3 and generally addressing the remainder of limitations *model-based transformational approaches* are studied. Note that some approaches cannot be limited to a specific group, hence they are classified accordingly to its main contribution.

A. Dynamic Solutions for Static-Logic Approaches

The limitation concerning the lack of time information has been addressed by several authors to deal with system *dynamics* such as redundancy or repair strategies.

Dugan et al. [9] paved the way to cope with configuration changing analysis using FTs by defining Dynamic Fault Tree (DFT) methodology. New gates were introduced accounting for event ordered situations like common cause failures and redundancy strategies. In [13], temporal notions and FT models were integrated in order to handle the timed behaviour of the system. The model reflects how modular FT models are switched through discrete points in time taking into account time dependant basic events.

Other alternatives to analyse DFT models are based on Monte Carlo simulations (MCS) by specifying temporal failure and repair behaviours of components through state-time diagrams [14]. In [15], an approach based on Simulink [16] for DFT modelling and reliability analysis is presented. The technique integrates MCS and FT methodologies resulting in a intuitive model-based simulating environment.

Following the way of DFTs, an approach emerged based on Reliability Block Diagrams (RBD) [2]. RBD is focused on the analysis of the success of the system, instead of the failure analysis orientation of FTs. Dynamic RBDs (DRBDs) [17] models failures and repairs of components based on their dependencies and state machines.

Lopatkin et al. [18] utilise FMEA models for system formal specifications. The approach defines generic patterns establishing direct correspondence between FMEA and state-based Event-B [19] formalism. Consideration of error detection and recovery patterns lead to analysing and verifying whether safety properties are preserved in the presence of faults. Utilization of these patterns, allows tracing from static FMEA considerations towards system *dynamics*.

Progression in the conjoint use of event and state formalisms is reflected with Boolean logic Driven Markov Processes (BDMP) [20]. BDMP employs static FT as a structure function of the system and associates Markov processes to each leaf of the tree. Similarly, State-Event Fault Tree (SEFT) [21] formalism combines elements from FT with both Statecharts and Markov chains, increasing the expressiveness of the model. SEFT deals with functional and failure behaviour, accounts for repeated states and events and

 ISBN: 978-1-61208-212-7

allows straightforward transformations of SEFT models into Deterministic and Stochastic Petri Net (DSPN) models for state-based analysis.

The compositional and transformational features of the SEFT approach, provide an adequate abstraction of the system structure and behaviour. As far as our knowledge, there is no available tool for the evaluation and transformation of SEFT models. Developing a model-based tool which extracts DSPN models from SEFT models, would create an adequate environment for constituting a sound approach for manageable and consistent dependability analyses.

B. Compositional Failure Propagation Analysis Approaches

The common factors for Compositional Failure Propagation (CFP) approaches are: the characterization of the system architectures by design components; the annotation of the failure behaviour of each of them; and the system failure analysis based on inter-components influences. Conceptually they all are very similar: the main objective of CFP approaches is to avoid unexpected consequences resulting from the failure generation, propagation and transformation of components.

Generally, CFP approaches characterise the system as component-wise developed FT-like models linked with a causality chain. System architectural specifications and subsequent dependability analyses of CFP approaches rely on a hierarchical system model. This model comprises components composed from subcomponents specifying system structure and/or behaviour. CFP approaches analyse the system failure behaviour through characterizations of individual components, which lead to achieving a manageable failure analysis procedure. Failure Propagation and Transformation Notation (FPTN) [22], Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS) [23] and Component Fault Tree (CFT) [10] are the principal CFP approaches. Their main difference is in the failure annotations of components, which specify incoming, outgoing and internal failures to each component. In order to annotate the logical combinations of these failures, FPTN uses logical equations, HiP-HOPS makes annotations using Interface Focused FMEA (IF-FMEA) tables and CFT associates to each component individual FTs. Subsequently, the connections between system components determines the *failure flow* of the system, linking related failure annotations of each component.

Concerning the different contributions of CFP approaches, FPTN first addressed the integration of system-level deductive FTA (from known effects to unknown causes) with component-level inductive FMEA (from known causes to unknown effects). HiP-HOPS integrates design and dependability analysis concepts within a hierarchical system model. However, instead of exclusively linking functional components with their failure propagations like in FPTN, first the hierarchical system model is specified and then compositional failure annotations are added to each component by means of IF-FMEA annotations. These annotations describe the failure propagation behaviour of the component in terms of outgoing failures specified as logical combinations of incoming and internal failures. Subsequently, a FT synthesis algorithm analyses the propagation of failures between connected components. Traversing the hierarchical system model, while parsing systematically the IF-FMEA annotations of its constituent components, allows the extraction of the system FT and FMEA models. CFT works in a slightly different way, it aims at linking FTs of the components with the architecture design. The component FTs can be combined and reused to systematically obtain the FT for any failure without having to create and annotate a FT for each failure.

They all have been extended to cope with occurrences of temporal events. Temporal extensions for FPTN [24] and HiP-HOPS [25] have been influenced by the DFT methodology. Focusing on non-repairable systems, the order of events is examined in order to identify specific sequence of events leading to failures. Integration of CFT concepts with state-based techniques resulted in SEFT formalism, which is able to handle availability and maintainability properties of repairable systems.

Transformation of CFP approaches into dependability analysis formalisms is an ongoing research subject (see Subsection II-C). HiP-HOPS extracts FTA and FMEA models thanks to its underlying logic and SEFT applies a translation algorithm to generate DSPN models.

Other interesting extensions include mechanisms to automate and reuse analysis concepts. Failure Propagation and Transformation Calculus (FPTC) [26] approach introduces notations to indicate nominal behaviour within FPTN models and concepts to generalise and manage FPTN equations. Moreover, an algorithm is implemented handling the general inability of CFP approaches to cope with cyclic dependencies of feedback structures. In [27], general failure logic annotation patterns were defined for HiP-HOPS. Similarly, the CFP approach presented in [28], emphasizes the reuse of failure propagation properties specified at the port level of components. These specifications centre on the physical properties of different types of flows, which allow reusing failure behaviour patterns for functional architectures.

The evolution of CFP approaches focus on reusability, automation and transformation properties. Since the annotations of components failure behaviour depend upon designers subjective considerations, reusing failure annotations leads to reducing the error proneness. Based on the fact that different dependability analyses have to be performed when designing a dependable system, definition of a unique consistent model covering all analyses would benefit these approaches. This is why recent publications in this field centre on integrating existing approaches (cf. Subsection II-C and Section IV).

 ISBN: 978-1-61208-212-7

C. Model-Based Transformational Approaches

The main aim of the transformational models is to construct dependability analysis models (semi-)automatically. The process starts from a compositional design description using computer science modelling techniques. The failure behaviour is specified either by extending explicitly the design model or developing a separate model, which is allocated to the design model. Transformation rules and algorithms extract dependability analysis models from it.

These approaches lead to adopting a trade-off decision between dependability design and analysis processes. On one hand, the automation and reuse of analysis techniques in a manageable way makes it a worthwhile approach for design purposes. The impact of design changes on dependability attributes are straightforwardly analysed. On the other hand, from purist's point of view of classical analysis techniques, the automation process removes the ability of these techniques to identify and analyse hazards or malfunctions in a comprehensive and structured way.

Architectural description languages (ADLs) provide an adequate abstraction to overcome the limitations. Simulink [16], AADL [29] and UML [30] have been used for both architectural specification and failure behaviour specification. UML is a widely used modelling language, which has been extended for dependability analyses following model driven architecture (MDA) concepts. Namely, profiles [31] allow extending and customizing modelling mechanisms to the dependability domain.

Lately, a wide variety of independently developed extensions and profiles have come up for dependability analysis [32]. However, some generally applicable metamodel is lacking. In an effort to provide a consistent profile CHESS ML [33] emerged. Its high-level specifications are transformed into Intermediate Dependability Model (IDM) in order to facilitate transformations. CHESS ML development is currently ongoing and seems to provide all necessary mechanisms to model dependable systems and extract either event-based (FMECA, FPTC) or state-based (SPN) analysis models.

CFP approaches have been shifted towards the transformational paradigm. The toolset for FPTC approach [26] relies on a generic metamodel in order to support transformations from SysML and AADL models. In [34], a metamodel is developed so as to extract CFT models from functional architecture models specified in UML. This process permits the generation of reusable CFT models consistent with the design model. In the same line, integration of HiP-HOPS model with EAST-ADL2 automotive UML profile is presented in [35]. Translations from high-level ADLs to well established CFP analysis techniques, enable an early dependability analysis and allow undertaking timely design decisions.

Architecture Analysis and Design Language (AADL) captures the system architectural model in terms of components and their interactions describing functional, mapping and timing properties among others. The core language can be extended to meet specific requirements with annex libraries. Behaviour and error model annexes are provided with the tool. The error annex links system architecture components to their failure behaviour specification making possible the analysis of the dependability attributes of the system. It has been used for both state-based [36] and event-based [37] analysis.

AltaRica [38] is a dependability language, which enables describing the behaviour of systems when faults occur. The model is composed of several components linked together representing an automaton of all possible behaviour scenarios, including those cases when reconfigurations occur due to the occurrence of a failure [39]. It is possible to process such models by other tools for MC or generation of FTs [40]. Transformations from AADL to AltaRica are presented in [41], based on MDA concepts so as to perform dependability analyses and avoid inconsistencies while working with different formalisms.

In [42], a method for RAMS analysis is defined centred on SysML [43] modelling language from where a FMEA model is deduced. SysML diagrams define a functional model connected to a dysfunctional database enabling the identification of FMs. This database contains the link between system architecture and failure behaviour giving the key for FMEA extraction. Further, the methodology for dependability assessment is extended using AltaRica, AADL and Simulink models. They address reliability and timing analysis and simulation of the effects of faults respectively.

Definition of a model for the extraction of all necessary formalisms for dependability analysis is the common goal for the aforementioned works. Interconnections between different formalisms in order to take advantage of the strengths of each ADL, allow analysing dependability properties accurately. AltaRica and AADL cover adequately the analysis of reliability, availability and maintainability attributes. Extraction of the main CFP approaches from ADLs should help to analyse comprehensively system safety properties. Moreover, Simulink model simulations allow evaluating the effects of failure and repair events in the system. Thereby, integrations between language specific models like in [42] helps evaluating accurately all dependability aspects of a system.

D. Classification of Techniques

In order to classify the covered approaches, Table I groups them taking into account limitations specified in Section II.

Approaches gathered within the group 5 contain all necessary features in order to analyse dynamic systems consistently and in a manageable way. Compositional failure annotation, dynamic behaviour and automatic extraction of analysis models are the key features addressed by these

 ISBN: 978-1-61208-212-7

Table I
SUMMARY OF LIMITATIONS OVERCOME BY APPROACHES

Group	Approach	Limitations
1	[9] [13] [14] [20]	L1
2	[10] [22] [26]	L2, L4
3	[23] [28] [37]	L2, L3, L4
4	[17] [24] [39]	L1, L2, L4
5	[21] [18] [25] [33] [36] [42]	L1, L2, L3, L4

approaches. Utilization of failure annotation patterns promote flexibility and reuse and consequently, reduce the error proneness. Nevertheless, as noted in [44], characterization of the failure behaviour of components depends on the component context, which conditions compositional and reuse properties. Moreover, automatic generation of the analysis model does not completely alleviate the dependency on the knowledge of the analyst. However, it lets managing and specifying the failure behaviour in a clear and consistent way.

III. DEPENDABLE DESIGN: TRADE-OFF BETWEEN DEPENDABILITY AND COST

Generally, dependability design decisions and objectives are related to trade-off decisions between system dependability attributes and cost. Dependability requirements often conflict with one another e.g., safety-availability compromise when a faults leads the system to a safe shutdown in order to prevent it from propagating. The time at which design decisions are taken, determines the cost that the design process can incur.

Designing a dependable system within considered requirements requires a process to match and tune combination of architectural components so as to find an optimal solution satisfying design constraints. For the sake of analysing the applicability of the aforementioned analysis techniques, three demonstrative works are chosen. Their underlying structure is illustrated in Figure 1.

Dependable design methodologies are proposed by Benard et al. [45] and Clarhaut et al. [46]. The former focuses on quantification and comparison of RAMS properties of alternative components. The latter overcomes static-logic limitations by integrating temporal functions. Their design methodology is based on the *operational model*, which aims at mapping the *functional model* onto a *compatible physical model* (cf. Figure 1).

The functional model is developed in a top-down hierarchical manner tracing from system level functions up to lower level functions. At the lowest level, physical components are linked with corresponding functions. Dependability considerations lead to characterizing hardware components through failure modes (*FM*) and *redundancy structures*. The way in which *dependability analysis* is performed differ both approaches. While the former uses MCS to analyse the

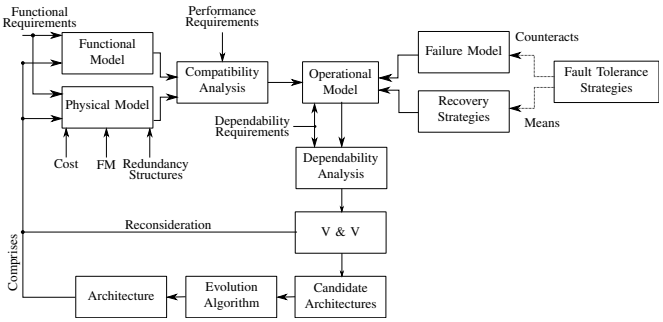

Figure 1. Abstract Design Process (Adapted from [45])

impact of allocations of functions into physical components; the latter focuses on identifying component-wise temporal failure contributions to the system-level undesired event.

In [47], HiP-HOPS approach is extended with *recovery strategies*. These capabilities are formally represented using patterns. They characterize the potential to detect, mitigate and block affecting component failures identified in the *failure model*. Dependability analysis is performed by means of FTA and FMEA.

Trade-off analysis between dependability and cost determines optimal *architectures*. In [47], fault tolerant configurations are introduced without violating user constraints and an *evolutionary optimization algorithm* is used to converge through dependability and cost requirements. Similarly [46] identifies set of optimal *candidate architectures* by minimizing failure contributions and cost of necessary components to accomplish system functions.

IV. DEPENDABLE DESIGN VERIFICATION: FAULT INJECTION APPROACHES

Fault Injection (FI) techniques focus on evaluating system behaviour in the presence of faults according to target dependability properties. The outcome of this process may lead to considering design changes. However, changes adopted late in the design process are costly. This is why we focus on FI approaches adopted at the preliminary design phase. This process is based on the analyst's knowledge to reason about the functional and failure behaviour of the system. As a result, the effectiveness of fault detection, isolation, recovery and reconfiguration strategies are evaluated. Timely evaluation of these properties provides a valuable feedback for design purposes. However, difficulties arise from the accuracy of the system behaviour, which requires an accurate knowledge of the system.

Instead of focusing on purely verification oriented FI approaches, we address *integrative verification* approaches. These works result from the integration of design, analysis and verification tasks. Covered approaches aim at combining dependability analysis techniques examined within the group 5 (cf. Table I) with FI approaches. They express system

 ISBN: 978-1-61208-212-7

behaviour using a compositional model, which gathers nominal, failure and recovery behaviours. Integrating approaches using model transformations, allows using a single design model for dependability and verification analyses.

The system *design model* takes into account functional and failure behaviour of components. Temporal logic languages are used to define system *requirements*. They describe how the truth values of assertions change over time, either qualitatively (Computation Tree Logic (CTL), Linear Time Logic (LTL)) or quantitatively (Continuous Stochastic Logic (CSL), probabilistic CTL (PCTL)). Model-checking (MC) engine assesses whether such requirements are met or not by the design model, using a *analysis model*. To do so, it is necessary to transform the design model into the analysis model. When the analysis model fails to meet these requirements, its effects are deduced automatically identifying the paths that violate the conditions (counter-examples (CEs)) [7]. The logical orientation of this analysis process results in FMEA-like cause-effect analysis.

There are some limitations hampering the analysis and interpretation of these approaches. Representation structures of the results, state-explosion problems, technical specification difficulties, qualitative nature of MC analysis and model inconsistencies are some challenges to be addressed.

The COMPASS project [48] addresses these limitations based on SLIM (System-Level Integrated Modeling) language [49]. The semantics of SLIM cover the nominal and error behaviour of AADL. The complete specification of SLIM consists of a nominal model, a failure model and a description of the effects of failures in the nominal model (*extended model*). Due to its underlying formal semantics, various types of analyses are possible: validation of functional, failure, and extended models via simulation and MC; dependability analysis and performance evaluation; diagnosability analysis; and evaluation of the effectiveness of fault detection, isolation and recovery strategies.

Similarly, Güdemann and Ortmeier [50] proposed an intermediate (IM) tool-independent model called Safety Analysis Modelling Language (SAML). SAML describes a finite state automata, which is used to characterise the extended system model. This model specifies the nominal behaviour, failure occurrences, its effects and the physical behaviour of the surrounding environment. From this single model, quantitative and qualitative MC analyses are performed. The former identifies minimal critical sets using CEs to indicate time-ordered combinations of failures causing the system hazard. The latter calculates per-demand and per-time failure probabilities.

TOPCASED project [51] aims at developing critical embedded systems including hardware and software. TOPCASED integrates ADLs and formal methods. The approach transforms high-level ADL models (SysML, UML and AADL) into an IM model specified in Fiacre language [52]. Fiacre specifies behavioural and timing aspects of high-level models making use of timed Petri nets primitives. Subsequent transformations of the IM model into MC tools (TINA and CADP), make possible the formal verification and simulation of the specified requirements. TINA [53] analyse requirements specified in the state variant of LTL proposition logic (State/Event LTL (SELTL)) focusing on timeliness properties. CADP [54] transforms Fiacre models into LOTOS programs, which are handled by its underlying tools for validation via MC and simulation.

Albeit these approaches provide a means to extract classical dependability models from high-level models, none of them focus on integrating existing CFP approaches. There are some incipient works linking CFP and verification approaches. They are influenced by HiP-HOPS [55] and FPTC [56]. Both approaches address the integration of qualitative design models with quantitative analysis via probabilistic MC. These approaches in particular and CFPs in general, provide useful resources when characterizing the failure behaviour of systems. The pros and cons of the covered works are summarized in the Table II.

The addressed works integrate well known tools and formalisms. However, integration of analysis and verification approaches when designing a dependable system is an ongoing research subject. There is an increasing interest in reusing and generalizing CFP approaches (e.g., transformation of CFP approaches into metamodels [26] [34] [35] and integration of CFP and verification approaches [55] [56]).

V. Hybrid Design Process

The goal of this section is not to provide a new design approach. Our aim is to make use of the reviewed analysis, design and verification approaches so as to outline a consistent and reusable model-based design process. This process emerges from the structure of the integrative verification approaches (cf. Section IV).

The separation of dependability analysis and verification tasks may lead to hampering the system design since results identified from either task need to be reconsidered during the design process (cf. Section III). On one hand, dependability analyses characterized by transformational approaches (cf. Section II-C), allow tracing from design considerations towards dependability analysis models. These approaches evaluate the dynamic system behaviour, as well as the effect of particular component failure occurrences at system level. On the other hand, purely verification oriented approaches mainly focus on the verification of the adequacy of the design model with respect to RAMS requirements. This is why we centre on covering integrative verification approaches.

When matching and tuning design components so as to find optimal design solutions satisfying design constraints, possible inconsistencies may arise due to the independent considerations of these approaches. This is why we should focus on outlining a model-based hybrid design process, which unifies design, analysis and verification tasks. This

 ISBN: 978-1-61208-212-7

Table II
SUMMARY OF FAULT INJECTION APPROACHES

Works	*Design Model*	*Analysis Model (*Auto)*	*Reqs. (*Auto)*	*Results*	*Specific Features*	*Future works*
[49]	SLIM	NuSMV*, MRMC*	CTL*, LTL*, CSL*	DFT, FMEA, Prob. Calc.	Req. patterns; Integrated verification and dependability and performance analyses of extended models.	Manual extension of the nominal model; Redundant FTA-FMEA results; State-explosion.
[50]	SAML	NuSMV*, PRISM*, MRMC*	CTL, PCTL	Time-Ord. CE, Prob. Calc.	Combination of qualitative and quantitative analyses on the same model.	Manual extension of the nominal model; Transf. ADLs (Simulink, Scade) into SAML; Req. patterns.
[52]	Fiacre	TINA, CADP	LTL, SELTL	Timing, Prob. Calc.	Req. patterns; Integrated design, analysis and verification approaches.	State Explosion; Back annotation of results.
[55]	Simulink	PRISM*	CSL	Prob. Calc.	Systematic generation of analysis models from CFP design models.	No CE; State-explosion; Dynamic behaviour.
[56]	FPTC	PRISM	CSL	CE	Integration of CFP approach and prob. model checking.	Fail Prone Manual transformation; Translate CE to design model.

process relies on initial system requirements, models, transformations and reuse of designer's considerations and results extracted from analysis and verification tasks (cf. Figure 2).

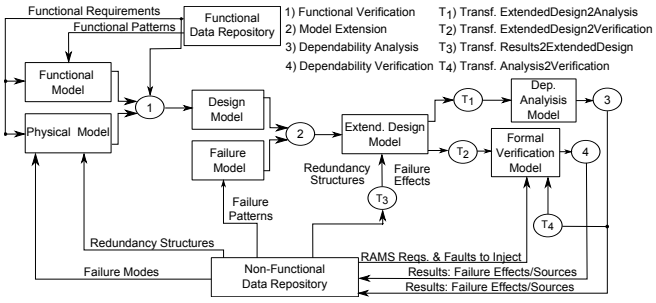

Figure 2. Hybrid Design Process

This design process starts from initial functional and physical considerations. *Functional verification* analysis evaluates the adequacy of the allocation of the *functional model* into the *physical model* according to functional requirements. The outcome of this process allows considering the verified *design model* (operational model, cf. Figure 1). Subsequently, this model is extended with the *failure model* accounting for failure occurrences of the considered model. Failure patterns aid in the construction of the failure model allowing the reuse non-functional considerations. Further, the effects of the considered failures and recovery strategies are annotated in the *extended design model* in order to counteract failure occurrences and its effects. With the aim to carry out *dependability analysis* and *formal verification* evaluations of the extended design model, twofold transformations need to be performed. The means to perform these transformations have been presented in Subsection II-C and Section IV respectively. Transformations of these models make the evaluation of the adequacy of the extended design model respect to RAMS requirements possible. Dependability analysis and verification tasks enable finding further failure effects and failure sources (apart from occurrence probabilities) either by CEs or dependability specific models. These results need to be transformed in order to reconsider for design and analysis purposes. For the sake of reusing and refining the design process, data repositories have been considered consisting of annotation patterns for requirements and models (both functional and non-functional) and reusable recovery strategies.

On one hand, the outlined design approach enables benefiting from consistent design considerations. Moreover, data repositories allow the reuse of designer's considerations as well as analysis results. Furthermore, user-friendly means make the annotation processes more evident. On the other hand, the automation of the extraction of dependability models hides information about the failure behaviour. Additionally, the flexibility of the approach depends on the system context, which would determine the reusability of functional and non-functional considerations.

VI. CONCLUSION AND FUTURE WORK

Designing a dependable system, poses a wide variety of challenges on all its phases. This paper groups different approaches in order to identify and classify them.

The listed limitations guided the evolution of the analysis techniques towards Compositional Failure Propagation (CFP) and transformational approaches. Automatic extraction of analysis models from design models is an ongoing research field, which leads to achieving consistency between design and analysis models.

However, this is not the cure-all remedy, which alleviates analysts from identifying and analysing failure behaviours, but helps obtaining a manageable analysis compared to the difficult and laborious traditional process. User friendly resources, such as design components or failure annotation libraries, enable the reuse of nominal and failure models.

When designing a new system, special care should be taken, since reuse properties depend on the system context.

 ISBN: 978-1-61208-212-7

The reuse of failure annotations in the design process, eases the architectural iterative refinement process. This makes possible the analysis of different implementations using the same component failure models.

For verification purposes, fault injection (FI) approaches have been studied. Since the adoption of FI approaches is made late in the traditional design process, we have considered integrative works. Their main objective is to address consistently dependability analysis, design and verification tasks at the preliminary design phase. An early integration of these tasks would add value to the dependable design process. There are many challenging tasks to address when constructing an end-to-end dependable design methodology. Integration of the CFP approaches within this methodology or validation of the correctness of the faults to be injected are some of the subjects to be addressed.

Therefore, we hypothesize that instead of developing independent approaches to identify, analyse and verify dependability requirements, future directions will focus on integrating different approaches. This process requires tracing verification results to the initial dependable design model and vice versa. Consequently, accounting for these considerations, we have sketched an abstract integrative design process. The integration of the approaches should allow undertaking timely design decisions by reducing costs and manual failure-prone annotations. Additionally, it will alleviate the need to clutter a model with redundant information. In this field, challenging work remains to do sharing information between existing approaches so as to take advantage of complementary strengths of different approaches.

REFERENCES

[1] A. Avizienis, J.-C. Laprie, B. Randell, and C. Landwehr, "Basic concepts and taxonomy of dependable and secure computing," *IEEE Trans. Dependable Secur. Comput.*, vol. 1, pp. 11–33, January 2004.

[2] M. Rausand and A. Høyland, *System Reliability Theory: Models, Statistical Methods and Applications Second Edition*. Wiley-Interscience, 2003.

[3] A. Arora and S. Kulkarni, "Component based design of multitolerant systems," *IEEE Trans. on Sw. Eng.*, vol. 24, no. 1, pp. 63–78, 1998.

[4] M. Hiller, A. Jhumka, and N. Suri, "An approach for analysing the propagation of data errors in software," in *Proc. of DSN'01*, 2001, pp. 161–170.

[5] W. Vesely, J. Dugan, J. Fragola, Minarick, and J. Railsback, "Fault Tree Handbook with Aerospace Applications," NASA, Handbook, 2002.

[6] US Department of Defense, *Procedures for Performing, a Failure Mode, Effects, and Criticality Analysis (MIL-STD-1629A)*. Whasington, DC, 1980.

[7] C. Baier and J.-P. Katoen, *Principles of model checking*. MIT Press, 2008.

[8] Y. Prokhorova, L. Laibinis, E. Troubitsyna, K. Varpaaniemi, and T. Latvala, "Derivation and formal verification of a mode logic for layered control systems," in *Proc. of APSEC'11*, 2011, pp. 49–56.

[9] J. Dugan, S. Bavuso, and M. Boyd, "Dynamic fault-tree models for fault-tolerant computer systems," *IEEE Trans. on Reliability*, vol. 41, no. 3, pp. 363–377, 1992.

[10] B. Kaiser, P. Liggesmeyer, and O. Mäckel, "A new component concept for fault trees," in *Proc. of SCS'03*, 2003, pp. 37–46.

[11] A. Galloway, J. McDermid, J. Murdoch, and D. Pumfrey, "Automation of system safety analysis: Possibilities and pitfalls," in *Proc. of ISSC'02*, 2002.

[12] C. Price and N. Taylor, "Automated multiple failure FMEA," *Reliability Eng. & System Safety*, vol. 76, pp. 1–10, 2002.

[13] M. Ĉepin and B. Mavko, "A dynamic fault tree," *Reliability Eng. & System Safety*, vol. 75, no. 1, pp. 83–91, 2002.

[14] Rao, K. Durga, V. Gopika, V. V. S. Sanyasi Rao, H. S. Kushwaha, A. K. Verma, and A. Srividya, "Dynamic fault tree analysis using Monte Carlo simulation in probabilistic safety assessment," *Reliability Eng. and System Safety*, vol. 94, no. 4, pp. 872–883, 2009.

[15] G. Manno, F. Chiacchio, L. Compagno, D. D'Urso, and N. Trapani, "MatCarloRe: An integrated FT and Monte Carlo Simulink tool for the reliability assessment of dynamic fault tree," *Expert Systems with Applications*, vol. 39, no. 12, pp. 10 334–10 342, 2012.

[16] "MathWorks," http://www.mathworks.com; Last access: 2012/06/13.

[17] S. Distefano and A. Puliafito, "Dynamic reliability block diagrams vs dynamic fault trees," *In Proc. of RAMS'07*, vol. 8, pp. 71–76, 2007.

[18] I. Lopatkin, A. Iliasov, A. Romanovsky, Y. Prokhorova, and E. Troubitsyna, "Patterns for representing FMEA in formal specification of control systems," in *Proc. HASE'11*, 2011, pp. 146–151.

[19] "Event-B and the Rodin platform," http://www.event-b.org; Last access: 2012/06/13.

[20] M. Bouissou, "A generalization of Dynamic Fault Trees through Boolean logic Driven Markov Processes (BDMP)," in *Proc. of ESREL'07*, vol. 2, 2007, pp. 1051–1058.

[21] B. Kaiser, C. Gramlich, and M. Forster, "State/event fault trees a safety analysis model for software-controlled systems," *Reliability Eng. System Safety*, vol. 92, no. 11, pp. 1521–1537, 2007.

[22] P. Fenelon and J. A. McDermid, "An integrated tool set for software safety analysis," *J. Syst. Softw.*, vol. 21, pp. 279–290, 1993.

[23] Y. Papadopoulos, M. Walker, D. Parker, E. Rüde, R. Hamann, A. Uhlig, U. Grätz, and R. Lien, "Engineering failure analysis and design optimisation with HiP-HOPS," *Engineering Failure Analysis*, vol. 18, no. 2, pp. 590–608, 2011.

 ISBN: 978-1-61208-212-7

[24] R. Niu, T. Tang, O. Lisagor, and J. McDermid, "Automatic safety analysis of networked control system based on failure propagation model," in *Proc. of ICVES'11*, 2011, pp. 53–58.

[25] M. Walker and Y. Papadopoulos, "Qualitative temporal analysis: Towards a full implementation of the fault tree handbook," *Control Eng. Practice*, vol. 17, no. 10, pp. 1115–1125, 2009.

[26] R. Paige, L. Rose, X. Ge, D. Kolovos, and P. Brooke, "FPTC: Automated safety analysis for Domain-Specific languages," in *MoDELS Workshops '08*, vol. 5421, 2008, pp. 229–242.

[27] I. Wolforth, M. Walker, L. Grunske, and Y. Papadopoulos, "Generalizable safety annotations for specification of failure patterns," *Softw. Pract. Exper.*, vol. 40, pp. 453–483, 2010.

[28] C. Priesterjahn, C. Sondermann-Wölke, M. Tichy, and C. Hölscher, "Component-based hazard analysis for mechatronic systems," in *Proc. of ISORCW'11*, 2011, pp. 80–87.

[29] P. Feiler and A. Rugina, "Dependability Modeling with the Architecture Analysis & Design Language (AADL)," Technical Note CMU/SEI-2007-TN-043, CMU Software Engineering Institute, 2007.

[30] "The Unified Modeling Language," http://www.uml.org/; Last access: 2012/06/13.

[31] L. Fuentes-Fernández and A. Vallecillo-Moreno, "An Introduction to UML Profiles," *UPGRADE*, vol. 5, no. 2, pp. 5–13, 2004.

[32] S. Bernardi, J. Merseguer, and D. Petriu, "Dependability modeling and analysis of software systems specified with UML," *ACM Computing Survey*, In Press.

[33] L. Montecchi, P. Lollini, and A. Bondavalli, "An intermediate dependability model for state-based dependability analysis," University of Florence, Dip. Sistemi Informatica, RCL group, Tech. Rep., 2011.

[34] R. Adler, D. Domis, K. Höfig, S. Kemmann, T. Kuhn, J. Schwinn, and M. Trapp, "Integration of component fault trees into the UML," in *MoDELS'10*, 2010, pp. 312–327.

[35] M. Biehl, C. DeJiu, and M. Törngren, "Integrating safety analysis into the model-based development toolchain of automotive embedded systems," in *Proc. of LCTES '10*. ACM, 2010, pp. 125–132.

[36] A. Rugina, K. Kanoun, and M. Kaâniche, "A system dependability modeling framework using AADL and GSPNs," in *Architecting dependable systems IV, LNCS*, vol. 4615. Springer, 2007, pp. 14–38.

[37] A. Joshi, S. Vestal, and P. Binns, "Automatic Generation of Static Fault Trees from AADL models," in *DNS Workshop on Architecting Dependable Systems*. Springer, 2007.

[38] A. Arnold, G. Point, A. Griffault, and A. Rauzy, "The AltaRica formalism for describing concurrent systems," *Fundamenta Informaticae*, vol. 40, no. 2-3, pp. 109–124, 1999.

[39] B. Romain, J.-J. Aubert, P. Bieber, C. Merlini, and S. Metge, "Experiments in model based safety analysis: Flight controls," in *DCDS'07*, 2007, pp. 43–48.

[40] P. Bieber, C. Castel, and C. Seguin, "Combination of fault tree analysis and model checking for safety assessment of complex system," in *Proc. of EDCC'02*, vol. 2485. Springer, 2002, pp. 624–628.

[41] K. Mokos, P. Katsaros, N. Bassiliades, V. Vassiliadis, and M. Perrotin, "Towards compositional safety analysis via semantic representation of component failure behaviour," in *Proc. of JCKBSE'08*. IOS Press, 2008, pp. 405–414.

[42] R. Cressent, V. Idasiak, F. Kratz, and P. David, "Mastering safety and reliability in a Model Based process," in *Proc. of RAMS'11*, 2011.

[43] "OMG Systems Modelling Language," http://www.omgsysml.org/; Last access: 2012/06/13.

[44] O. Lisagor, "Failure logic modelling: A pragmatic approach," Ph.D. dissertation, Department of Computer Science, The University of York, 2010.

[45] V. Benard, L. Cauffriez, and D. Renaux, "The Safe-SADT method for aiding designers to choose and improve dependable architectures for complex automated systems," *Reliability Eng. & System Safety*, vol. 93, no. 2, pp. 179–196, 2008.

[46] J. Clarhaut, S. Hayat, B. Conrard, and V. Cocquempot, "Optimal design of dependable control system architectures using temporal sequences of failures," *Ieee Transactions On Reliability*, vol. 58, no. 3, pp. 511–522, 2009.

[47] M. Adachi, Y. Papadopoulos, S. Sharvia, D. Parker, and T. Tohdo, "An approach to optimization of fault tolerant architectures using hip-hops," *Softw. Pract. Exp.*, 2011.

[48] "Correctness, Modelling and Performance of Aerospace Systems," http://compass.informatik.rwth-aachen.de; Last access: 2012/06/13.

[49] M. Bozzano, A. Cimatti, J.-P. Katoen, V. Nguyen, T. Noll, and M. Roveri, "Safety, dependability and performance analysis of extended aadl models," *Computer Journal*, vol. 54, no. 5, pp. 754–775, 2011.

[50] M. Güdemann and F. Ortmeier, "Towards model-driven safety analysis," in *Proc. of DCDS 11*, 2011, pp. 53 – 58.

[51] "The Open-Source Toolkit for Critical Systems," http://www.topcased.org; Last access: 2012/06/13.

[52] B. Berthomieu, J.-P. Bodeveix, P. Farail, M. Filali, H. Garavel, P. Gaufillet, F. Lang, and F. Vernadat, "Fiacre: an intermediate language for model verification in the topcased environment," in *ERTS'08*, 2008.

[53] "TINA," http://projects.laas.fr/tina; Last access: 2012/06/13.

[54] "CADP," http://http://www.inrialpes.fr/vasy/cadp/; Last access: 2012/06/13.

[55] A. Gomes, A. Mota, A. Sampaio, F. Ferri, and J. Buzzi, "Systematic model-based safety assessment via probabilistic model checking," in *ISoLA'10*. Springer, 2010, pp. 625–639.

[56] X. Ge, R. Paige, and J. McDermid, "Probabilistic failure propagation and transformation analysis," in *SAFECOMP'09*, 2009, vol. 5775, pp. 215–228.

 ISBN: 978-1-61208-212-7