Addressing Challenges of Hazard Analysis in Systems of Systems

George Despotou, Robert Alexander, Tim Kelly
High Integrity Systems Engineering, Department of Computer Science
University of York
York, United Kingdom
george, robert.alexander, tim.kelly@cs.york.ac.uk

***Abstract*—Hazards are situations that can result in accidents. Depending on the domain, this can include loss of lives, injuries and economic or environmental disasters. For example, a common hazard in the aviation domain is in flight engine shutdown. Hazard analysis is the process of discovering hazards in a system. This activity has been performed for many years in safety engineering and is a straightforward activity in most domains. In recent years a new class of systems has emerged, distinguished from traditional (monolithic) systems by a combination of characteristics such as autonomous and independently developed components, increased complexity and geographic dispersion. These characteristics introduce a number of challenges for traditional hazard analysis. This paper describes these challenges and proposes two complementary approaches that address them: Dependability Deviation Analysis (DDA) and simulation-based hazard analysis (SimHAZAN). The paper then describes a model-driven approach that combines the two and thereby provides an underlying framework for their application during system development.**

Keywords: Hazard analysis, deviation analysis, safety simulations, safety requirements, hazard analysis metamodel

I. INTRODUCTION

Safety is a system attribute acquired as an integral part of system development. It is a negative attribute – safety requirements express events and situations that we want to avoid. In safety engineering those unwanted situations are called hazards, and (in a sense) every safety requirement is derived from the idea that "Hazard X (or X, Y and Z) should not occur" [1]. It is accepted that a safety lifecycle should be followed in parallel with the system's main lifecycle, and a key part of this lifecycle is hazard analysis. In this analysis, safety engineers (along with the system stakeholders and domain experts) identify the hazards that could manifest during system operation.

Traditionally, hazard analysis techniques have involved engineers manually working over paper system models, brainstorming deviations of expected behaviour and mentally projecting possible safety consequences [2]. In most domains (such as manned vehicles and process plants) there are methods that are well-established and effective. Increasingly, people are proposing and promoting systems that are highly decentralised, yet highly integrated (through computer network technology). Some of these are safety-critical, and performing hazard analysis on them is a major new challenge.

II. SYSTEMS OF SYSTEMS

There are many problems today that cannot be solved with traditional monolithic systems. The phrase "Systems of Systems (SoS)" has been introduced to describe classes of systems such as the Network Centric Warfare (NCW) paradigm and Air Traffic Control systems. Figure 1 illustrates a NCW SoS in which a command centre, artillery, troops and unmanned aerial vehicles collaborate to suppress guerrilla activity. In order to capture the different facets of the operation of a SoS, it is quite common that modelling frameworks such as DODAF [3] to be used, to model the SoS.

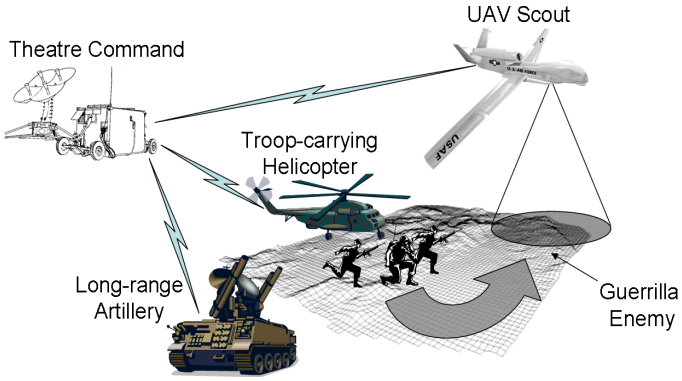

Figure 1. An example of a (NCW) System of Systems

There are a number of definitions of 'SoS'. Although there are variations depending on the particular domain in which the term System of Systems is defined, there is a (essential) set of characteristics that a System of System demonstrates [4]:

- *Overall objectives*: A SoS is tied together by a set of high level goals of interest to its stakeholders, such as provision of air traffic management (for ATC) or accomplishment of a mission (for NCW).
- *Complexity*: SoS are typically used in problems of high complexity (e.g. aircraft route planning or targeting).
- *Multiple elements*: A SoS consists of many elements which are systems in their own right and can or have

been developed independently from the SoS (e.g. a radar or an unmanned aircraft).

- *Autonomy*: Elements are able to make their own decisions with varying degrees of autonomy (e.g. autonomous vehicles).
- *Geographical dispersion*: SoS often involve elements that are geographically dispersed and mobile – changing their position according to the overall SoS objectives.
- *Collaboration*: SoS elements collaborate, each contributing different functions in order to achieve the overall SoS objectives (e.g. sharing of intelligence requires that some elements sense while others analyse data).
- *Communication*: Collaboration between the SoS elements requires exchanges of information.
- *Heterogeneity*: Elements have been developed independently from each other, potentially with different technologies and from different developers.

Overall, a System of Systems can be considered as being an organised complex unity assembled from dispersed, highly collaborating, autonomous systems – each of which is capable of operating independently.

III. Hazard Analysis Challenges in Systems of Systems

Systems of Systems demonstrate a combination of characteristics that traditional safety-critical systems don't have. These characteristics introduce a number of challenges that make hazard analysis difficult to perform.

A. Evolving Systems

It is often claimed that creating a SoS is not an issue of design but an issue of integration of different systems. Although this does not completely represent reality, it demonstrates the fact that it is very difficult to even describe the development lifecycle of a SoS. SoS constantly evolve and adapt according to their environment – their concept of operations changes, new elements are added and new capabilities are needed. This means that as the SoS changes, the hazards on which safety analysis was based change.

Hazard analyses will need to be performed during any SoS change to identify new hazards so that they can be mitigated. Traceability between hazards and system design is vital for achieving this. It allows understanding how the behaviour or the SoS elements individually and collectively can result in the identified hazards. Hence revealing the effectiveness or limitations of the hazard mitigation and prevention measures in place in the SoS.

B. Emergent Behaviour

The overall behaviour of a SoS cannot be described purely in terms of the behaviour of its elements. Instead, its behaviour *emerges* from that lower-level behaviour. Moreover, there is often competition between local objectives and SoS objectives. This means that a goal of an element of the SoS may not necessarily be beneficial for the entire SoS. For example, routing an airborne radar over high risk areas will affect (increase the risk) its safety. However it can be beneficial for other elements of the SoS such as troops and other aircraft that will receive more detailed intelligence.

C. Functional Composition

Overall SoS functions are usually composed from many element functions. For example maps depicting intelligence about the theatre of operations can be thought of as a single SoS capability. However this capability involves the functions of many individual SoS (sensor) elements such as radars, infra-red sensors and human reports. It follows that it is not always apparent if a given element function can result in a hazard. Moreover, it can be the case that all elements perform as expected, but the resulting overall SoS behaviour is unsafe.

D. Network Centered

SoS elements are distributed and often highly autonomous, but they also rely heavily on a communication network in order to collaborate. This combination of mobility and autonomy with network-dependence leaves SoS very vulnerable to network problems. The network itself allows many errors (particularly sensing or judgement errors) to be propagated, having safety effects elsewhere in the system.

E. Multi-attribute Failures

Depending on how the SoS elements collaborate, failures can propagate and manifest themselves as different types (of failure) in different elements of the system [5]. For example, a reliability failure at the network level (e.g. loss of a relay) may result in performance bottlenecks at element level and unavailability of capability at the SoS level. Additionally, achieving the required behaviour in order for a hazard to be mitigated is not always straightforward, as there are conflicts between system attributes that will inevitably result in trade-offs. Due to conflicts between SoS attributes, it is inevitable that not all SoS requirements will be satisfied as originally expected. For example, consider the SoS of figure 1. Safety can be expected to be at odds with response performance. Focusing entirely on safety without considering response time may result in a safe but useless design. SoS developers need to understand the relationship between these attributes and make an informed decision as to the most appropriate and in the same time most optimal design. In the same time the SoS has to maintain the satisfaction of the stakeholders' requirements to (at least) tolerable levels for its purpose.

Given challenges A-E above, it follows that we need an approach to hazard analysis that is still viable given the increased scope mandated by challenge E and the practical problems raised by challenges A-D. For example, consider safety analysis by constructing a fault tree, a very common method in traditional safety analysis. One of the problems is that fault trees are configuration dependant. Hence SoS reconfiguration would invalidate the calculated probabilities. Capability in a SoS can be dynamic composed out of a number of different platforms. A safety analyst would have to know precisely the dependency (of capability) on each platform to

provide a realistic probability. Dynamic operation and reconfiguration can be an impediment to that. Furthermore, most established analysis methods are manual, and do not work well in the presence of the practical problems above – it is difficult for humans to manage this complexity unaided. Such techniques also limit themselves (again, for reasons of manual practicality) to safety failures only – they do not consider the safety effects of failures in other attributes.

The approach presented in this paper consists of two analysis methods combining the advantages of exploratory and (semi-automated) simulation analysis. Beginning from hazards requires (top-down) deductive reasoning, making in the same time a number of assumptions about the SoS (e.g. configuration). Exploratory analysis is inductive and combines top-down and bottom analyses. Instead of identifying the contributing factors to hazards, exploratory analysis focuses on SoS elements identifying the effects of potential deviations. The effects then are associated with the identified hazards. The benefits of SimHAZAN become apparent particularly in SoS, the complexity of which makes manual analysis methods difficult to apply. The two methods work complementary to each other and are combined in a common framework. Although the two approaches offer a number of advantages they do not solve all problems. The framework allows the integration of other safety analysis methods (e.g. fault trees) in a single framework.

IV. Using Dependability Deviation Analysis to Identify Hazards

DDA is an analysis method used to identify potential failure conditions with respect to dependability attributes and examine how these interrelate. DDA can identify problems that can impact any dependability attribute. A failure that occurs during the operation of the system may impact on the achievement of a dependability attribute, which in its turn may have an impact on other attributes or result in unacceptable system operation. Impact on safety is the single focus in this paper. DDA allows considering failure conditions from the perspective of (dependability) attributes of interest to the stakeholders, such as performance, availability and security. DDA is an exploratory method focusing on the effects of failure conditions and (in this paper) its impact on safety.

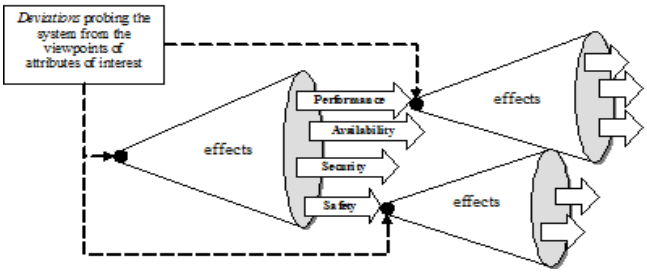

Figure 2. Propagation of multi-attribute failures

In contrast, other safety analysis methods investigate causes. However the complexity of a SoS makes it difficult to exhaustively identify potential causes of a failure. DDA is a deviation based analysis technique such as HAZOPS and FMEA [6]. In the fashion of these already established a representative set of guidewords is used to prompt the elements of the SoS, in order to reveal possible failures from the viewpoints of dependability attributes. Figure 3 shows an example of DDA on the (MODAF) OV-2 model of the SoS of figure 1 (MODAF is the DODAF equivalent used by the UK MOD [9]).

System Element Type	System Element ID	Guideword	Failure Condition	Failure Condition ID
Needline	1	Public (Security)	Disclosure of aircraft position	FC1
	7	Overload (Performance)	Slow transmission of target data	FC2
		Fake (Security)	Artillery receive fake target data with malicious intent	FC3
		Public (Security)	Enemy receives firing intent and target information	FC4
		Omission (Availability)	Artillery will not receive any target data	FC5
		Value (Reliability)	Artillery will receive the wrong location/order	FC6
		Late (Performance)	Delay or possibly loss of request of target data	FC7

Figure 3. An example of DDA using DODAF OV-2 model

DDA provides a methodical way of identifying the relation between failures. The analysts then identify the credible failure conditions and examine their effects. The effect of a deviation may constitute the cause for a failure of a different dependability attribute. Ultimately a chain of dependability failures that may result in a SoS hazard affecting its overall safety (which is the single attribute of interest in our case), is established. A failures map is a composition of the chains of effect of the identified failure condition. Figure 4 shows a failures map extract from SoS of figure 1. The ovals represent deviations from intended operation and consist of guideword and a system element (e.g. *fake::needline7*). Credible deviations are captured as failure conditions (rectangles) and their effects as associations (arrows) between failure conditions.

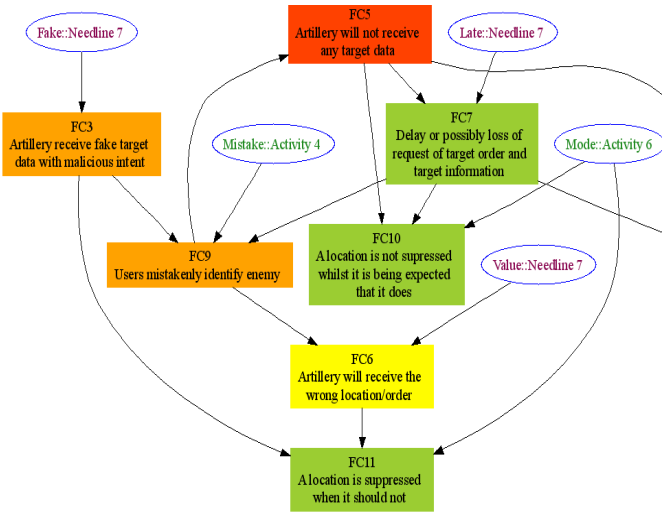

Figure 4. Extract of a failures map of the SoS example in figure 1

Having identified and assessed the effects of the failure conditions, analysts elicit the safety related SoS behaviour (e.g.

needline X needs to response within Y milliseconds). DDA provides a methodical way to identify the relation between failures, which have been revealed after examining the design from the viewpoint of each attribute of interest.

V. Simulation Based Hazard Analysis

As noted, the major hazard analysis challenge of SoS comes from the potential distance between an SoS-level hazard and its causes. This distance may be in terms of space, time, or logical network steps. In a manual technique (even an advanced one such as DDA) it is difficult to find all these hazards – we would need to consider a huge number of complex scenarios.

An alternative to this is to use a simulation-based technique, as described in [7]. In this approach (henceforth SimHAZAN), we build a simulation model of the SoS and analyse it to see what accidents can occur, and how. The model uses an agent to represent each entity in the SoS. Typically, several different vignettes will be defined for a given SoS – these will vary in terms of mission, environment, opposition and other parameters. Possible deviations are defined for each agent, and the vignette is run with repeatedly with different combinations of those deviations. When accidents occur in the simulation (in response to deviations) they are logged, and we can watch the simulation run again to study them. The output of the simulation is then examined to explain how the accidents occurred, and out of this we can identify system hazards.

To supplement manual review of the simulated accidents, we can use machine learning algorithms to find general patterns that relate deviations to accidents. These give us hypotheses about accident causation that we can investigate further (for example, "Deviation X and Deviation Y are safe individually, but together they can cause Accident A"). We can also use agent tracing techniques to build explanations of individual accidents. The tracing tool works by working backwards through the log of simulation events, using hand-crafted rules that relate events to known causes of those events. For example, if an agent fires at a given target, then a prior plan to attack that target can be inferred as the cause.

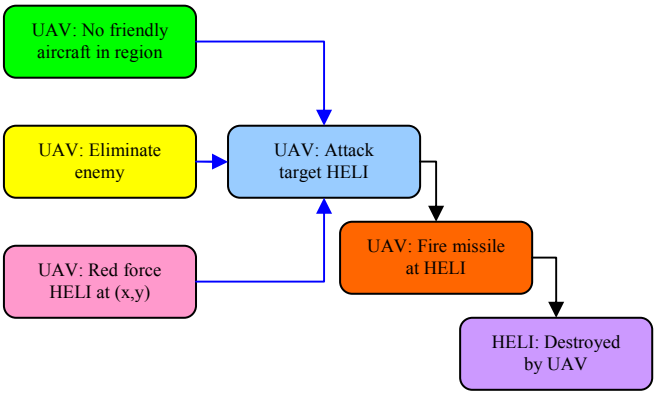

Figure 5. Agent behaviour trace from a SimHAZAN model

Figure 5 gives an example of the output from a tracing tool: the accident event "HELI: Destroyed by UAV" is explained by prior events. In this case, the helicopter ("HELI") was (incorrectly) assessed by "UAV" to be a hostile ("red force"), and as "UAV" had the prior goal to "Eliminate enemy" it attacked the helicopter.

In order to perform SimHAZAN, the simulation creator must base their simulation models on system models of the SoS. This creates a need for traceability between models created during system definition and the models created for SimHAZAN.

The deviations that are imposed on the simulation can be taken from a pool of deviations that were defined during DDA. Establishing traceability between the output from DDA and the output from SimHAZAN and comparing their results can provide a measure of sensitivity analysis between the two techniques.

SimHAZAN, in practice, has a number of limitations:

- Creating simulation models, evaluating and accrediting them, is very expensive. It may be possible to share this effort with modelling for other purposes, but this is problematic because useful simulation modelling must take account of the purpose of the model. There is no way to create a generic "simulation of the system".
- A simulation model can be extensively evaluated and still be wrong.
- Simulations can be very complex; if a simulated accident occurs it can be difficult to determine why it happened. Learning and tracing tools can help but this still requires human intelligence and creativity.
- It is difficult to argue that a SimHAZAN analysis is adequately complete. There are no extant criteria for establishing this – there are no obvious "stopping rules" for SimHAZAN modelling or analysis.

Most of the above problems, however, are inherent in the complexity of the SoS systems that SimHAZAN is designed for; we cannot expect other techniques to do dramatically better. Just like other (manual) approaches, SimHAZAN has relative strengths and weaknesses. It is the position of these authors that an automated approach to hazard analysis is necessary if complex SoS are to be safe. SimHAZAN provides such an automated approach.

VI. Integrating the Processes During SoS Evolution

SimHAZAN and DDA constitute two different approaches for performing hazard analysis. The two approaches complement each other and work together towards informing the safety engineers about the potential hazards in the SoS.

Figure 6 show an overview of how the two approaches can work together during system development. Initially, the SoS stakeholders define the concept of operations (CONOPS) of the SoS. A concept of operations is a (usually) free-form description of the envisioned capabilities of the SoS, the deployment environment and usage scenarios. CONOPS usually contain abstract information and do not go into technical detail. Subsequent refinement of the CONOPS produces the models that define the SoS. Modelling

frameworks that allow the definition of multiple viewpoints, such as MODAF and DODAF, are often used to represent SoS. It is common for models such as activity diagrams and information exchange diagrams (in DODAF terminology, OV-5 and OV-2 respectively) to be the starting point of CONOPS refinement and model definition.

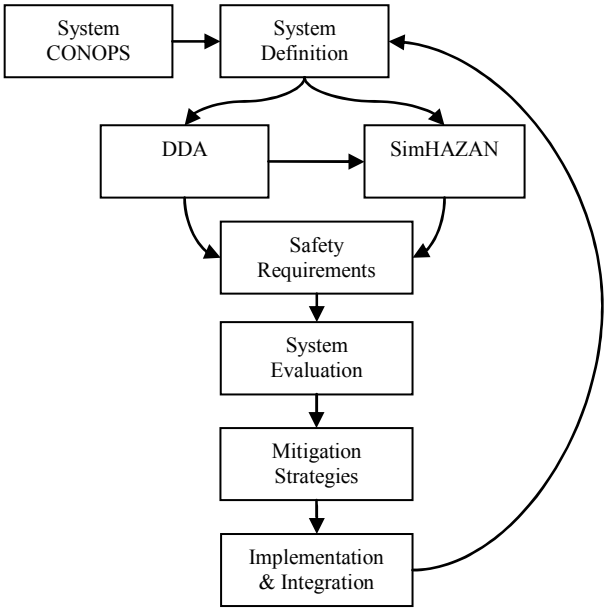

Figure 6. Use of hazard analysis techniques to inform the SoS lifecycle

Definition of SoS models allows application of DDA and SimHAZAN. DDA uses the system elements of the SoS (DODAF) models to prompt them with guidewords. DDA will then produce the failures map, recording how a failure at one system element can result in a failure of different type at another system element. The failures the effects of which can impact safety are further examined to elicit safety requirements.

SimHAZAN uses the SoS models as the basis on which the simulation is ran. Upon completion (or sometimes in parallel) to DDA, SimHAZAN is performed, using the system models created during system definition, and the deviations that have been defined during DDA. SimHAZAN will produce a set of hazards, which combine with the hazards that were identified during DDA. Motivated by the identification of hazards, developers will elicit (safety derived) requirements. These can include both negative (e.g. hazard of artillery friendly fire has been mitigated) and positive (e.g. intelligence update should take place every 100ms). At the end of this stage, developers will have identified the (safety related) dependability attributes. A requirement profile is then built for each SoS element constituting a collation of requirements identified from the perspective of all safety related dependability attributes [5]. Nevertheless application of DDA and SimHAZAN should not be limited only to distinct SoS elements. Capabilities that result from the composition of individual element functions (e.g. synthetic maps, information exchanges) need to be examined carefully using the appropriate models (those models that distinctly capture these elements). Examining this aspect of SoS may require strategies that take into account emergent behaviour and autonomy of the SoS. Definition of safety policy [8] is one such strategy. Following elicitation of safety requirements, the SoS developers evaluate whether the SoS can satisfy these requirements. This includes the degree of satisfaction of derived requirements – not all requirements will be (practicably) achieved – and the ability of the system to detect and prevent safety related failures, and mitigate their effects.

For the failures that are not addressed satisfactorily, new mitigation strategies are proposed, which are then designed implemented and integrated to the SoS. Following the implementation of the mitigation strategies or any other safety related design decision, the system models are updated to record the new information. A new cycle of safety analysis will examine the behaviour of the updated SoS.

VII. A Model Driven Unifying Framework

Good traceability between the hazard analysis and system development activities and their artefacts is necessary in order to perform hazard analysis. For example, consider traceability between simulation models and system models. Good traceability improves the fidelity of the simulation as any simulation model element can be traced to the real system. Moreover, traceability is also required, so that the different artefacts of the hazard analysis activities to be examined collectively. For example consider a set of failure conditions, some of which were identified during DDA, some during SimHAZAN, and a number of which identified during both processes. Good traceability is required to uniformly relate these failure conditions (and most importantly relating their effects) without duplication or misinterpretation.

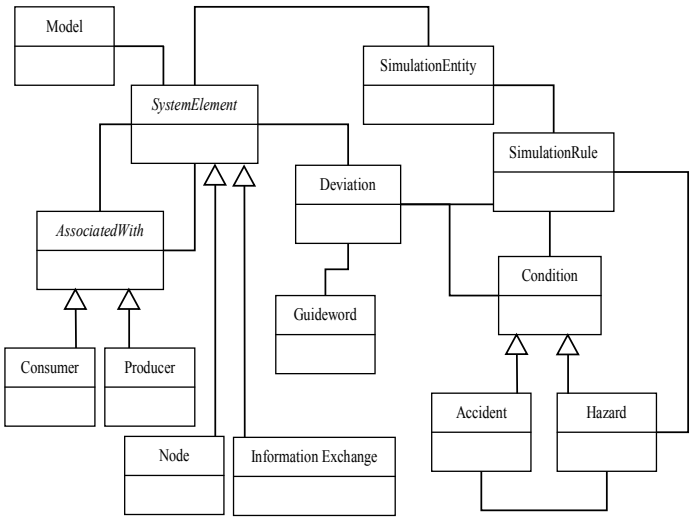

Figure 7. A UML metamodel capturing and associating the concepts of hazard analysis

Usually, most concepts used in hazard analysis methods are not explicitly represented, but are entangled with their graphical representation. A model-driven approach has been

used to define a unifying framework. As part of this, a metamodel was defined capturing the concepts used in the hazard analysis activities, as well as their legitimate associations. Figure 7 illustrates an extract of a (a simplistic to present clarity) metamodel combining DDA and SimHAZAN. The metamodel demonstrates traceability between concepts relevant to different aspects of the system development: system modelling, deviation analysis, SimHazan and hazard analysis. The abstract class *SystemElement* constitutes the super-class that represents any entity of the real system. For example, nodes and information exchanges are subclasses of the class *SystemElement.* All the instances of the *SystemElement* class are associated with a model (e.g. OV-2).

Depending on the level of granularity and the viewpoint of the model, it can be associated with system elements such as software packages, needlines, and data. In a system and particularly a SoS, system elements are related to each other. Nevertheless simply recording the associations is not sufficient as there can be different association types. For example, nodes are connected with information exchanges. Moreover certain nodes are consumers (of information) whereas others are producers. Hence the abstract class *AssociatedWith* has been defined (as an association class) using which the system elements are connected.

In this way a realistic representation of the system elements can be captured and used to trace the effects of each deviation. Deviations are captured by the class *Deviation* and consist of a *Guideword* (suggesting the deviation) and the *SystemElement* being prompted. Similarly, the simulation modeller can use system elements to create simulation entities (*SimulationEntity),* and deviations to create simulation rules (*SimulationRule)*. The SimHAZAN engine then runs the simulations, and thereby identifies (safety related) conditions.

Existence of a rigorous metamodel that captures the concepts of the assurance cases domain can deliver a number of benefits. Most importantly, it offers a common vocabulary and consensus on the concepts related to the domain of application (i.e. safety analysis). This can contribute to avoiding 'deviant' implementations across tools from different vendors. Moreover, evaluation of a tool implementation can be difficult as there is no common point of reference. Also, a uniform serialization format can provide a tool-independent platform to facilitate information exchange between different vendors' tools.

SUMMARY

The term Systems of Systems is increasingly being used to describe a particular class of systems. Systems of Systems demonstrate a combination of characteristics such as dynamic operation, emergent behaviour and complexity. These characteristics provide a number of difficulties to the application of safety analysis and particularly to deductive (safety) reasoning. Deviation analysis is an approach used for exploratory analysis of a system, identifying credible deviations and their effects. Failure maps document the effect chain between safety related (dependability) failures of the various SoS elements.

One challenge in SoS safety analysis is the scale and complexity of these systems, making manual approaches inefficient. SimHAZAN is a semi-automated approach, able to consider a large number of deviations and identify safety related ones. The two approaches are not independent of each other. Instead they use each others artefacts during the lifecycle. A common framework allows the seamless integration of the two approaches, establishing clear traceability between concepts.

ACKNOWLEDGMENTS

Parts of this work were carried out under the High Integrity Real Time Systems Defence and Aerospace Research Partnership (HIRTS DARP), funded by the MoD, DTI and EPSRC. The DARP members were BAE SYSTEMS, Rolls-Royce plc, QinetiQ and the University of York.

REFERENCES

[1] D. Pumfrey, "The Principled Design of Computer System Safety Analyses", PhD Thesis, Department of Computer Science, University of York, 1998.

[2] T. Kletz. "HAZOP and HAZAN. Identifying and Assessing Process Industry Hazards. Institution of Chemical Engineers, 1992.

[3] US Department of Defense. Architecture Framework Working Group "DODAF version 1 – Deskbook", Department of Defence 2004.

[4] G. Despotou, R. Alexander, M. Hall-May. Key Concepts and Characteristics of Systems of Systems (SoS). February 2003. Defence and Aerospace Research Partnership (DARP-HIRTS) Public Document

[5] G. Despotou, "Managing the Evolution of Dependability Cases for Systems of Systems", PhD Thesis, Department of Computer Science, University of York, 2007.

[6] R. Stephans and T. Warner "System Safety Analysis Handbook", 2nd Edition. System Safety Society, 1997.

[7] R. Alexander, "Using Simulation for Systems of Systems Hazard Analysis ", PhD Thesis, Department of Computer Science, University of York, 2007.

[8] M. Hall-May, "Ensuring Safety of Systems of Systems. A Policy Based Approach", PhD Thesis, Department of Computer Science, University of York, 2007

[9] UK Ministry of Defence, "MoD, MODAF Partners. "MODAF Handbook, Technical Specification for MODAF" Ministry of Defence, 2005.