More Automated Formal Methods?! If so, why, where & how?

Position Paper towards a Contributed Talk at AFM 2017 workshop associated with NFM 2017

A. Chakrapani Rao
Warwick Manufacturing Group
(WMG), University of Warwick,
Coventry CV4 7AL
UK
A.Chakrapani-Rao.1@warwick.ac.uk

ABSTRACT

Formal Methods (FM) have been around for decades and many have been improving all the time. Automated formal methods techniques and tools have been making a mark in real world applications across industry domains. So, why and where do we need more automation? How much automation is required? This paper attempts to cover an assessment on where I believe we are, based on my own not-so-limited but diverse enough experience in automated industry-strength formal methods and model based systems engineering area, where we might need to get to and possibly how. Key characteristics of future automation needed for success are outlined. No attempt is made to be exhaustive as the world of FMs is vast and the collective work of FM expert researchers, developers and users is needed for exploration – "to boldly meet future challenges which no FM has ever met before"!

CCS CONCEPTS

• Computing methodologies~Model verification and validation • Software and its engineering~Model-driven software engineering • Software and its engineering~Software verification.

KEYWORDS

Automated, Interactive, Connected, Formal Methods, Systems Engineering

ACM Reference format:

A. Chakrapani Rao. 2017. SIG Proceedings Paper in Word Format. In *Proceedings of AFM 2017, CA USA, May 2017 (AFM'17)*, 8 pages.
DOI:

INTRODUCTION

Systems and software engineering have undergone an enormous transformation over the last several years. Most systems and software developed today have been designed, developed and tested using a variety of modelling languages, programming languages and tools. The systems and software engineering process models (eg., V-model, Agile) may also be quite different depending on the industry domain, the size and nature of the system or software, the particular company's standards and standards pertinent to the domain itself. Automation in the overall systems engineering process, for example in automatic code generation or in automated testing potentially bring enormous benefits especially in terms of efficiency, consistency and objectivity.

Figure 1 shows different phases of Systems Engineering within the common process model, the V-model. The phases represent distinct abstraction levels. Of course, within each phase, a model may undergo refinements to result in a more detailed model for that phase. Irrespective of the process model, each phase, within a Model Based Design (MBD) workflow, consists of either a model or, in reality, many models. For instance, the Requirements phase may have requirements documents, DOORS database of requirements and/or a SysML model. They are sometimes termed the "Descriptive Model" [14]. Let's refer to the requirements documents as the Informal Model (IM) and the SysML model as the Semiformal Model (SM). If we develop the requirements in a formal language or provide precise semantics to a SysML model, we could term it as the Formal Model (*FM*). In a similar way, we could have different models at the Design phase etc. In other words, a Phase_i model could have one or more IM(s), SM(s) and/or FM(s). Of course, in the context of systems engineering of complex systems and system of systems, all models have relevance. Whilst an IM may largely be based on natural languages, an SM may be visual and hence more suitable for

communication amongst various stakeholders, despite not being fully formal.

For a seamless Model Based Systems Engineering, there is a need for relationships between these models and a level of integration. We can define and refer to them as: $IM_{ip(n)} \Leftrightarrow IM_{jp(n)}$, $IM_{ip(n)} \Leftrightarrow FM_{ip(m)}$, $FM_{ip(n)} \Leftrightarrow FM_{jp(n)}$ and so on where i) the first subscript i, j etc. stand for the model number and ii) p(n) with n=1,2 etc. denotes the phase number. This integration could be a connection between model elements implemented as traceability links or a tighter connection between parameters or model attributes across models in different phases. Without such relationships including potentially a tight integration, there are likely to be problems in any analysis based on models which are all used towards the final product and potentially in claiming certification credits as per the standards across different domains. For example, a formal analysis based on an incorrect FM derived from an IM would be a costly effort with practically no benefits.

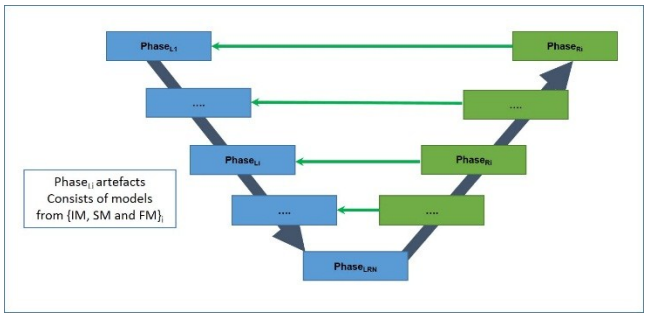

Figure 1: Models in Systems Engineering

Figure 2 depicts the many different models that may be part of any phase in the systems engineering process.

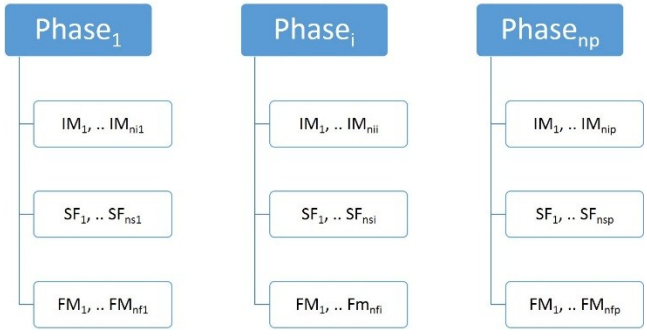

Figure 2: Phases and models

Figure 3 depicts the fact that there may be implicit or explicit connections between the different models. In a typical large scale project, within the industry, models may have been created in such a way that there are inconsistencies between them!

Hence, the success of automated formal methods techniques and tools depends, in my opinion, on the following key factors:

1. How well connected they are to the other models (whether other FMs or SMs or IMs) in the overall process?
   a. Let's call it as the *Connectivity* property. This "connection" may take one or more forms, as appropriate; for example, simple "traceability links" between model elements across phases or an advanced level model integration connecting system-level attributes from a system requirements (descriptive) model to appropriate parameters in an analysis model [9, 14].
   b. We may optionally grade it on a scale from Level 0 to Level 3 where Level 0 is *Totally Unconnected* and Level 3 is *Seamlessly Connected*.

2. How well automated are the formal methods techniques and tools?
   a. Let's call it as the *Automation* property.
   b. We may optionally grade it on a scale from Level 0 to Level 3 where Level 0 is *Fully Manual* and Level 3 is *Fully Automated.*

3. How synergistic are they to cater to the varied needs of the users?
   a. Let's call it as the *Synergy* property.
   b. We may optionally grade it on a scale from Level 0 to Level 3 where Level 0 is *No User Interaction* and Level 3 is *Fully Synergistic.*

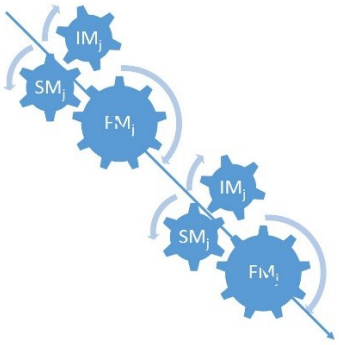

Figure 3: Model Connections within and across Phases or Abstraction Levels

The above points will be illustrated with some examples, from my experience within a few projects [1-10, 18, 24], in the next section.

Automation with formal methods as well as systems engineering will benefit from improvements and/or new developments in the following areas:

1. **Systems Engineering**

   i) Traceability between different model elements, for example a set of requirements in an informal requirement model (i.e, the IM) to a relevant part of the design in a semi-formal design model (i.e., the SM).
   ii) Automatic generation of an analysis model, eg., a system-level simulation model (i.e., SM), from a descriptive model, eg., a system-level model in SysML (i.e., SM).
   iii) Automatic code generation (say C code i.e., SM) from a design model, say a Statechart based model (i.e, SM), such that it is correct-by-construction.
   iv) Automatic generation of information to support change management when one model in a linked set of models changes.
   v) Automatic checking of models for specific modelling guidelines, eg., MISRA generic modelling design and style guidelines.

2. **Formal Methods**

   i) Automatic generation of specification properties from pre-defined patterns in a natural language.
   ii) Automatic formal verification of real-life design level models, via model-checking.
   iii) Automatic checking of healthiness conditions of software, eg., run-time errors like divide-by-zero, via static checking.
   iv) Automatic theorem proving based on tool-in-built proof strategies captured from experts.
   v) Automatic refinement of models, eg., abstract specification in ITL, towards an executable model, e.g., an executable Tempura program.
   vi) Automatic generation of test cases from a model, eg., TargetLink model, to support testing at different levels of abstraction like Model-in-the-Loop and Software-in-the-Loop.
   vii) Automatic decomposition of a large model into sub-models together with top-level property into sub-properties for compositional verification.
   viii) Automated tools to assist in education and training in the area of formal methods.
   ix) Automation in the integration of different complimentary formal techniques and tools, e.g., model-checking and theorem-proving, to address the scalability issues in model-checkers.
   x) Automatic synthesis of a design model from formal specification.

The above list may need to be expanded, of course, and by 'automatic', full automation is not implied. The level of automation may need to be appropriate to the specific process with the user kept in the loop, as necessary, for confidence not only in the techniques and tools but also in the overall process and the resulting artefacts. In addition, effective human and machine collaboration may be needed for proofs due to incompleteness and undecidability issues [12].

There are alternative possibilities in terms of further approach to automation: i) focus on the FMs and develop it as an area where all phases of the software-intensive system development lifecycle are addressed via FMs and ii) involve the FM within an overall process consisting of IMs, SMs and FMs as appropriate. The other alternative of not involving any FMs at all may still be suitable depending on the size and nature of the application but even in such cases, due to the need for complimentary techniques to improve product quality, the business case for some FMs could indeed be favorable.

There is significant literature in terms of automation within a specific formal analysis technique, for example model refinement or parallelization of model-checking algorithms. Recent literature on automated analysis has been discussed in [13] and covers parallelization of model-checking algorithms, SAT and SMT solving, runtime verification (for verifying single traces as opposed to full models) and probabilistic systems' verification. Automated controller synthesis from formal specifications is briefly covered in [7].

HOW CONNECTED, AUTOMATED AND FLEXIBLE

The following are some examples, from personal experiences in automated formal methods techniques and tools. They, i.e., the then current state-of-the-art with those techniques and tools, are used to illustrate their key connectivity, automation and/or synergy properties. The purpose of these examples are merely to propose a potential initial approach to assess the current state with automation and plan future strategies for further automation.

EXAMPLE 1 – Formal Requirements Analysis

In a recent project I was part of [1, 2], there was a need to integrate existing requirements engineering methods and tools with new formal methods based techniques and tools [19] for requirements analysis. Whilst the existing requirements were informal and semi-formal, the new formal tools had the capability to analyze requirements for consistency and completeness. They incorporated formal techniques like SAT solving. However, it was

unrealistic to expect large teams to start modelling requirements using the new language involved, although based on natural language. Hence, there was a need to bridge the gap and evolve towards an integrated approach providing such advanced requirements analyses. Due to this necessity in an industrial environment, the team developed systems and methods for integrating current state-of-the-art with newly developed internal techniques and tools. The new implementation automated the generation of suitable requirements artefacts not only to assist the formal requirements modelling and analysis activities but also to support regular requirements reviews carried out by relevant domain experts. The purpose of this example is to illustrate the needs of integrating formal methods techniques and tools to the current state of the art, ie., FM⇔IMs, rather than developing them as a totally niche area.

The accomplishments of this project could be classed roughly ("roughly" because, in my opinion, this is only for illustration) as Connectivity Level 1, Automation Level 1 and Synergy Level 1. To achieve a higher level of connectivity, automation and synergy, there is need for going beyond proof-of-concept and pilot projects by involving engineers, developers and researchers to take the technology to higher Technology Readiness Levels (TRLs) through technology transfer efforts.

Key features of the overall automated formal methods relating to this example: Connection to informal models; automatic generation of semi-formal and formal models; requirements-based simulation.

EXAMPLE 2 – (Design-level, Formal) Model Checking

In another industrial R&D project [18], one of our industrial partners had a need to formally verify design models created in MATLAB/Simulink and targeted for production code generation using the TargetLink code generator. The team integrated formal methods techniques and tools to this specific MBD environment to create a prototype tool that was subsequently evaluated across industry domains. Eventually, a commercial product was born that is considered a pioneer in the formal verification area involving Simulink/Stateflow models especially developed for production code generation using dSPACE TargetLink. Several core algorithms, techniques and tools evolved from other projects including [24].

This example demonstrates the connectivity property due to the model-checking techniques being integrated to the state-of-the-art modelling environment across industry domains. The fact that model-checking techniques provide counter-examples to assist the user in understanding why a property failed demonstrates a high level of synergy. Model-checking requires only little intervention by users mainly for providing relevant settings for the algorithm and hence the overall technique is highly automated.

Today, there are several more commercial tools for example the Simulink Design Verifier, for Simulink/Stateflow and SCADE Design Verifier, for SCADE models. The users have a choice in terms of not only their MBD tools but also how they choose to do their formal verification project activities. They are all examples of integration of FM⇔SMs. Within the core of the formal methods techniques involved, there are different specific techniques such as Bounded Model Checking, Complete Model Checking and SAT Solving which are all brought to the user in a convenient manner. It is worth stating that the Simulink models were once verified by translating them to SCADE models [11]. In this case, accurate translation of a Simulink/Stateflow model to a SCADE model was important as was automation. Automated formal methods techniques and tools are now being increasingly evaluated and used within the industry for many use cases.

Although model-checking is highly automated, there can be several challenges with respect to i) using the requirements and formalizing them as appropriate properties i.e,, functional, temporal, reachability in terms of graphical states or variable values, ii) preparing the models for formal analysis by taking care of any unsupported blocks and datatypes and iii) interpreting the results from the model-checker to carry out any further analysis.

The accomplishments of this Example 2 project could be classed "roughly" as Connectivity Level 2, Automation Level 2 and Synergy Level 2.

Key features of the overall automated formal methods relating to this and other similar examples: Connection to semi-formal models; automatic generation of formal models (so that the users are transparent to formal methods); scenario for reachability analysis (with a range of choices of selection of states in the model – hence involving the user); automatic generation of counter-example, if any; providing corresponding test harness for simulation.

EXAMPLE 3 – Formal Property Specification

Requirements specification has been a hard problem not only in the formal domain but also in the informal! It is non-trivial to write numerous requirements, typically in thousands, that are free from ambiguities, inconsistencies and incompleteness when they are also written by several diverse stakeholders with inherent natural language issues not making it any easier for them. In terms of specifying properties (based on requirements), especially involving temporal logic, it becomes even harder. In the case of the formal verification project for ML/SL/TL mentioned in Example 2, a pattern language was defined and implemented based on related work on Symbolic Timing Diagrams (STD) [16, 22]. The pattern provided a template to capture properties by stating i) whether the property had to be satisfied after an optional

"N" steps (of execution of the model), immediately or after reaching a particular condition "R", ii) whether the property was an invariant, iterative or whether it had to be satisfied initially once, and iii) what the mapping of given terms in the pattern, for example "P" and "Q" were to model variables and values. An example pattern in shown in Figure 4 indicating its usage below the graphical illustrations. The set of patterns developed was well received by customers and similar approaches are noticeable or emerging in other tools as well. However, this by no means is a conquered area and still poses challenges which requires further research, automation and training! Nevertheless, this effort, across the research and development community, shows how the formal methods and tools can be in synergy with users' necessities, especially convenience, ease of use of formal notations and languages. In a large team environment, possibly across global locations, languages and cultures, an easy-to-use visual pattern language and tool support helps. Appropriate visual descriptions to explain the pattern are a bonus in addition to the automation involved. Such efforts are crucial to make the formal techniques and tools accessible to the current state-of-the-art and a wide spectrum of users.

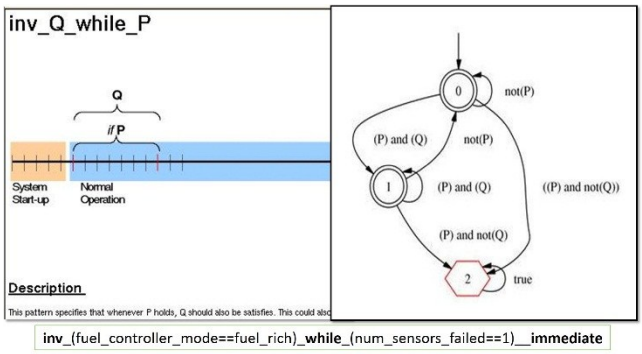

Figure 4: An example pattern with visual descriptions

The accomplishments of this project could be classed "roughly" as Connectivity Level 2, Automation Level 2 and Synergy Level 2.

Key features of the overall automated formal methods relating to this and other similar examples: Connection to informal models; abstraction of temporal logic details from the user; automatic generation of formal models.

EXAMPLE 4 – (Formal, Code Level) Static Analysis

In a project based on static checking [4], a key part of the research was to evaluate tools in terms of how "sound" and "accurate" they were in detecting defects. Soundness property of the technique ensures that there are no false negatives whereas a high level of accuracy ensures that false positives are kept to a minimum. This is analogous to other industries, such as the medical domain, where testing techniques have not only got to be sound but also accurate and affordable by all or a vast majority of people. Despite a technique having some inherent shortcomings, it may still have its place for quick checks before users going on to more powerful and/or other complimentary analyses. Although the formal methods automation level can be considered high here, there is a need to provide the user more assistance in making sense of the results much the same as a doctor needs to quickly analyze the results and suggest a particular treatment to the patient quickly enough! For instance, any false positives would need adequate tool assistance and useful guidance to resolve them.

The tools were evaluated using a diagnostic test-suite consisting of various C programs both from the public domain as well as some internal code. Apart from soundness and accuracy, a number of other criteria were considered including usability and whether the tool used previous results to speed up its new analysis.

The accomplishments of this project could be classed "roughly" as Connectivity Level 2, Automation Level 2 and Synergy Level 2.

Key features of the overall automated formal methods relating to this and other similar examples: Connection to semi-formal models; color coding of analysis for reviews.

EXAMPLE 5 – Formal Test Generation

In the area of formal methods based automatic test generation, I was involved in an engineering production project [10], within the automotive industry, for assessing the coverage of test cases and improving them. Test cases were initially developed manually based on requirements and experience from on-field issues encountered in previous versions of the software. As requirements and models evolved in successive projects, there was no clear idea on structural coverage of models based on the existing test cases. Hence formal methods based automatic test case generation was utilized to automatically generate test cases for the gaps in structural coverage. Here, the formal methods technique of test case generation was well-connected to the models used and had immediate relevance in the systems engineering workflow. The overall level of automation was good although certain custom scripts had to be developed. In terms of synergy with the user, it was much higher than with "model-checking" as the coverage concepts were well-known from code-level coverage analysis and the technique was more of a push-button approach.

The accomplishments of this project could be classed "roughly" as Connectivity Level 3, Automation Level 2 and Synergy Level 3.

Key features of the overall automated formal methods relating to this example: Connection to semi-formal models; automatic generation of formal models; test case generation for specific coverage criteria; identification of gaps in coverage for existing test cases.

EXAMPLE 6 – Formal Specifications

During my research work for my doctorate [8], I was involved in developing a visual language and prototype tool to support Interval Temporal Logic (ITL), used for specification of (execution) behaviors, especially electronic system-level digital circuits. As my undergraduate (UG) and graduate level backgrounds were in materials science and engineering, I had a good foundation in mathematics and I quite liked the idea of the mathematics and logic behind ITL and Tempura, the ITL's executable subset. I also had a great liking for programming (in FORTRAN and Pascal) and used FORTRAN extensively for scientific computations, during my UG days, out of my own interest, hours on end! Time was spent on developing and running code, in many iterations, to get results that would have been unachievable through manual computations. However, during the doctoral studies, when I had my first awareness of formal methods, I also encountered the challenges first hand! I learnt that visualization and tool support enhance the uptake and usefulness of formal methods, whether it is for education or use within a research or commercial environment. It seems that, to bridge the gap, the gap must disappear, in some ways, through appropriate level of visual aids, automation and synergism to/with the diversity of users.

This example illustrates that there are many facets to any automation for formal methods including the need to support education and training needs.

Although this project was for a PhD dissertation and not for connecting to industrial state-of-the-art models, the accomplishments of this project could be classed "roughly" as Connectivity Level 1, Automation Level 1 and Synergy Level 1.

The team working in this research area have developed various other features such as an animation tool for Tempura and integration with other formal techniques such theorem proving [15].

Key features of the overall automated formal methods relating to this and other similar examples: Visual language; animation; executable specifications; connection to other formal methods including theorem proving.

A recently concluded project I was involved in, where automated formal methods were used, is the Proving Integrity of Complex Systems of Systems (PICASSOS) project which was part-funded by the UK Advanced Manufacturing Supply Chain Initiative (AMSCI) [21]. The project was a collaboration led by Ricardo, with partners Jaguar Land Rover, Johnson Matthey Battery Systems, YorkMetrics, D-RisQ and the universities of Oxford, Coventry and Warwick. End-of-project dissemination presentations and other information can be accessed from [21].

Current and future applications which will benefit from advances in automation with regard to MBSE and Formal Methods include Connected and Autonomous Vehicles. Techniques like machine learning will pose challenges not only to the various formal methods techniques, tools and the automation involved but also to the overall systems engineering process. Meeting such new challenges will help ensure that the relevant desired properties (wrt. safety, security etc.) are satisfied by the complex systems and systems of systems involved!

New and improved automated formal methods may benefit from the reuse potential in various core formal techniques and tools but caution must be exercised to leverage them appropriately, with a suitable systems engineering process, so that they have key properties, such as the ones discussed in this position paper, for success.

KEY CONSIDERATIONS RELATING TO AUTOMATION

In this section, the role of standards and risks with automation are discussed briefly.

Standards

Standards, such as DO-178C, provide a number of objectives that need to be satisfied based on the safety-criticality level of software. In case automation is involved, as in the case of automatic code generators for producing code from design models, the automatic code generation tool would need to be qualified to a particular tool qualification level [17]. After such a tool qualification, the automatically generated code may undergo less review than a manually coded one and hence brings efficiency and more confidence in the code and the overall process.

In the case of DO-178C, there are three different categories of tools identified and accordingly, the tool qualification process would be different. Some formal methods tools may help in reducing the activities with the development as well as verification part of the process and hence, after suitable qualification, such a tool may be very valuable in some safety-critical systems' projects.

Similar standards for other industry domains, for example ISO 26262 for automotive, have emerged more recently, and adopt similar principles.

Risks

Whilst automation brings many benefits in all industries such as semiconductor manufacturing, in terms of speed and accuracy, it is known to have some drawbacks, especially when it comes to automation of complex tasks where human interaction and/or awareness is also crucial. For example, inadequate training in autopilot technology and/or inadequate design of the user interface in the cockpit, sometimes combined with other system failures or weather scenarios, have resulted in air crashes. As a result, it must be understood that there are many risks to automation [20] and hence must be factored into the overall process planning.

CONCLUSIONS

To summarize, Figure 5 visually depicts that the eventual successful launch of a product depends on how the various models, whether IM, SM or FMs, are developed but also whether they are used both in a seamless and timely manner!

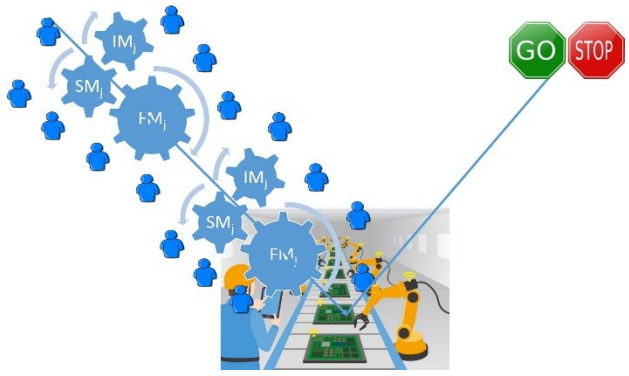

Figure 5: Success / failure in an eventual product launch!

Hence, I wish to state that, yes! indeed we require more automated formal methods but at the same time, they have to be well connected with the overall systems and software engineering process, the level of automation must be high enough to abstract the user from the intricacies of the formal notations and mathematics involved but at the same time be synergistic with a varied level of users, be it beginners or experts in background theory! For each different formal method consisting of a formal language and associated analysis techniques and tools, we need to have an idea of the current state in order to take it to the next level! As the overall process is a mix of models at varying degrees of formality, the role of guidelines becomes crucial at any level of abstraction.

ACKNOWLEDGMENTS

This position paper contribution for a proposed talk has been written based on my background and experience in the area of formal methods and systems engineering acquired at not only various global academic, industrial R&D organizations but also within the automotive and aerospace industries, including the European Commission funded Advanced Design Tools for Aircraft Systems And Airborne Software project at OFFIS R&D Institute, Germany, Premium Automotive R&D Programme, UK, General Motors Technical Centre, India and Rolls-Royce Sheffield UTC, UK. I wish to acknowledge all colleagues, with diverse backgrounds and experiences, with whom I had the chance to work with, including those who hired me for the specific job and role in the first place! Relevant experience considered includes the research and development as well as evaluation of formal methods techniques and tools, application in pilot projects within industry and technology transfer of industry-strength automated formal methods to support Model Based Systems Engineering across industry domains, primarily automotive and aerospace. Experiences from interaction with aerospace and automotive system, software and safety engineers as well as other researchers whilst in different roles, including a technology transfer role, have been influential in my continuing interest in a career in this area. The personal opinions expressed are based on an attempt to understand the progress of the state-of-the-art as I experienced it, in my own career, and, of course, debatable! Last but not the least, I wish to acknowledge the support of WMG's PICASSOS and UK Connected and Intelligent Transport Environment (UKCITE) projects for supporting my attendance at NFM 2017 and AFM 2017.

REFERENCES

[1] A Chakrapani Rao, M. Dixit and R. Sethu. 2015. Systems and methods for generating high-quality formal executable software feature requirements. (Oct. 2015). United States Patent No. 9,152,385, Filed Feb. 22nd 2012, Issued Oct. 6th 2015.

[2] A Chakrapani Rao, M. Dixit and R. Sethu. 2011. Formal Requirements Analysis Techniques for Software-Intensive Automotive Electronic Control Systems. *SAE Technical Paper* 2011-01-1002 (Apr. 2011). DOI: http://dx.doi.org/ 10.4271/2011-01-1002.

[3] A Chakrapani Rao, R. McMurran and P. Jones. 2008. A Critical Analysis of Model-Based Formal Verification Efforts within the Automotive Industry. *SAE Int. J. Passeng. Cars - Electron. Electr. Syst. 1(1)* (2009) 77-83. DOI: http://dx.doi.org/ 10.4271/2008-01-0220.

[4] A. Rao, R. McMurran, R. Peter Jones, M. A. Smith, N. Tudor and A. Burnard. 2007. Assessing the real worth of software tools to check the healthiness conditions of automotive software. In proceedings of the *3rd Institution of Engineering and Technology Conference on Automotive Electronics*. IET, UK, 1-8.

[5] A Chakrapani Rao. 2014. Model Based Systems Engineering for Complex Aerospace Systems. Invited Talk at Airbus (as part of UK-India Education and Research Initiative (UKIERI) visit from University of Sheffield, UK), Bengaluru, India, March 12th 2014.

[6] A Chakrapani Rao, A. C. Rajeev and A. Yeolekar. 2011. Applying Design Verification Tools in Automotive Software V&V. *SAE Technical Paper* 2011-01-0745 (Apr. 2011). DOI: http://dx.doi.org/ 10.4271/2011-01-0745.

[7] A. Chakrapani Rao and Jun Liu. 2014. Advances in Addressing Challenges in Complex Control Systems Design. *Proceedings of the Third International Conference on Advances in Control and Optimization of Dynamical Systems, Indian Institute of Technology Kanpur, India, March 13-15, 2014.*

[8] A. Chakrapani Rao. 2002. A Visual Framework for Formal Systems

Development Using Interval Temporal Logic. PhD Thesis, De Montfort University, UK.

[9] A. Chakrapani Rao. 2016. MBSE/SysML to NPSS Integration. Rolls-Royce Sheffield UTC report RRUTC/Shef/R/16105, July 2016, Rolls-Royce UTC Internal Technical Report, https://www.sheffield.ac.uk/systemsutc/index.

[10] A. Chakrapani Rao, Rupesh Kakade and Mohan Murugesan. 2013. Utilization of Simulink verification and validation (V&V) and simulink design verifier (SDV) for HVAC controls software. In: MATLAB Virtual Conference 2013 - European Track, Sheffield, United Kingdom, March 20th 2013.

[11] A. Joshi and M.P. Heimdahl, 2005. Model-based safety analysis of Simulink models using SCADE design verifier. In Computer Safety, Reliability, and Security. 2005, Springer. p. 122-135.

[12] Christian S. Calude, Declan Thompson 2016. Incompleteness, Undecidability and Automated Proofs. In Gerdt V., Koepf W., Seiler W., Vorozhtsov E. (eds) Computer Algebra in Scientific Computing. CASC 2016. Lecture Notes in Computer Science, vol 9890. Springer, Cham.

[13] Erika Ábrahám & Klaus Havelund. 2016. Some recent advances in automated analysis. Int J Softw Tools Technol Transfer (2016) 18: 121. doi:10.1007/s10009-015-0403-0.

[14] Hongman Kim, David Fried and Peter Menegay. 2012. Connecting SysML Models with Engineering Analyses to Support Multidisciplinary System Development. In 12th AIAA Aviation Technology, Integration, and Operations (ATIO) Conference and 14th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference, Aviation Technology, Integration, and Operations (ATIO) Conferences. http://dx.doi.org/10.2514/6.2012-5632.

[15] ITL Publications. 2017. ITL Related Publications. (April 2017). Retrieved April 2, 2017 from http://www.antonio - cau.co.uk/ITL/itlhomepagese8.html#x9-200008.

[16] Konrad Feyerabend and Bernhard Josko. 1997. A visual formalism for real time requirement specifications. In Miquel Bertran and Teodor Rus, editors, Transformation-Based Reactive Systems Development, Proceedings, 4th International AMAST Workshop on Real-Time Systems and Concurrent and Distributed Software, ARTS'97, Lecture Notes in Computer Science 1231, pages 156–168. Springer-Verlag, 1997.

[17] Leanna Rierson. 2013. DEVELOPING SAFETY-CRITICAL SOFTWARE A Practical Guide for Aviation Software and DO-178C Compliance. CRC Press, Taylor & Francis Group.

[18] OFFIS. Industrial Formal Verification Projects funded by automotive companies including DiamlerChrysler and BMW . OFFIS R&D Institute, Germany, 2001-04.

[19] P. Sampath, S. Arora and S. Ramesh. 2011. Evolving specifications formally. In Proceedings of IEEE 19th International Requirements Engineering Conference, Trento, 2011, pp. 5-14. doi: 10.1109/RE.2011.6051651.

[20] Peter G. Neumann. 1995. COMPUTER-RELATED RISKS. Addison-Wesley Publishing Company, 1995.

[21] PICASSOS. 2017. Project's end-of-project dissemination presentations and other information. In proceedings of PICASSOS Formal Methods Seminar, British Motor Museum, Gaydon, 28th Feb. 2017, Available from http://picassos.info/.

[22] R. Schlör, B. Josko and D. Werth. 1998. Using a visual formalism for design verification in industrial environments. In: Margaria T., Steffen B., Rückert R., Posegga J. (eds) Services and Visualization Towards User-Friendly Design. Lecture Notes in Computer Science, vol 1385. Springer, Berlin, Heidelberg.

[23] SAE ARP4761. Guidelines and Methods for Conducting the Safety Assessment Process on Civil Airborne Systems and Equipment. SAE International, December 1996.

[24] SAFEAIR II. Advanced Design Tools for Aircraft Systems and Airborne Software. Funded by the European Commission. OFFIS R&D Institute, Germany, 2002.