

## Automation of System Safety Analysis: Possibilities and Pitfalls

Andrew Galloway, University of York, Heslington, York YO10 5DD UK

John McDermid, University of York, Heslington, York YO10 5DD UK

John Murdoch, University of York, Heslington, York YO10 5DD UK

David Pumfrey University of York, Heslington, York YO10 5DD UK

Keywords: automated safety analysis, safety analysis tools

### Abstract

As systems become more complex it is progressively less tractable to carry out hazard and safety analyses by hand. Both hazard identification/analysis and confirmatory safety analyses, e.g. FMEA and FTA, present significant (although distinct) problems, and the difficulties are perhaps most intense for software, as this is often where much of the complexity is found. To counter these problems there is growing demand for automation of safety analyses.

Automation presents many possible benefits, but also some drawbacks. It must also be acknowledged that humans are better at some activities, e.g. reasoning inductively, than computers. The practical issue is how to gain benefit from automation whilst reducing risks which could arise from over-reliance on tools.

This paper concentrates on the automation of analyses during system design and development, although many of the observations and conclusions are equally applicable to tools used to manage and analyse safety for in-service systems. The paper discusses some potential risks of, and challenges for, highly automated safety analysis, particularly the effects of removing human skill and experience from the analysis process. It considers:

- how to design tools to identify design weaknesses, and/or produce derived safety requirements (and knowing when they are more or less effective than humans);
- how to facilitate human involvement in automated analyses, recognising that improved understanding of a system is an important output of the safety process.

Underlying this is the question of where it is possible, and appropriate, to “take the human out of the loop”.

### Introduction

A large number of methods and techniques have evolved to support safety analysis work (in this paper, we use the term “safety analysis” to encompass all the hazard identification, risk assessment and safety analysis activities required in a safety critical system development process). Many have emerged from particular industries, reflecting the concerns of different areas of product technology, although some methods have established themselves as useful across many product types.

Unfortunately, most of these methods suffer from a range of negative characteristics. They are often repetitious and tedious to apply and, as system designs become ever more complex, current manual approaches, using predictive techniques such as HAZOP [1] or Functional Failure Analysis [2], and confirmatory analyses such as Fault Trees [3] or Sneak Analysis [4], become increasingly intractable. The time taken to carry out safety analysis becomes a significant drag on project timescales, and the manpower required represents a significant cost, which is often perceived to have little benefit. The problem is worse for software than hydro- and electro-mechanical systems, as it is often the most complex bespoke part of the system, and there is still no consensus on what approaches to software safety are truly effective.

Perhaps most problematic of all, almost all safety analysis is open-loop, in that safety practitioners do not have the satisfaction of producing a working artefact at the end of the process. Indeed, safety engineers may only achieve closure at the end of a system's working life, when it is possible to review its in-service safety record against the predictions and calculations made when it was designed.

Unsurprisingly, therefore, there is a lot of interest in "push button" safety tools, whereby all the necessary analyses could be generated mechanically from system models, and completed products automatically assessed to ensure that they satisfy their safety requirements.

In some areas, automated tools already exist, and several research projects are working to extend the capability of these toolsets. The next sections examine what automation is currently possible, and identify the limitations of current technology, with a particular focus on formal methods and the development of safe software. The paper then considers the strengths and limitations of automation before drawing some conclusions on how to maintain a balance between human involvement and automation in designing the next generation of hazard and safety analysis tools.

### Automation: Principles, Pragmatics and Prospects

The benefits that can be expected from automating safety analyses are initially a result of the speed, consistency, reliability and accuracy of search and analysis algorithms. In the ideal situation, safety analysis would involve so little effort on the part of engineers that it would become feasible to carry out full and detailed analyses of many different design proposals to assist in investigating and selecting between alternatives, and to ensure that analyses were promptly updated after every design change. In the longer term, we would hope to see tools that have the capability to draw conclusions from the analysis, such as identifying key safety risks, or even suggesting improvements to designs.

At present, commercial tools are available to support many safety analysis techniques. However, their capabilities are usually limited to improving consistency in tasks that would be tedious and error-prone to implement manually, especially the application of mathematical methods, the storage, searching and reliable retrieval of safety information, and neatly formatted presentation of results. Tools have traditionally been dedicated to one single analysis method. Tool set integration has tended to focus on the sharing of safety information, for example ensuring the consistency of data used in different views and analyses of an artefact. Some approaches (e.g. the Safety Argument Manager [1]) provide computer-based support for safety argument structures, using these to integrate disparate data and tool-based results.

Few, if any, of the current tools automate the actual analysis; most require the safety engineer to manually identify hazards, failure modes and failure effect propagation paths. In addition, many current tools (and, indeed, analysis methods) require the safety engineer to build a new model of the system (e.g. a functional block diagram or fault tree), and it is in supplying this information that the valuable and creative part of the safety engineer's role lies.

The use of system modelling in safety-related design and fault diagnosis has been a consistent strand of research over many years, involving different research communities. The key issue in automating the actual safety assessment, using models from the core design process, is how to replace the knowledge and insight contributed by the safety engineer. The most common approaches being investigated are either to explicitly extend the system design model to include faults and failures, or to construct a separate database of models of components, including their failure modes, and use a tool to compose these component models to give a complete system model. The disadvantage of the first approach is the complexity (and consequent difficulty of change and maintenance) that it adds to the main system design model; the second approach avoids this, but there are issues of ensuring that the composed model accurately reflects true system behaviour.

Research with roots in the qualitative modelling community [1] has developed tools for the support of FMEA and Sneak Analysis in the domain of low voltage electrical circuits. System models are built like

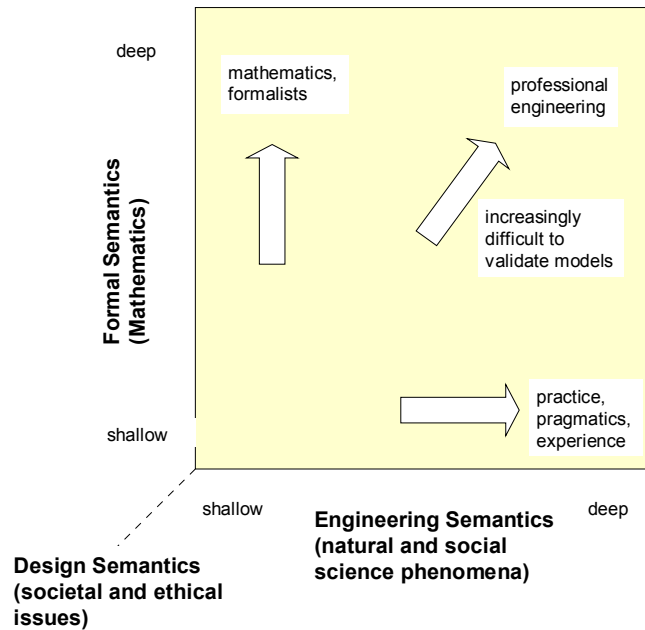
conventional circuit diagrams using components from a library which includes failure mode information, and the tools combine qualitative simulation of the circuit with functional knowledge to associate component failure modes with functional failures. These tools (AutoSteve) are now in use in the automotive industry [2].

Similar work has been pursued by other groups (e.g. [3]). For example, Papadopoulos [4] has developed a MATLAB application which generates fault trees automatically from system block models in which failure modes and the effects of failure modes at block level are specified. An approach to automating the HAZOP technique has been reported [5] using digraph models of a chemical plant and expert system techniques. An ongoing European research programme [] is also investigating the automated production of safety assessment from system models developed in a range of tools including MATLAB and StateMate.

A further strand of research, pursued in the software engineering community, focuses on the use of formal modelling, where ‘formal’ means a representation based in mathematics, and amenable to the construction of mathematically rigorous proofs. This approach depends on there being a common mathematical framework within which the specification (problem) and the designed artefact (solution) can both be expressed. Thus it is feasible to use discrete mathematics to formally prove that a software component is consistent with its specification, because of the logical nature of software.

Constructing a formal model often leads to greater understanding and precision of thought about a design or analysis problem, supports collaborative work (if understood by co-workers) and opens up the possibility of mathematical techniques being deployed to solve the problem. There are limitations to mathematically formal approaches at the ‘boundaries’ of these models, particularly at the interfaces with actuators and sensors (assuming a control application) and at the implementation of code in physical devices. However, work in this field is primarily concerned with fundamental questions about the role of mathematical formalisms in product development, questions which lie at the heart of analysis automation.

As Figure 1 illustrates, the formal semantics of a model are different from its “engineering semantics”, by which we mean the correspondence between the model and the real physical or social phenomena that are being represented. Extending the depth and scope of a formal model increases the difficulty of model validation, and it becomes progressively more difficult to develop trust in more complex formal models. This is the classic concern of applied mathematics (e.g. [4]). Recent work in the software requirements field [5] has called this the ‘designation problem’ and developed a systematic approach to recording the correspondences between formal variables and informal definitions.



**Figure 1 Three areas of meaning in engineering**

The application of sophisticated mathematical techniques to design and analysis problems in such a way that deep engineering semantics are preserved is the hallmark of professional engineering. In traditional engineering fields, the development of trust in modelling approaches and tools is developed within a specialty over many years. Model validation is achieved gradually by repeated comparison between model predictions and observation. This approach is the basis of the learning loop model discussed later in this paper.

A third area of meaning in engineering relates to the societal/ethical issues in product development; safety engineering, in common with other specialties, has its roots in ethical value judgments. This aspect is not considered further in this paper.

In the following sections, the prospects for automation of safety analysis are explored. We treat systems and software separately because the issues are slightly different in the two domains.

#### Automation in systems safety analysis

The systems engineering field is characterized by having to manage disparate technologies and engineering specialties. System models are traditionally used to express designs and support trade off studies between alternative architectures. Two different styles of model are usually identified; (1) models of specific properties or views of a system e.g. reliability block diagrams, control loop models, and (2) integrating models, which represent global data about the design and ensure consistency between different views.

The use of these models for safety analysis presents precisely the potential benefits and drawbacks discussed above. If the design models are developed as part of the core design effort (e.g. using CAD systems), additional safety-related modelling costs are minimised. There is potential for systematic and complete exploration of the safety-related behaviour of the system as designed, and of the consequences of specified failure modes (or types of failure mode). The analysis results will be consistent and complete within the scope of the product models and failure modes represented. However, it is easier to base safety analyses upon the specific property (type 1) models than the integrating models (type 2), and there is

again the risk that important safety properties will be missed because they are (cannot be) represented in the model.

We can distinguish three ‘boundaries’ (sets of assumptions) about the scope of a system model and the associated failure modes:

1. assumptions about component failure modes. For example, the possibility of components failing in ways that have not previously been encountered may not be considered.
2. assumptions that the effects of component failures fall within the range of phenomena covered by the product model. For example, a product model based solely on electrical circuit theory would not capture the consequence of a component failure (such as overheating) that lies outside the electrical circuit domain.
3. assumptions about the safety implications for the total product of failure effects identified at the interface of the modelled system.

These assumptions are related to the engineering semantics of the models.

A further general concern is the additional complexity that arises in dynamic systems, that is, systems which cannot be adequately represented by static structures. For example, finite state machines are widely used to model the switching behaviour of complex components. However, the behaviours exhibited by systems with dynamic feedback require different techniques. The field of systems engineering has developed the concept of 'emergent property' to describe the non-obvious behaviour of a system which arises as a result of the interaction of its components. It seems necessary for the safety engineer to assess different levels of aggregation of a system in order to uncover 'emergent' failure modes, that is failure modes which would not be exposed by considering component failure modes and their effects propagated within models of design intent.

#### Automation in software safety analysis

Formal representations are relatively more powerful in software engineering because of the abstract nature of software. Many discrete formalisms have been investigated to support software development including set theory, graph theory and group theory. Continuous formalisms found in other branches of engineering (differential calculus) have been approximated to with discrete formalisms.

Software failures are systematic. Assuming the integrity of the hardware: in a deterministic system, the probability of a hazardous response is either zero — if the software has been designed to avoid it — or equivalent to the probability of the environmental conditions arising under which the software exhibits it.

Thus, if our probabilistic causal model factorises out probabilities concerning the hardware on which the software runs, and the integrity of the process (analysis tools, compilers correct etc, product model and source code consistent etc) then one possible use of formalism is to predict whether the software exhibits a hazard. If so, an even more valuable use of formalism might be to predict which environmental conditions give rise to the software exhibiting such a hazard.

An example of this kind of use of formalism is the use of “Formal Methods” such as Z [ref], RTL [ref], VDM[ref] or CCS[ref] to show that a particular safety property (the negation of a hazard) is guaranteed by a system specification or design.

Formal descriptions of products and safety properties may also be used to derive test cases. However, testing cannot prove the absence of errors and placing an accurate probabilistic estimate failure rates based on testing can be problematic.

There are significant practical problems associated with formalising hazards so that they can be analysed. Different formal methods emphasise different views of the system. For example, model-based methods such as Z, VDM emphasise the data and step operations on that data that the system make take; RTL emphasises the valid sequences of data states that system may evolve through; CCS distinguishes the patterns of communication a distributed system may exhibit. Each method comes with its own style of

reasoning and effectively sets the agenda for the kinds of safety properties which may be analysed. These may not necessarily be the kinds of safety properties the safety engineer wishes to express.

For example, a product model written in Z might describe the step-wise ways in which the data of the system can be altered, in terms of a set of inputs and producing a set of outputs. The kinds of reasoning classically available to the practitioner allows them to check whether an invariant holds over the system state, or perhaps to show whether an operation is a refinement of some abstract “safe” operation. However, the safety engineer might be interested in ensuring that a particular differential relationship on the system outputs holds. Moreover, they might want to derive the differential constraint on the input variables required to guarantee some differential constraint in the outputs. It is not easy to see how to do this given the classical application of the method.

Even if the engineer can express the safety property of interest within the chosen formalism, there are limitations to what can be achieved with the product models themselves. Safety critical systems are often complex systems. They can involve data structures, operations on those structures which must be implemented in code, asynchrony and communication, and real-time constraints. The older formal methods tend to focus on one particular viewpoint making representation and reasoning about complex behaviour extremely difficult. There are two extremes:

- A formal model is constructed which tries to encompass the full complexity of the system;
- Several formal models are constructed, each representing different facets of the system.

The first approach’s main drawback is the lack of precedent. There are not many examples in the literature of formal models embracing many different viewpoints (e.g. time, data, communication). Another problem could well be the tractability of the resulting model when it comes to analysis. A complex model will almost certainly involve a large state-space, probably too large for conventional model-checking limitations. If theorem proving is preferred, there is still likely to be a large axiom base describing the behaviour of the system under each combination of viewpoints. This may make the already expensive theorem proving approach infeasible.

The second approach also has its drawbacks. Firstly, different viewpoints are not necessarily independent of each other. For instance, communication patterns can depend on the data in the system, or the time taken for some event to occur. One must be careful not to “over-abstract” when producing different formal viewpoints of the system, as the resulting model may not contain the information necessary to verify a safety property. Secondly, there is a rather extreme validation problem. How does one know that each formal model faithfully captures some abstraction of the system — especially when we do not understand what that abstraction is in formal terms. Indeed, when each model is not necessarily based on the same formalism, how can we even be sure the models are consistent with each other?

#### Automation: challenges and potential pitfalls:

The automated safety analysis of a design has to be based on two sources of explicit data; (a) a design model or description, held in computer-processable form, and (b) data and algorithms which can be applied to the design data and which implement a safety analysis. The quality and completeness of the safety analysis is limited by the quality of the design data and of the formalised safety information. We summarise this situation by characterising the design and tool data as existing in a *virtual* world; validating such models against *real* world phenomena is a key responsibility of the safety engineer.

Validating virtual world models and tools against real phenomena is problematic because the real world is informal; there is no one-off test we can conduct to demonstrate ‘correctness’. Instead, trust has to be established over time, with repeated use and demonstration of consistency between modelled and real world outcomes. Trust is easier to establish in generic applications, where standardisation and wide use establishes trust within a community of practitioners. It is more difficult to build trust in specific tools and

models. The more formally complex (the deeper the formal semantics) embodied in a tool, the more difficult is the validation task against real world phenomena. The rate of change in an application domain compared with the rate of building of trust in a model is a factor in the usability of automation in a particular setting. A safety engineer may trust his/her engineering judgement (based on 'engineering semantics' more readily than a tool which, although formal and rigorous, is less familiar and where the process by which results are generated is unclear. Visibility of (and participation in) tool development are perhaps the best ways to build trust, but are rarely practical strategies.

Tools cannot at present help with "fundamental safety" – what we might call safety by design. The system model, which is formally represented and subject to automated analysis, is an input. To consider automating design decisions as a result of safety analysis raises many issues, and is not feasible today.

Automating an analysis does not remove problems which often arise in the application of tools. Typical adverse behaviours include:

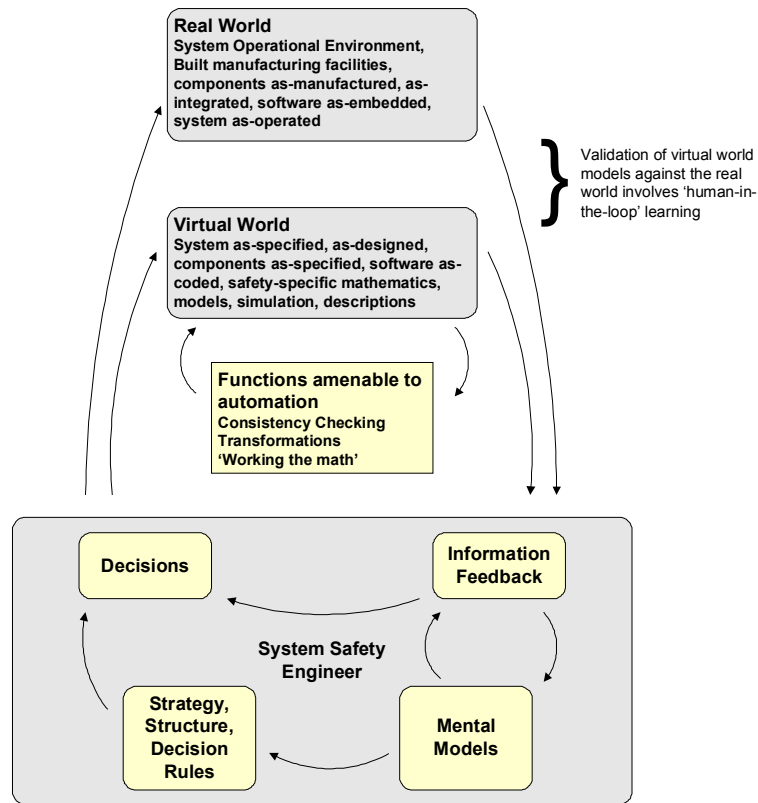
- optimisation for "best" result from tool, rather than real safety;
- use of tools just to be seen to be generating numbers, or to comply with externally imposed requirements;
- inappropriate use and/or belief in the outputs of tools;
- over-reliance on tools and tendency not to assess the gaps and joins between formally supported aspects of a design.

A further problem with analysis tools is the generation of results which are difficult to interpret, possibly due to the volume of data generated and a lack of prioritisation of importance or significance. Tools which assimilate data, highlight significant features and offer conclusions would be desirable provided the trust and validation concerns can be satisfied.

The demand for increased levels of automation in safety analysis (as in other areas of knowledge work) can be interpreted in several ways. If the motivation is one only of efficiency gains and short term cost reductions, we must be wary. Business managers may be placing the interests of the shareholder over those of the at-risk user of the system product. Tools vendors are interested in sustaining profitable businesses and do not carry responsibility for the appropriate use of their products.

#### Automation: A strategy

A strategy based on the concept of *professional practice* is proposed to assess, manage and deploy automation in safety analysis. We view tools, including automated analyses, as extensions of our human capability. After all, we develop them to meet our needs and have to take responsibility for the results of their use. In this sense, taking "humans out of the loop" is not an option. On the contrary, we need strengthened professional practice and regulatory oversight to deploy automation appropriately.



**Figure 2 Double-loop learning model, adapted from [1]**

Figure 2 shows a simple model of a system safety engineer interacting with a product domain. We have borrowed the double-loop learning model from [3], a model which in its turn was informed by feedback control systems thinking. The model has found wide acceptance in the social and management sciences; the version used here has been especially influenced by Sterman's [1] application in the system dynamics field. Figure 2 depicts professional practice as a continuous process of interaction with real world problems, developing understanding and the exercising of judgment in making decisions about how to affect the real world. The real world includes the actual system products after they have been manufactured and integrated and then operated. This is the central concern of safety analysis – the risks arising from unintended behaviour of the system in its operating environment. In 'single loop' learning, the affects of the decisions made are sensed and generate further decisions and understanding.

In engineering design (as in many other fields) we construct models, descriptions, mathematical theories and other 'virtual world' constructs to support our professional practice. A system safety engineer will construct safety-specific models and make use of received design models and subject them to analysis. The block labelled 'Functions amenable to automation' indicates that, once formalised data has been constructed, we can make use of automated tools to process it. However, the interpretation and use of the output in decision-making about the real world involves the safety engineer.

Also shown in figure 2 are the *mental models* which practitioners develop in the course of their working lives. These models play a crucial role in judgment and decision-making. The double-loop learning model allows for continuous learning in the mental models, changes which can arise from interaction with both the real and virtual worlds.

A central concern of safety engineering is the degree to which the virtual world models correspond with the real world. For example, the functional and architectural block diagrams used extensively in system

engineering are virtual world constructs and the degree to which as-manufactured and as-operated systems conform with the interfaces and interactions implied by such models is subject to safety assessment. When automating safety analysis, we must be aware of the assumptions being made by the tools and the aspects of the design which are not actually being challenged by them. For example, in aggregating failure effects, a tool may assume that as-designed interfaces successfully contain and channel failure effect propagation mechanisms.

With this (simplified) model of professional practice, we can develop a strategy for selecting and deploying automation in ways which strengthen professional practice in the safety field. In considering tools and automation, we need to:

- enable assessment of the correspondence between the virtual world model and the real world;
- ensure that models and analysis techniques are understandable to the practitioner, enabling trust to be established over timescales which match the rate of change of the application domain problem;
- support double-loop learning on the part of practitioners; i.e. avoid separating practitioners from models of the system, for example, which might damage the accumulation of understanding.

The power and efficiency gains offered by automation are recognised and welcomed, but real world engagement remains the key concern of the safety engineer, rather than automation inside formal worlds. Given the interpretive context within which tools are used, 'fine-grained' models, applied flexibly and which are easier to validate, may be more appropriate than more complex, formal approaches, for many applications.

Viewing automation under the 'virtual world' category encourages consideration of other approaches to description and modelling. Descriptive, informal and semi-structured models are very important and can be stored, shared and manipulated to varying degrees by computers. Information systems which store and communicate 'lessons learnt' are one example of the use of database tools in the safety field. Integration of information-based approaches with mathematically formal models appears an interesting area; validation of stored information and 'fitness for purpose' remain concerns and demand involvement of the professional practitioner.

The model of figure 2 encourages a time-based view in considering automation. It is not just a question of whether a model can be built and analysis algorithms developed to process the design model. We have also to consider the time required to validate. A range of criteria for choosing an appropriate level of automation can be identified (and specialised to a particular domain), for example:

- degree to which safety properties of real world domain can be formalised;
- single domain or cross-disciplinary, multi-component domain;
- the severity of the hazards being considered;
- rate of change of domain in terms of a project and product line/ industrial sector;
- generic characteristics which are amenable to general purpose tools, or product specific;
- level of precedence of the design domain, the degree to which failure modes are known and the physics of failures understood;
- degree of complexity and subtlety of the system.

Such criteria amount to a risk assessment of the automation deployment. The cost and resource requirements associated with deployment have to be justified against potential benefits.

#### Potential Solutions in the Software Domain

The safety analysis of software, as in other domains, depends on the product design and implementation descriptions that are available. The abstract nature of software results in formal mathematical descriptions

being able to represent most of what a software component does, at least within the boundaries discussed above. Research directed at placing software development on a more formal foundation also serves the interests of the safety engineer.

Recent work on the application of formal approaches to the development of embedded control software [ ] has indicated a practical approach to deploying automation in this domain. A *domain specific* approach is being developed in which models of systems and safety properties are developed which are understandable to practitioners and to which underlying formal theories are mapped. This approach can be characterised as establishing mappings between the formal and engineering semantics of figure 1, approaching from the formal side, as it were.

This approach to formalising software development echoes the traditional use of applied mathematics in engineering. The domain specific language can be succinct, specific and unambiguous. It need only be rich enough to describe the kinds of designs common to the domain. For example, in embedded control systems, there may be no need to specify complex relationships between stored information structures. Domain specific abstractions can be developed to support reasoning about safety properties.

The choice of underlying formal theory depends on the domain. The Integrated Formal Method approach [IFM refs] is a promising one being pursued by the formal methods research community. Such an approach supports different views of a software system, exploits the mathematics associated with different formalisms and offers the possibility of an integrated representation designed to support specific analyses. One example, based on integrated process algebras [ ], uses symbolic operational semantics [ ] to temporarily suppress process inter-communication aspects of a design, when reasoning about data constraints.

Once a sufficiently rich underlying formalism has been identified, the development of the domain specific elements of the approach should be an on-going effort — an exercise in process improvement. This corresponds with the iterative validation against real world phenomena described in figure 2 above. The effectiveness of domain specific notations would be reviewed constantly, especially with a view to adding succinct notation for regularly occurring structures and properties. This approach is feasible if the underlying formalism is stable, or only changes slowly, which should be the case in a particular embedded software domain.

### Conclusions

The deployment of automation in safety analysis has been considered, focusing on the automatic analysis of product designs to derive safety properties. A general framework, with professional practice as the central concern, has been used to consider the advantages and pitfalls of automation. The possible gains include greater efficiency in the conduct of analyses, more systematic and complete application of techniques and analyses independent of particular interests or biases.

However, in the spirit of safety engineering, we must also be sensitive to the potential shortcomings of automated analyses. The validation of the system models and analytical methods embodied in tools is a major concern. Because real world phenomena are informal, ‘virtual world’ models cannot be tested in a once-and-for-all manner against real physical systems. Instead, trust has to be developed over time and by a community of users. It has been argued that the applicability of automation depends on:

- the generality/ specificity of the models used;
- the rate of change of the modelled domain and the time needed to build confidence in a system model;
- the feasibility of validating the safety-related assumptions in tools (e.g. postulated failure modes and failure effect propagation mechanisms) as well as the models of intended design.

Issues in automation at system and software levels have been discussed. In summary, even with extended tool integration and automatic application to design models, the safety engineer remains responsible for the outputs of the analysis and has to interpret them in the context of a wider safety analysis process. Such tools effectively hold and re-apply past knowledge of product designs and component failure modes. The appropriateness of such knowledge for a current project is a 'fitness-for-purpose' decision which the safety engineer must make.

#### References

1. Price, C.J., *Function Directed Electrical Design Analysis*. Artificial Intelligence in Engineering, 1998. **12**(4): p. 445-456.
2. AutoSteve, <http://www.firstearth.co.uk/index.php>
3. Eubanks, C.F., S. Kmenta, and K. Ishii. *Advanced Failure Modes and Effects Analysis using Behaviour Modeling*. in *ASME Design Engineering Technical Conference*. 1997. Sacramento, CA.
4. Papadopoulos, Y. and J.A. McDermid. *Extending and Automating Classical Safety Analyses to Rationalise and Simplify Complex Safety Arguments*. in *Design Methods and Tools for Dependable Systems and Quality of Service (Proceedings of 10th European Workshop on Dependable Systems)*. 1999. Vienna: Austrian Computer Society.
5. Vaidyanathan, R. and V. Venkatasubramanian, *Digraph-based models for automated HAZOP analysis*. Reliability Engineering and System Safety, 1995. **50**: p. 33-49.

#### Biographies

A. Galloway, Department of Computer Science, University of York, York, YO10 5DD, UK, telephone - +44 (0) 1904 432778, email - Andrew.Galloway@cs.york.ac.uk.

Dr. Galloway has spent almost a decade researching and teaching applied formal methods. He has developed practical approaches to formal specification which underpin work on modelling and verification. He is a member of the programme committee for numerous international conferences, including B (ZB conference), IFM (Integrated Formal Methods), APSEC (Asia Pacific Software Engineering Conference) and ICFEM (International Conference of Formal Engineering Methods).

J. A. McDermid, PhD., Professor of Software Engineering, Department of Computer Science, University of York, York, YO10 5DD, UK, telephone - +44(0)1904 432726, facsimile - +44(0)1904 432708 e-mail - john.mcdermid@cs.york.ac.uk.

John McDermid has been Professor of Software Engineering at the University of York since 1987 where he runs the high integrity systems engineering (HISE) research group. HISE studies a broad range of issues in systems, software and safety engineering, and works closely with the UK aerospace industry. Professor McDermid is the Director of the Rolls-Royce funded University Technology Centre (UTC) in Systems and Software Engineering and the BAE SYSTEMS-funded Dependable Computing System Centre (DCSC). He is author or editor of 6 books, and has published about 250 papers.

J. Murdoch, Department of Computer Science, University of York, York, YO10 5DD, UK, telephone - +44(0) 1904 433375, email - John.Murdoch@cs.york.ac.uk.

Dr. Murdoch has worked for over fifteen years in the aerospace industry in a variety of specialist, systems engineering and R& D roles. His current work with the UTC is in the fields of systems engineering, product and process modelling and safety engineering for high integrity systems. His research is conducted with Rolls-Royce plc, and has the objective of supporting continuous improvement in product design and development processes. Products of interest are the electronic controllers of aero engines, with

attention directed towards the multi-disciplinary 'total system' view. Coherent design of systems comprising hardware (electro-mechanical, hydraulic etc), digital electronic hardware and software components is his central concern.

D. Pumfrey, Department of Computer Science, University of York, York, YO10 5DD, UK, telephone - +44(0) 1904 432735, email - David.Pumfrey@cs.york.ac.uk.

Dr. Pumfrey is a Teaching and Research Fellow in System Safety Engineering. His research interests include the adaptation of classical safety analysis techniques for use on computer and software systems, low-level safety analysis of the hardware / software interface, and improving the way in which systems engineers and software developers work together on safety critical systems projects. He has been involved in setting up MSc and Certificate courses in Safety Critical Systems Engineering, as well as presenting many on-site courses in system safety for companies such as BAE SYSTEMS, Rolls-Royce and TRW Aerospace.