Model-Based Safety Assessment

Review of the Discipline and its Challenges

Oleg Lisagor, Tim Kelly
Department of Computer Science
The University of York
United Kingdom
{oleg.lisagor, tim.kelly}@cs.york.ac.uk

Ru Niu
State Key Laboratory of Rail Traffic Control & Safety
Beijing Jiaotong University
Beijing, China
rniu@bjtu.edu.cn

Abstract— **Since its emergence in 1990s, Model-Based Safety Assessment (MBSA) has enjoyed significant interest from both academia and industry. The last decade has seen not only the development of a number of methods, techniques and tools, but also the gradual adoption of MBSA techniques by industry and its acceptance by regulators. However, the field of MBSA encompasses a large number of fundamentally dissimilar techniques. This paper presents a simple classification schema for MBSA techniques based on two criteria – provenance of the model and engineering semantics of component dependencies captured by the model. The classification organizes the existing techniques into a number of coherent families. Applicability, limitations and challenges of most prominent families of MBSA techniques are presented, and some of the common challenges faced by MBSA discipline are discussed.**

Keywords- System Safety Engineering, Safety Assessment Methodology, Model-Based Safety Assessment.

I. Introduction

Since its emergence in 1990s, Model-Based Safety Assessment (MBSA) has enjoyed significant interest from both academia and industry. In the last decade MBSA methods have been gradually adopted by the industry and increasingly accepted by the Regulators (especially in the aviation sector). For example, Flight Control System of Dassault 7x aircraft has been certified on the basis of models specified in a dataflow dialect of AltaRica Language [2]; the forthcoming revision of ARP4761 ("Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment") document – a de-facto safety engineering standard of the aviation industry – is likely to include explicit provisions for the use of MBSA.

Original MBSA techniques, such as Failure Propagation and Transformation Notation (FPTN) [14], Hierarchically Performed Hazard Origin and Propagation Studies (HiP-HOPS) [23] and AltaRica language [3] have sought to unify 'classical' safety analysis methods such as Fault Tree Analysis (FTA) and Failure Modes and Effects Analysis (FMEA) and to provide a formalism for capturing a single authoritative 'safety model' of the system. Classical analysis artifacts (fault trees, minimal cut sets or FMEA tables) could be automatically extracted from such a core model for any condition of interest (such as a system-level failure condition).

Supported by the Key State Laboratory for Rail Traffic Control and Safety (contract number RCS2010K003), Beijing Jiaotong University and by the European Commission MISSA project (contract number ACP7-GA-2008-212088)

In addition to the unification and at least partial automation objectives above, virtually all MBSA techniques seek tighter integration between safety assessment and design artifacts (models). However the exact approach to such integration ranges from improving traceability by introducing the notion of components to performing safety analysis on the basis of design models themselves.

Finally, many MBSA methods have declared objectives of compositional and reusable safety assessment. Some have also sought to expand on the combinatorial nature of the FTA and to provide expressive power that is seen as more suitable for complex and highly-reconfigurable safety critical systems.

Many of the MBSA methods share little technical characteristics apart from the abstract objectives above. The nature of the information captured in the safety assessment models and the process of their construction differs significantly. Challenges posed by different families of techniques often remain hidden and unaddressed under the umbrella term of "model-based safety assessment". For example, we have shown previously that for some MBSA methods compositionality and reusability objectives are fundamentally unattainable and argued that for others the achievement comes at a cost of reducing the scope of the assessment and/or introduction of a "common point of failure" between design and safety assessment processes. [19].

In the remainder of this paper we organize MBSA methods into coherent and internally consistent families that share key technical features, strengths, limitations and challenges. We use two criteria for such classification – model provenance and engineering semantics of component interfaces – described in sections 2 and 3 respectively. Section 4 presents classification of some of the MBSA techniques according the two criteria and Section 5 discusses the applicability and future challenges of different families of methods. We conclude with the summary and some remarks on challenges faced by the MBSA research discipline as a whole.

II. Model Provenance

The first criteria that distinguishes MBSA methods is concerned with the process of definition of the safety models and its relationship with the system design process. There are currently two approaches to model construction:

a) Safety assessment models can be defined through *extension* of the models used in the system development process,

b) Safety assessment can be performed on the basis of *dedicated models* defined by safety engineers and obtained through ‘manual’ assessment of the system.

An example of the first approach is Extended System Model (ESM) / Failure Injection (FI) methods developed by ESACS and ISAAC project over the past 10 years [2, 9]. Under this approach safety engineers receive design models of the system, specified in languages such as SCADE or Matlab Simulink, from the development process. The model is then extended with Failure Mode (FM) models – simple model components ‘injected’ into the flows of original model. Each FM component (Fig.1) typically has two inputs and an output. One of the inputs and the output are used to insert the FM into a flow; their types are therefore the same as the type of that of the original flow. The remaining input is Boolean as is called ‘activation’ of the FM. The component itself defines the effect of the Failure Mode – such as value being stuck at zero or an inversion of a Boolean value. When activation input is set to true this deviation is applied to the flow, whereas inactive FM components simply propagate original inputs to outputs, thus, having no effect on the behavior (Fig. 1).

ESACS and ISAAC projects have developed a comprehensive library of failure modes and adapted modeling tools such as SCADE to provide graphical user interface to support the model extension process and definition of bespoke failure modes. The platform also ‘hides’ implementation of the failure modes (i.e. FM components) from the user, thus, avoiding excessive cluttering of the model and ensuring that, even when extended, the model can still be used in the development process. Finally, tools were also developed tools to enable analysis of the extended system models with respect to a system-level failure conditions specified over model variables. Whilst often based on model checking technology, the analysis performed by these tools is logically equivalent to an exhaustive search through activation of all possible permutations of failure modes. Permutations that lead to the satisfaction of the failure condition expression are reported as Minimal Cut Sets (MCSs).

The key advantage of the model extension approach is consistency, by construction, of the safety analyses and the ‘real’ design model of the system. Furthermore, development and safety processes are capable of sharing a common modeling environment, languages and tools.

However, such ‘unquestioned’ utilization of the design model may introduce new risks into the safety assessment process. As any models, system design models are “abstractions defined with an intended goal in mind” [30]; however, safety assessment is clearly not an intended goal when design models are defined. Consequently, the use of these models may impose undue constraints on the safety assessment leading to incomplete analysis results with respect to the real-world behavior of the system. For example, failure of components may not only lead to the deviations of outputs, but may also result in new, unintentional, interactions between the components and, even possibly establish the unintended dependencies paths between apparently unconnected components. Short circuits of electrical power distribution systems are an example of such behavior. Failure injection approach is typically incapable of adequately uncovering these scenarios.

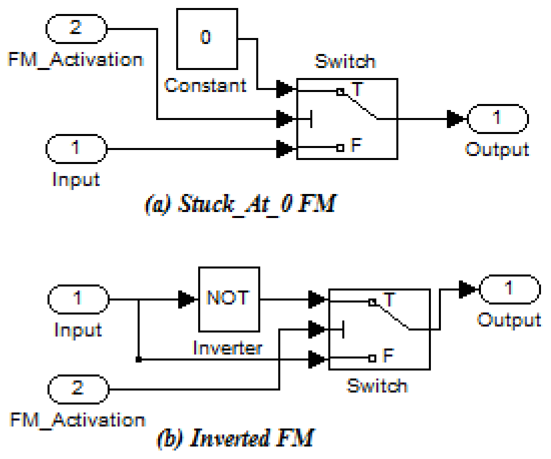

Figure 1 - Two Simple Failure Mode Components in Matlab Simulink

Furthermore, assuming adequacy of the design models, injection of an incomplete set of failure modes into the model will potentially lead to incomplete safety analysis. Completeness of the injection, however, is notoriously difficult to guarantee since each failure mode must be defined in detail and concrete terms. A “signal is provided with the delay” failure mode, that can be often found in classical safety analyses, for example, would have to be modeled differently at least for constant delay (offset) and for every variable delay profile (e.g. continuously increasing, continuously decreasing, wave-form).

Finally, the approach does not permit model abstraction from the level of detail set by system developers. This means that the computational complexity of complete extended system models of modern systems is often intractable with current analysis tools.

The second approach to MBSA, based on dedicated ‘stand-alone’ safety assessment models, alleviates many of the above problems. As models are created specifically for the goal of safety assessment, engineers can adjust the richness of component interfaces and the overall level of detail, thus, avoiding unnecessary complexity whilst ensuring that the models are adequate and fit-for-purpose. This comes at the cost of losing the provable validity of the safety analysis results with respect to design, and replacing consistency by construction with some form of traceability between models in design and safety domains. The latter is typically achieved through the notion of component (or module) and hierarchical organization of the safety model architecture that reflects structure and hierarchical organization of the system (and its design models). Most of the pioneering MBSA techniques and notations – such as HiP-HOPS [23], FPTN [13] and AltaRica [3] – follow this approach. In such models component behavior is characterized primarily as dependency of the outputs on components inputs and internal malfunctions. (Note: the engineering semantics of such input and outputs is considered in the following section.) In an attempt to provide expressive power capable of modeling dynamic and reconfigurable systems some of the MBSA techniques have introduced the notion of the state into component characterizations. For example a characterization of a shut-off hydraulic valve can be written as following:

- The valve is either closed or open and, orthogonally, either failed or operational;

- The valve may become failed as a result of an internal malfunction (failure);
- If the valve is operational and electronic control signal is received the valve moves to an appropriate state (closed or open); if the valve has failed then further control signals are inconsequential;
- If the valve is open its hydraulic output (which models provision of hydraulic pressure) takes the value of the hydraulic input; otherwise – the output is false (no pressure).

Such a characterization can be trivially translated into an AltaRica Dataflow [7, 27] node specification (Fig. 2).

```
node Valve
 flow
  Ctrl : {Open,Close} : in ;
  HydIn : bool : in ;
  HydOut : bool : out ;
 state
  FailureState : {Operational,Failed} ;
  FunctionalSt : {Closed,Open} ;
 event
  Malfunction, Update ;
 init
  FailureState := Operational ;
  FunctionalSt := Closed ;
 trans
  FailSt = Operational |- Malfunction -> FailState := Failed;
  FailSt = Operational and FunctSt = Closed and Ctrl = Open
    |- Update -> FunctSt := Open;
  FailSt = Operational and FunctSt = Open and Ctrl = Close
    |- Update -> FunctSt := Closed;
 assert
  (if FunctSt = Open
    then HydOut = HydIn
    else HydOut = false)
edon
```

Figure 2 - AltaRica Characterization of the Valve

Individual characterizations are then composed through interfaces to produce an overall system model. Such models can be analyzed to generate sets of malfunctions (i.e. Minimal Cut Sets) for a particular unsafe system-level condition [29]. Alternatively, if the modeling language is restricted to combinatorial logic, models can be trivially parsed to synthesize results in a fault tree format [23, 24].

In essence, this approach to MBSA can be seen as modularization and, optionally, expansion of the expressive power of the fault trees. Kaiser's State/Event Fault Trees technique [16, 17] highlights this perspective most clearly.

Overall, the approach ensures consistency between various safety analyses of the system – as these are based on a single 'central' model. However, it does not guarantee consistency with the design models and safety engineers maintain responsibility for the adequacy of MBSA results. Furthermore, although the system assessment process (that is clearly at the core of the model elicitation under this approach to MBSA) is critical to the confidence in the model and, by extension, analysis adequacy, there is little repeatable guidance on how such assessment can be conducted and how safety models can be constructed in a systematic and repeatable fashion.

Finally, it is important to note that a number of hybrid techniques have been proposed. These utilize the architecture of the design model whilst tasking safety engineers with characterizing behavior of individual components for the purpose of the safety assessment. Examples of such techniques are the integration of HiP-HOPS with Matlab Simulink proposed by Papadopoulos and Maruhn [25] and the Error Modelling Annex of the Architecture Analysis and Design Language (AADL) [12, 31]. Whilst such techniques strengthen the traceability between development and safety processes and partially automate model construction, they exhibit some vulnerabilities of both extended and dedicated safety model approaches. In particular models produced under hybrid techniques typically limit safety assessment to considering only intentional interactions between components (as in ESM approach). Similarly, such models are not provably correct with respect to the design (as under the dedicated safety model approach).

III. Engineering Semantics of Component Interfaces

The process of model construction, discussed in the previous section, is not the only important difference between various MBSA techniques. Regardless of how the models are produced, it is clearly important to ask such question as: What do they represent? What safety engineering concepts are captured by the theoretical constructs? It is therefore important to establish the safety engineering semantics of these models.

System safety and reliability can be conceptualized in different terminology and from different viewpoints. Different standards, regulators and academic researchers promote different ontologies of key safety engineering terms and concepts [34]. Whilst some, such as that by Avizienis, Laprie and Randell [4], are more prominent than others, no set of safety engineering terms is universally accepted and, more importantly, no single ontology is appropriate for use in all contexts.

MBSA is no exception. The difference between the conceptual viewpoints adopted by MBSA techniques is most revealing on components interfaces. The original MBSA methods, such as HiP-HOPS and FPTN, have modeled dependencies between the components in terms of deviations of their behavior from *design intent*. In the terminology of these techniques, such models capture dependencies of components on failure modes exhibited by other components of the system. Failure modes are typically defined through abstract categories (such as omission, commission, early, late or value deviations) and represented in the models as either individual Boolean flows or symbols of an enumerated type. Assuming that a dedicated safety model is being created, we call this general MBSA approach Failure Logic Modeling (FLM).

To briefly illustrate, a failure logic specification of a shut-off valve mentioned in the previous section would typically state that omission of hydraulic output could be caused by omission of hydraulic input or omission of the control input or an internal malfunction (failure) of the valve itself. Assuming that the valve is loaded to closed position and that its hydraulic input is intended to be constantly pressurized (e.g. the valve is connected directly to a pump), commission of the output could be caused by either commission of the control input or an internal failure.

On the other side of the semantics spectrum, lie approaches where component dependencies are captured in terms of abstracted flows of information, matter or energy and "real" physical or logical characteristics of such flows (e.g. pressure, volume, voltage, electrical current, items of data, etc.). Such flows are sometimes referred to as "nominal flows". If dedicated safety models are created, mathematically component outputs are still a function of component inputs and any internal malfunction (or, optionally, the state of the component). The example of the valve provided in the previous section follows this approach. We call this approach Failure Effects Modeling (FEM). Other examples can be found in some AltaRica models [6] as well as techniques proposed elsewhere (e.g. [22]).

FEMs can be captured using the same languages as FLMs. What distinguishes the two approaches is the model semantics and the engineering viewpoint adopted in their elicitation. This distinction, however, has serious consequences in terms of the qualities of the models and the overall safety assessment model. In particular, FLM components are characterized with respect to an implicit design intent. Such component models are generally not reusable as the intent of the physically identical components depends on their system context [19]. For example, consider again the shut off valve. Assume further that two identical valves are connected in series and controlled by an identical signal. The hydraulic input of upstream valve is connected to a pump (i.e. it is intended to be always pressurized); the output of the downstream valve is connected to the actuator assembly. Consider the commission failure mode of the valve (i.e. provision of pressure when pressure is unintended). The characterization of the upstream valve is as previously presented, i.e. commission can be result of the commission of control input or internal failure. For the downstream valve, however, this propagation condition is too pessimistic – in addition to either control commission or internal failure this valve must be exposed to a commission of the hydraulic input in order to exhibit this failure mode on the output. Thus, whilst valves are physically identical, their failure logic characterizations are different and, thus, not easily reusable.

Furthermore, we have previously demonstrated that FLMs of complex reconfigurable systems are not compositional [19] – the model must contain more information than is contained in all of its components and characterization of individual components must refer to modes of operation (that, similarly to global variables, can only be captured at higher levels of decomposition). The resultant FLMs are complex both in terms of the construction process and computational complexity of model analyses.

In contrast, FEMs are mostly compositional and reusable. Construction of such models is often more intuitive to engineers and their analysis is less complex. However, in their pure form and similarly to the FI/ESM approach described in the previous section, FEMs are limited to intentional interactions over intended path. To overcome this limitation some researchers have adapted hybrid modeling techniques [5, 18] whereby intentional interactions are modeled using a FEM approach whilst unintentional interactions – such as short circuits of electrical systems or leaks in hydro-mechanical systems – are characterized as Failure Modes dependencies and utilize the FLM approach.

Despite apparent benefits of such hybrid approach, we have discovered that the process of FLM construction requires a higher level of intellectual engagement by safety engineers and more thorough review of the system design than construction of FEMs or Hybrid Models. During the FLM construction system design proposal is reviewed from a perspective fundamentally dissimilar to that of the development process. Consequently, important limitations of the system architecture are often identified as a direct result of such review and before the model is even completed and analyzed. In contrast, safety assessment based on FEMs or hybrid models adopts a similar perspective to the development process and cannot fully satisfy the objective of having conducted an analytically diverse review of the design.

IV. Classification of Techniques

Table 1 shows classification of existing MBSA techniques according to the two criteria described in the previous sections. It is important to reiterate that the criteria are orthogonal. For example, whilst modeling of FM propagation between components is typically associated with construction of dedicated models we are aware of at least one technique – Software Deviation Analysis [28] – that combines FM-focused perspective with the full utilization of the design model. Also worth noting is a technique developed by Joshi and Heimdahl [15] that extends the "pure" Failure Injection approach by allowing safety engineers to define new dependencies between the components arising from unintentional interactions in presence of failure. Such dependencies are characterized in terms of failure mode propagation leading to an interesting hybrid between the FI and FLM approaches.

TABLE I. Classification of MBSA Techniques

		Engineering Semantics of Component Dependencies		
		Only Nominal (Energy, Matter & Information) Flows	***Both Nominal & FM Flows***	***Only FM Flows***
Model Provenance	***Dedicated Safety Model***	Failure Effects Modeling e.g. some AltaRica models [6], Majdara & Wakabayashi technique [22]	FEM/FLM Hybrid Approach e.g. some AltaRica Models [5, 18]	Failure Logic Modeling e.g. FPTN [13], FPTC [37], HiP-HOPS [23]
	Partial Utilization of Design Model (e.g. architecture)	Hybrid Approach e.g. ESACS/ISAAC ESM approach if underlying system models have to be simplified [10, 11]	Hybrid Approach e.g. Joshi & Heimdahl [15]	Hybrid Approach e.g. HiP-HOPS integration with Simulink [25]
	Automated Construction / Utilization of Design Model	FI/ESM approach [1, 8, 26]		Hybrid Approach e.g. Software Deviation Analysis [28]

Finally, it is important to note that we have intentionally excluded the Error Modeling Annex of AADL (previously mentioned in section 2) from the classification. In terms of model provenance criteria the annex clearly belongs to the middle, ‘hybrid’, row of the table since characterizations of the components have to be manually constructed whilst the structure, connectors and bindings of the ‘core’ AADL model are also utilized. Classification in terms of the engineering semantics of component dependencies is however less clear. The Error Modeling Annex terminology and documentation [31] suggests that that the authors intent was to support Failure Logic Modeling. However, constructs such as *Guard_Event* and *Guard_Transition* allow integration of the Error Modelling Annex with the behavior of the ‘core’ model thereby enabling a hybrid approach (that would be placed in the middle cell of Table 1). Overall, in our view, the annex constructs can be equally used for construction of Failure Effects Models even though it is unlikely that this was the original intent of the developers. Therefore we believe that the place of the AADL in our classification depends on the usage of the language by the safety engineers.

These observations about AADL also highlight an important confusion in MBSA research – that between modeling language and methodology. Original MBSA techniques (e.g. FPTN and HiP-HOPS) have been based on their own idiosyncratic modeling languages. In this context modeling constructs were inherently linked to safety engineering concepts they represented. Extension or revision of the methodology required changes to the modeling language (and vice versa). However, usage of idiosyncratic languages, whilst being a useful tool for research, impedes industrial adoption of MBSA techniques: such languages are unfamiliar to even specialist engineers, they are often not fully validated in use and usually not supported by sufficiently mature tools. The pragmatic approach to industrial application of MBSA is likely to require use of third-party, commercial and industrially-mature languages and modeling environments such as Matlab Simulink or SCADE. Such languages can be shared by a number of engineering disciplines making continuous improvement of modeling environments economically feasible as well as facilitating more thorough review and validation of the language and software by a larger community of professional engineers and researchers.

V. Applicability & Challenges of MBSA Techniques

Different MBSA approaches presented in this paper do not necessarily conflict and, instead, are often applicable at different stages of the safety lifecycle. Whilst different models of safety lifecycles exist in different domains and regulatory jurisdictions these are based on similar concepts. For example in aviation domain ARP4754 and 4761 documents [32, 33] define the following three primary stages of the safety process:

Functional Hazard Assessment (FHA): a hazard identification stage described by the standard as “a systematic, comprehensive examination of aircraft functions to identify and classify Failure Conditions of those functions according to their severity” [32]. FHA is conducted at both aircraft and system levels.

Preliminary System Safety Assessment (PSSA): “a systematic evaluation of a proposed system architecture and its implementation, based on the Functional Hazard Assessment and failure condition classification, to determine safety requirements for all items in the architecture” [32]. PSSA is intended to be a design-driving and, thus, iterative process that starts at the earliest stages of system design.

System Safety Assessment (SSA): “a systematic, comprehensive evaluation of the implemented system to show that the relevant safety requirements are met” [32]. Unlike PSSA, SSA is performed on the finished design feeding into a V&V rather than design function. In other words, it lies on the right hand side of the standard “V Model” of the development process.

These three stages are supported by a Common Cause Analysis (CCA). Furthermore, the recently published revision of the standard – ARP4754a – also defines Preliminary Aircraft System Safety Assessment (PASA) that can be broadly seen as multi-system PSSA. However, FHA, CCA and PASA lie outside the scope of this paper and we focus our discussion exclusively on PSSA and SSA.

Similarly, we constrain our discussion to the most prominent families of MBSA techniques – ESM/FI, FLM and FEM/FLM Hybrid. Although historically used in industry, the “pure” FEM approach, in our view, is too constrained and provides no advantages over the FEM/FLM hybrid. The maturity of other MBSA techniques (except for those considered below) does not yet permit discussion about their applicability.

A. *FLM and Hybrid Techniques*

In our view both FLM and the hybrid approach can support exploratory safety assessment of system architecture (e.g. fulfilling the objectives of the PSSA). Both approaches can be applied early in the development process and in the context of iterative and incremental safety assessment [20]. However, the significant additional effort of constructing FLMs is not necessarily justified for every system in the context of product lines with a relatively stable architecture (such as an aircraft family). In such contexts it may be sufficient to use a less time consuming hybrid approach.

In contrast, when a proposed system design is based on a significantly new architecture (such as a typical design of systems for a new family of aircraft or design of a new railways control concept) the significantly higher effort necessary for constructing FLMs may be justified by the value added by a thorough review of the architecture under this approach. This scenario, of course, includes establishing of a *new* product line.

However, both approaches rely on construction of the new safety model that captures the safety engineers’ hypotheses about behavior of the system. The adequacy of these models would determine the adequacy of the model-based PSSA just as the adequacy of fault trees today determines the adequacy of the ‘classical’ PSSA. We have previously demonstrated that construction of adequate safety models relies on existence of systematic and repeatable modeling methods and comprehensive guidance [21]. Whilst we have developed such guidance for the FLM approach and performed partial evaluation, further independent evaluation is necessary in order to establish whether our “FLM Handbook” provides sufficient support to the construction of FLM models comparable with the support Fault Tree Analysis Handbook [35, 36] provides for construction of fault trees.

Furthermore, we have argued that whilst usage of modeling languages with well-defined semantics contributes to confidence in model adequacy it is not sufficient for such confidence. We are advocating implementation of the FLM and

FEM/FLM hybrid approaches on the basis of well-defined (and relatively limited) subsets of standard languages such as SCADE or Matlab Simulink which, in our view, will facilitate industrial adoption of the MBSA techniques as well as ensuring that MBSA discipline benefits from the extensive validation of these languages by large and cross-disciplinary community of professional engineers and researchers. We are currently conducting further research into implementation of the FLM methodology in the context of Matlab Simulink and Stateflow.

B. FI/ESM Techniques

In contrast to the FLM and hybrid FEM/FLM techniques, in our view the FI/ESM approach is incapable of fulfilling objectives of the PSSA in most cases. Safety analysis conducted on the basis of the FI/ESM is, in our view, too constrained by the design model of the system – a model that is typically constructed with no consideration for the needs of safety assessment. It is difficult to justify confidence in such models and subsequent analysis. Furthermore, FI/ESM techniques cannot be applied before an executable design model of the system emerges. This typically happens at the later stages of the development process when design of the system can no longer be easily influenced and the cost of design change is high.

Nevertheless, the guarantee of consistency between design models and safety analysis results makes FI/ESM an attractive approach to (partial) validation of the FLMs and FEM/FLM hybrids (or even ‘classical’ fault trees) and, therefore, applicable at the SSA stage of the ARP safety lifecycle.

We believe that use of FI/ESM techniques as the main or sole basis for safety assessment can only be justified when the system is being modified and only when extent of the modifications is small and does not affect the system architecture. Note that this scenario of application is fully consistent with the position above given that, in such circumstances, PSSA may be deemed as unnecessary and re-assessment may be bound to the confirmatory SSA.

Most research challenges associated with the FI/ESM approach lie in the Computer Science domain and are concerned with improvement of the power and time-complexity of analysis tools (such as model checkers) in the context of large-scale and complex system models. Indeed today many models of complete systems (e.g. individual aircraft systems) are not tractable by the analysis tools available. At the same time there are residual methodological issues to be addressed. In particular, as was previously mentioned, methods for systematic identification of failure modes that are to be injected into design models need to be developed. This is likely to include establishing coverage metrics of the extension (i.e. the set of all injected FMs) as well as developing comprehensive guidance for model extension process.

VI. Conclusions

This paper has presented a review of most prominent Model Based Safety assessment techniques. We have shown that a MBSA encompasses a large number of fundamentally dissimilar techniques that vary significantly in terms of applicability, limitations and challenges. We have presented a simple classification schema for MBSA approaches based on two criteria – provenance of the model and engineering semantics of component dependencies captured by the model. Whilst different classifications can be developed, our schema allows to rationalize most of the existing techniques into coherent families that share key features. To our knowledge no such classification has been previously proposed and we hope that this paper will contribute to principled and organized discussions of future MBSA research.

We conclude with two further observations on the state of MBSA research. Firstly, the cross-disciplinary nature of MBSA research should be stressed. The topic combines elements of System Safety Engineering and Computer Science disciplines. Issues such as the system assessment and model elicitation methods and guidance, ontologies of safety engineering terms (that form the abstract language of the model) and of the formulation of analyses that can be performed on the basis of the models lie in the System Safety Engineering domain. Issues such as the definition of the semantics of concrete modeling languages as well as development and improvement of the analysis algorithms and tools lie in the Computer Science (and applied software engineering) domain. Without a clear mapping between Computer Science and System Safety Engineering domains, the MBSA advances in the former remain infeasible whilst the Computer Science advances remain unjustified and unmotivated by real-world problems.

In particular, given the relative maturity of MBSA research and some of the techniques, we believe that development of further MBSA languages and notations is unlikely to have significant safety engineering impact. At the very least such development should be justified by a comprehensive review of the existing languages (within and outside of the immediate MBSA application domain) and motivated by inability to formalize certain safety engineering concepts or their relationship in any existing language.

Secondly, MBSA research publications are often adversarial in nature. Authors typically justify a technique they (or their close collaborators) have developed and report on successful case studies. Whilst such publications are important for newly developed techniques, we observe that impartial evaluation of techniques, their comparison, and negative results of evaluation activities are currently underreported. Instead to establish MBSA as a mature research discipline requires impartial, principled and comprehensive evaluation with detailed comparison of different techniques.

References

1. Abdulla, P.A., J. Deneux, G. Stålmarck, H. Ågren, and O. Åkerlund. Designing Safe, Reliable Systems Using Scade. in 1st International Symposium on Leveraging Applications of Formal Methods (ISoLA). 2004. Paphos, Cyprus: Springer-Verlag.
2. Åkerlund, O., P. Bieber, et al., ISAAC, a Framework for Integrated Safety Analysis of Functional, Geometrical and Human Aspects, in 3rd European Congress on Embedded Real Time Systems (ERTS). 2006: Toulouse, France.
3. Arnold, A., G. Point, A. Griffault, and A. Rauzy, The AltaRica Formalism for Describing Concurrent Systems. Fundamenta Informaticae, 2000. 34: p. 109-124.
4. Avizienis, A., J.-C. Laprie, B. Randell, and C. Landwehr, Basic Concepts and Taxonomy of Dependable and Secure Computing. IEEE Transactions on Dependable and Secure Computing, 2004. 1(1): p. 11-33.
5. Bernard, R., J.-J. Aubert, P. Bieber, C. Merlini, and S. Metge, Experiments in Model-Based Safety Analysis: Flight Controls, in IFAC Workshop on Dependable Control of Discrete Systems. 2007, The International Federation of Automatic Control: Paris, France.
6. Bieber, P., C. Castel, and C. Seguin. Combination of Fault Tree Analysis and Model Checking for Safety Assessment of Complex System. in 4th European Dependable Computing Conference. 2002. Toulouse: Springer Verlag.

7. Boiteau, M., Y. Dutuit, A. Rauzy, and J.-P. Signoret, The AltaRica Data-Flow Language in Use: Modelling of Production Availability of a Multi-State System. Reliability Engineering and System Safety, 2006. 91(7): p. 747-755.
8. Bozzano, M. and A. Villafiorita. Improving System Reliability via Model Checking: The FSAP/NuSMV-SA Safety Analysis Platform. in 22nd International Conference on Computer Safety, Reliability and Security (SAFECOMP). 2003. Edinburgh, UK: Springer-Verlag.
9. Bozzano, M., A. Villafiorita, et al. ESACS: an Integrated Methodology for Design and Safety Analysis of Complex Systems. in European Safety and Reliability Conference (ESREL). 2003. Maastricht: Balkema Publishers.
10. Bretschneider, M., Lessons Learnt About System Safety Assessment Based on Scade Models (Presentation), in Model-based Safety Assessment (Journées MISSA), C. Seguin, Editor. 2010, CISEC / MISSA Project Consortium: Toulouse, France.
11. Cavallo, A., System Safety Assessment Based on Formal Models: Lessons Learnt by Alenia Aeronautica (Presentation), in Model-based Safety Assessment (Journées MISSA), C. Seguin, Editor. 2010, CISEC / MISSA Project Consortium: Toulouse, France.
12. Feiler, P. and A. Rugina, Dependability Modeling with the Architecture Analysis & Design Language (AADL), in Performance-Critical Systems Initiative. 2007, Software Engineering Institute, Carnegie Mellon University (SEI/CMU): Pittsburgh, PA.
13. Fenelon, P. and J.A. McDermid, An Integrated Toolset for Software Safety Analysis. The Journal of Systems and Software, 1993. 21(3): p. 279-290.
14. Fenelon, P., J.A. McDermid, M. Nicholson, and D.J. Pumfrey, Towards Integrated Safety Analysis and Design. ACM Computing Reviews, 1994. 2(1): p. 21-32.
15. Joshi, A. and M.P.E. Heimdahl. Behavioral Fault Modelling for Model-based Safety Analysis. in 10th IEEE High Assurance Systems Engineering Symposium (HASE). 2007. Dallas, TX: IEEE Computer Society.
16. Kaiser, B. and C. Gramlich. State-Event-Fault-Trees – A Safety Analysis Model for Software Controlled Systems. in 23rd International Conference on Computer Safety, Reliability, and Security (SAFECOMP). 2004. Potsdam, Germany: Springer Berlin / Heidelberg.
17. Kaiser, B., P. Liggesmeyer, and O. Mäckel. A New Component Concept for Fault Trees. in 8th Australian workshop on Safety critical systems and software. 2003. Canberra, Australia: Australian Computer Society.
18. Kehren, C., C. Seguin, et al., Advanced Simulation Capabilities for Multi-Systems with AltaRica, in 24th International System Safety Conference (ISSC). 2004, System Safety Society: Providence, RI.
19. Lisagor, O., Failure Logic Modelling: A Pragmatic Approach, in Dep't Computer Science. 2010, The University of York: York.
20. Lisagor, O., M. Bozzano, M. Bretschneider, and T. Kelly, Incremental Safety Assessment: Enabling the Comparison of Safety Analysis Results, in 28th International System Safety Conference (ISSC). 2010, System Safety Society: Minneapolis, MN.
21. Lisagor, O., L. Sun, and T. Kelly, The Illusion of Method: Challenges of Model-Based Safety Assessment, in 28th International System Safety Conference (ISSC). 2010, System Safety Society: Minneapolis, MN.
22. Majdara, A. and T. Wakabayashi, Component-Based Modelling of Systems for Automated Fault Tree Generation. Reliability Engineering and System Safety, 2009. 94(6): p. 1076-1086.
23. Papadopoulos, Y. Hierarchically Performed Hazard Origin and Propagation Studies. in 18th International Conference on Computer Safety, Reliability, and Security (SAFECOMP). 1999. Toulouse, France: Springer-Verlag.
24. Papadopoulos, Y., Safety-Directed System Monitoring Using Safety Cases, in Dep't Computer Science. 2000, The University of York: York.
25. Papadopoulos, Y. and M. Maruhn, Model-Based Synthesis of Fault Trees from Matlab-Simulink Models, in International Conference on Dependable Systems and Networks (DSN). 2001, IEEE Computer Society: Goteborg (Sweden). p. 77-82.
26. Peikenkamp, T., A. Cavallo, et al. Towards a Unified Model-Based Safety Assessment. in 25th International Conference on Computer Safety, Reliability and Security (SAFECOMP). 2006. Gdansk, Poland: Springer-Verlag.
27. Rauzy, A., Mode Automata and Their Compilation into Fault Trees. Reliability Engineering and System Safety, 2002. 78(1): p. 1-12.
28. Reese, J.D. and N.G. Leveson. Software Deviation Analysis. in 19th International Conference on Software Engineering. 1997. Boston, MA: ACM.
29. Seguin, C., P. Bieber, C. Castel, and C. Kehren, Formal Assessment Techniques for Embedded Safety Critical Systems, in 2nd National Workshop on Control Architectures of Robots (CAR'2007). 2007: Paris, France.
30. Selic, B., The Pragmatics of Model-Driven Development. IEEE Software, 2003. 20(5): p. 19-25.
31. Society of Automotive Engineers, Architecture Analysis and Design Language (AADL), Annex E: Error Model Annex (Annex Volume 1). 2006, SAE Aerospace: Warrendale, PA.
32. Society of Automotive Engineers (SAE), Certification Considerations for Highly-Integrated or Complex Aircraft Systems. 1996, SAE International / EUROCAE: Warrendale, PA.
33. Society of Automotive Engineers (SAE), Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment. 1996, SAE International: Warrendale, PA.
34. van der Meulen, M., Definitions for Hardware and Software Safety Engineers. 2000, London: Springer-Verlag.
35. Vesely, W.E., J. Dugan, J. Fragola, J. Minarick, and J. Railsback, Fault Tree Handbook with Aerospace Applications. 2002, NASA Office of Safety and Mission Assurance.
36. Vesely, W.E., F.F. Goldberg, N.H. Roberts, and D.F. Haasl, Fault Tree Handbook. 1981, US Nuclear Regulatory Commission,.
37. Wallace, M. Modular Architectural Representation and Analysis of Fault Propagation and Transformation. in 2nd International Workshop on Formal Foundations of Embedded Software and Component-Based Software Architectures (FESCA 2005). 2005. Amsterdam: Elsevier.