Advances in Addressing Challenges in Complex Control Systems Design

Arun Chakrapani Rao*, Jun Liu**

**Rolls-Royce University Technology Centre in Control and Monitoring Systems Engineering, Department of Automatic Control and Systems Engineering (ACSE), University of Sheffield, S1 3JD, UK (Tel: +44(0)114-222-5687; e-mail: A.Chakrapani-Rao@sheffield.ac.uk).*
***Department of ACSE, University of Sheffield, S1 3JD UK (e-mail: j.liu@sheffield.ac.uk).*

Abstract: This paper gives an account of various challenges that are faced in the design and development of complex control systems and software, within the automotive and aerospace domains, in particular, which are highly relevant to the incorporation of active mechanisms for dynamic systems. It also analyses what new recent advances are helping some of these being overcome in the research and engineering environments.

Keywords: Active Mechanisms for Dynamic Systems, Model-Based Design, Systems Engineering, V&V.

1. INTRODUCTION

Today's systems, irrespective of the domain, whether automotive or aerospace, for example, are getting ever more complex due to a number of factors. In the automotive domain, a proliferation of electronics and software-intensive features over the last decade is enabling the implementation of new requirements in automotive vehicles for achieving greater comfort, safety, reduced emissions and many others. In the aerospace domain, aircraft engines are becoming safer, quieter and more efficient, through the introduction of new systems involving control systems and software.

Incorporation of active mechanisms for dynamic systems is a challenging new area which requires the fusion of smart material based structures with computational designs that are not only efficient but also of very high quality. Complex problems can only be solved when such systems are designed to be adaptable without any decrease in reliability. Applications in automotive and aerospace include control systems incorporating smart sensors and actuators. Examples include the use of a shape memory alloy to activate a brake or the hood of a car and active louver mechanisms to control the airflow into the engine compartment without using motors. Such materials, for example, can change their shape, stiffness and/or other properties in response to changes in applied temperature, electric field or magnetic field. Potential future applications in aerospace include the use of self-healing composite materials for damage tolerance.

As the introduction of new control systems and software adds complexity to the overall system, for ensuring that all these new systems are of high quality, free from all sorts of potential errors, advances in techniques and tools are ever more crucial. This paper will touch upon various advances, including theory and practice, which will together help meet the challenges of the current state-of-the-art as well as lay the foundations for further new directions in research and development.

1.1 Control System Development Lifecycle

Typical control system and software development lifecycles are illustrated in the form of a V-model, examples of which are in (Fig. 1). The key stages are Requirements Engineering and Management, Control System Architecture Design, Subsystem Design and Development including Software and various levels of Testing.

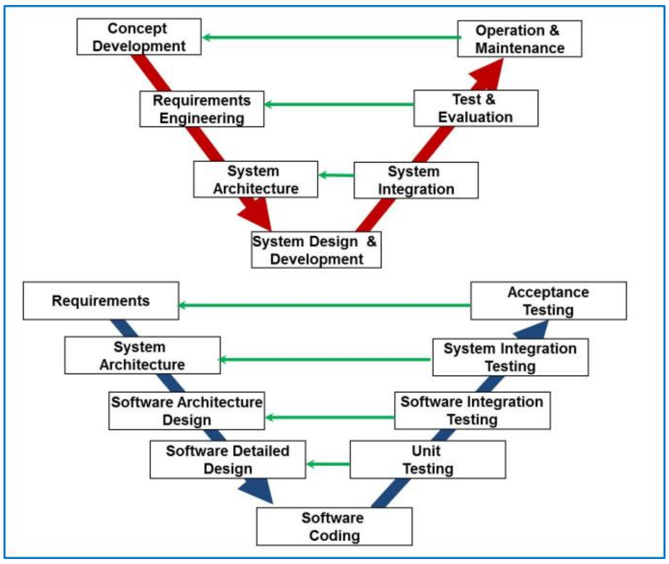

Fig. 1. V-model for Control System Development Lifecycle

In practice, engineers usually spend a lot of time going over the stages in the lifecycle many times. This can be due to reasons such as incremental development of the artefacts involved in that stage, defects found in testing or to respond to changes in requirements for that stage. Various statistics in literature including (Chakrapani Rao et al., 2011; Stecklein et al., 2004) indicate that a large number of defects occur early in the lifecycle as depicted abstractly in (Fig. 2). Such defects, if not found early, will be costly to fix later, if at all found later in testing. Hence it is extremely important to incorporate numerous types of analyses for verifying that the artefacts produced at each stage are of high quality.

1.2 Current Challenges in State-of-the-art

Current challenges in control system design involves increasing complexity of individual systems, integration aspects concerning incorporation of numerous individual systems designed separately, communication barriers between designers involved across disciplines and departments, gaps arising during successive stages of the development lifecycle depicted in Fig.1, to name a few.

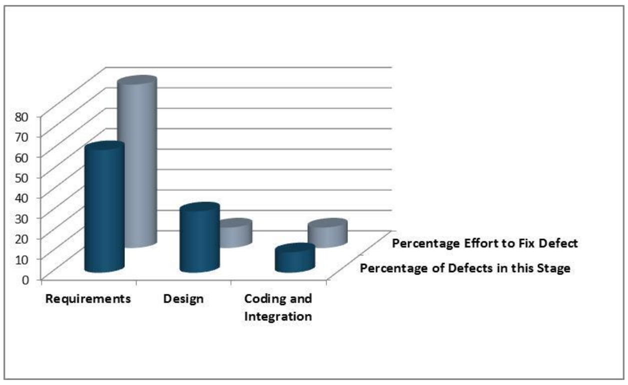

Fig. 2. Early stages critical in development lifecycle

Current state-of-the-art in control design testing in the industries is predominantly focussed on testing to detect defects in various stages of the development lifecycle, particularly late in the lifecycle when software code and hardware parts are available. Therefore, defects are usually discovered late with the risk of certain defects possibly missed or undetected. The consequence is that some defects can be discovered in-service by customers. In the area of safety-critical control systems, in particular, these challenges are met by incorporation of various advanced specification and verification techniques and tools in the development lifecycle.

2. MODEL BASED APPROACHES

Advances in the development of modelling languages and tools have now enabled control system algorithms to be developed and tested using a model-based approach. In this approach, the implementation comes much later while the model-based design focuses on a combination of virtual and physical models to trial out options and come out with a suitable design more easily and cheaply.

2.1 Model-Based Design (MBD) Paradigm

MBD is the preferred approach to dealing with the complexity of current control system development. Various modelling languages are available such as Simulink/Stateflow, SCADE, ASCET and Statemate providing a convenient level of abstraction and visualisation mechanism to develop control systems more conveniently.

2.2 Model-Based Systems Engineering (MBSE) Paradigm

According to INCOSE, “Systems Engineering is an engineering discipline whose responsibility is creating and executing an interdisciplinary process to ensure that the customer and stakeholder's needs are satisfied in a high quality, trustworthy, cost efficient and schedule compliant manner throughout a system's entire life cycle”. Model Based Systems Engineering (MBSE) has similarly been defined as “..fundamentally a thought process and provides a framework to allow the systems engineering team to be effective and consistent right from the start of any project” (Long and Scott, 2011).

MBSE approaches are now a focus for many companies, including those in the automotive and aerospace domains, in order to meet the challenges associated with the development of systems, including control systems.

3. SPECIFIC ADVANCES IN CONTROL V&V

In this section, we focus on advances that relate to V&V for control systems, particularly fitting with the MBD and MBSE approaches.

3.1 Traditional Simulation and Testing

Traditional approach to simulation and testing was ad-hoc with some simulations and testing carried out at various lifecycle stages without any relation between these efforts, for example different test cases being used at different stages. There is now a better co-ordination of all the testing efforts during the system development lifecycle. For example, in Model-In-the-Loop testing, incorporating models, test cases are written to check if the model implements the requirements correctly. Test cases test out various scenarios. These test cases are now usable in different testing environments, such as Hardware-In-the-Loop, Software-In-the-Loop and Processor-In-the-Loop environments which essentially incorporate physical components for the plant, actual embedded target hardware for the control algorithm and actual processor respectively. Newer environments incorporate driver models for Driver-In-the-Loop and pilot models for Pilot-In-the-Loop testing. All these new advances help verify the control algorithm and validate that the system being built is the right one for the application.

In addition, test case development methods have undergone improvements. For example, all execution paths within the algorithm can be covered by measuring the coverage achieved by existing test cases and developing new tests to cover the missing paths. COTS tools now assist in automatically generating test cases as well in addition to assessing the coverage of existing test cases.

3.2 Positive Influence from Standards

Previous and new standards, such as DO-178B and ISO 26262 (Conrad et al., 2011), and published industry best practice information, such as (Murphy et al., 2008) provide guidance and recommendations on developing systems with appropriate techniques and tools. In addition, they help ensure tool standards are high and fit-for-purpose with respect to the chosen applications.

3.3 Emergence of COTS Formal Methods Tools

This section briefly mentions commercial-of-the-shelf (COTS) modelling tools that are incorporating formal methods techniques. A detailed coverage of techniques would be out of scope for this paper but an outline would be provided.

Formal methods based languages and techniques involve the following, for example,

1. Advanced mathematical languages such as specification languages with precise semantics and associated techniques, for example Z specification language (Bowen, 1996).
2. Model-checking algorithms to check a formal description of a model, such as a control model, for certain properties (Huth and Ryan, 2004).
3. Static-checking techniques for code, such as a control algorithm in C code, for run-time errors without code execution (Chakrapani Rao et al., 2006).
4. Advanced automatic test case generation techniques for models, such as a control model in Simulink (Mohalik et al., 2013).

Prominent modelling tools incorporating formal techniques include Simulink Design Verifier, Embedded Validator, SCADE Design Verifier and Reactis Validator. On the controller code level, tools include PolySpace Verifier and LDRA. References to further information include ((Chakrapani Rao et al., 2006), (Chakrapani Rao et al., 2008), (Chakrapani Rao et al., 2011a), (Chakrapani Rao et al., 2011b), (McMurran et al., 2006)).

3.4 Other advances – Controller Synthesis

As an alternative to formal methods for verification, one can seek to formally design systems from specifications, known as correct-by-construction design. In fact, formal verification is often subject to the criticism that it is usually done after significant resources have already been put into the development of the system. As a result, if a problem is uncovered, it can be costly to fix. The alternative approach called system synthesis seeks to incorporate system specifications earlier in the development process, in order to design a provably correct system. However, this is challenging issue particularly for control system design. First of all, control systems, by definition, are systems designed to interact with other systems or processes. Hence, control systems are open systems, as opposed to closed systems, that are required to maintain ongoing interaction with their environments. The formal synthesis of open systems that are required to satisfy a given specification against all environments is, in general, a notoriously hard question and relies on progress from computer science in this area. Second, control systems are systems interacting with physical processes, which are inherently continuous in both time and space. Algorithmic formal methods are only effective for finite-state systems. Therefore, the gap between a continuous control system model and formal methods tools has to be resolved. This is usually done through a process called abstraction, which transform a concrete, possibly continuous model into a finite-state model, while preserving all the essential properties relevant to given specifications.

There have been recent breakthroughs in the computer science that, by restricting the class of specifications, the algorithmic difficulties can be relieved. This has sparked increased interest in the formal design of control systems over the last decade among the control community. The approach is often an abstraction-based, hierarchical design. The main workflow of these approaches has three steps: (i) construct finite abstractions of the dynamical control systems that preserve essential properties, (ii) solve a discrete synthesis problem based on the specification and abstraction and obtain a discrete control strategy, (iii) refine the discrete control strategy, often to a hybrid controller, that renders the closed-loop control system satisfy the specification. The approach is outlined in Fig. 3. This appears to a promising area of research in control systems.

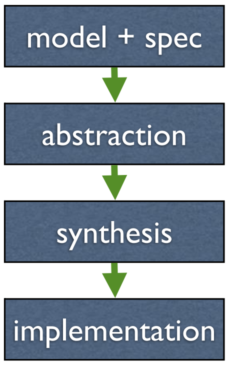

Fig. 3: An Abstraction-Based Hierarchical Approach to Correct-by-Construction Control System Design

However, despite the rich theory that has been developed for control system synthesis, little of this theory has been transformed into practice. A number of reasons may have contributed to this (Kupferman, 2012). We discuss a few here. First, despite the recent breakthroughs, the algorithm difficulties remain a hurdle to apply the methodology to industry-scale problems. Compositional or distributed techniques need to be developed to alleviate this issue by allowing the whole system to be constructed and certified systems incrementally. Second, current synthesis tools are mostly designed only for very limited specifications, such as linear temporal logic (LTL) or computation tree logic (CTL). For synthesis tools to have a comparable impact as industrial model checking tools (such as IMB's RuleBase), they need to embrace richer formalisms as specification, which may require further non-trivial research. Third, current synthesis tools only give a solution that satisfies a given specification, as opposed to an optimal solution, which is often sought after when designing a control system using classical tools such as optimal control theory. In fact, quantitative synthesis is still in its infancy and may again need further non-trivial research to make it suitable for practical implementations. Finally, the reason industry, especially that for safety-critical control systems where formal synthesis would be of value, is more willing to embrace formal verification than formal synthesis for the apparent reason that, formal synthesis requires a change of design paradigm, whilst formal verification does not. One approach to do so is of course ensure that ex-post verification is combined with the formal synthesis approach to deliver the level of confidence assurance needed, while allowing potentially significant reduction in costs for design and testing. Despite its promising aspects, this may take time to eventually happen for the above reasons.

4. FUTURE DIRECTIONS

There are substantial opportunities for further research and development and subsequent technology transfer to industry. Areas of research include the integration of multiple disciplines and multiple physics early in the design lifecycle. In the area of testing, some of the key areas include scalability of formal techniques to meet more complex designs, automatic test case generation from various model artefacts, including requirements, utilisation of HPC and making formal techniques transparent to the wider user community including especially users in the industry.

5. CONCLUSIONS

A short survey of state-of-the-art in research, development and engineering relating to complex control design and testing has been attempted, applicable to the design of active mechanisms for dynamic systems, based on authors' previous experiences. A brief indication of research that needs to be done is also indicated.

ACKNOWLEDGEMENTS

The authors wish to thank Prof. V. Kadirkamanathan, Head of Department of ACSE at The University of Sheffield, for encouraging us to write and submit a paper to this conference. Additionally, we wish to acknowledge the support of a UKIERI (UK-India Education and Research Initiative) grant: Integrated Sensing, Monitoring and Healing for Autonomous Systems.

REFERENCES

Bowen, Jonathan. (1996). *Formal specification and documentation using Z: a case-study approach*. ISBN 1-85032-230-9, International Thomson Computer Press, International Thomson Publishing, USA.

Chakrapani Rao, Arun, McMurran, Ross, Peter Jones, R, Anthony Smith, Michael, Tudor, Nick and Burnard, Andrew. (2006). Assessing the real worth of software tools to check the healthiness conditions of automotive software. *In proceedings of The IET Automotive Electronics Conference*, IEEE, Coventry, UK.

Chakrapani Rao, Arun, McMurran, Ross and Peter Jones, R. (2008). A critical analysis of model-based formal verification efforts within the automotive industry. *SAE Technical Paper 2008-01-0220*, SAE International, Detroit, USA.

Chakrapani Rao, Arun, Rajeev, A.C. and Yeolekar, Anand (2011a). Applying design verification tools in automotive software v&v. *SAE Technical Paper 2011-01-0745*, *SAE International, Detroit, USA*.

Chakrapani Rao, Arun, Dixit, Manoj and Sethu, Ramesh (2011b). Formal requirements analysis techniques for software-intensive automotive electronic control systems. *SAE Technical Paper 2011-01-1002*, *SAE International, Detroit, USA*.

Conrad, Mirko, Sandmann, Guido, Munier, Patrick. (2011). Software tool qualification according to ISO 26262. *SAE Technical Paper 2011-01-1005*, SAE International, Detroit, USA.

Huth, M. and Ryan, M. (2004). *Logic in computer science: reasoning about systems.* ISBN-13: 978-0521543101, Cambridge University Press, UK.

Kupferman, Orna. (2012). Recent challenges and ideas in temporal synthesis. *SOFSEM 2012: Theory and Practice of Computer Science*. Springer Berlin Heidelberg, 88-98.

Long, David, Scott, Zane. (2011). *A primer for model-based systems engineering*, ISBN 978-1-105-58810-5, 2nd edition, Vitech Corporation, USA.

McMurran, Ross, Chakrapani Rao, Arun and Peter Jones, R. (2006). Model based validation techniques for complex control systems. *In Proceedings of the Hybrid Vehicle Conference, Coventry, UK*.

Mohalik, Swarup, Gadkari, Ambar, Yeolekar, Anand, K. C., Shashidhar, S., Ramesh. (2013). Automatic test case generation from Simulink/Stateflow models using model checking. *Software Testing, Verification and Reliability*, DOI: 10.1002/stvr.1489, Wiley Online Library.

Murphy, Brett, Wakefield, Amory and Friedman, Jon. (2008). Best practices for verification, validation, and test in model-based design. *SAE Technical Paper 2008-01-1469*, SAE International, Detroit, USA.

Stecklein, Jonette M., Dabney, Jim, Dick, Brandon, Haskins, Bill, Lovell, Randy, Moroney, Gregory (2004). Error cost escalation through the project life cycle. *In proceedings of the 14th Annual International Symposium of International Council on Systems Engineering*, USA.