

Simulation to Support Reasoning about Aerospace Complexity

Colin Brain & Richard Maguire, SE Validation Limited
Dominic Manchee & Alastair Pidgeon, Scisys Limited

Introduction

Engineers have always used models to help them reason about solutions to problems. Often these are individual mental models, built up through education, experience and dominant logic. A characteristic of complex systems is that their successful realisation is beyond the compass of a single mind and the competence of any single discipline. Thus engineers' individual implicit models alone are inadequate to support the safe integration of complex aerospace systems. They need external support to help them reason about the correctness, consistency and completeness of their individual contributions. This paper addresses how models and simulations can be used to improve shared awareness of issues and goals and support for design decisions, particularly in the context of developing components that efficiently integrate into complex systems and systems of systems. However, the more reliance that is placed on these models and simulations, the greater the impact if they are wrong. Thus the paper stresses the role that verification and validation play in reducing the risks inherent in this increasing reliance. These notes are written to support the slides used in the presentation and each section refers to specific slides.

Slide 4 gives a brief overview of some basic indicators of complexity. The fact that the scale of a problem is too large for an individual mind immediately implies that social interactions are necessary in order to solve the problem – with all the implications that this has on communication barriers and lack of clarity. This means that problems can no longer be solved by 'common sense' alone, since the latter is certainly not commonly appreciated and its origin and biases will not be apparent to anyone. Thus increased formality is required to ensure that clear unambiguous communication is achieved. This implies mathematical and logical notations and graphical or pictorial representations, preferably animated.

The problems involved due to the inability to functionally partition either the problem, or the solution, will be well known to most system engineers. Similarly, since most approaches to system engineering are founded on the assumption of firm requirements, requirement and boundary uncertainty, for example in network-centric or service-based architectures, creates considerable complexity in the problem to be solved.

There are often trade-offs between what is done by the machine and what needs to be done external to the machine in its use domain. Often reducing the complexity of the solution increases the complexity for the users. Take for example 'standard' multi-lingual user handbooks – simpler for the supplier because 'one-size-fits-all' but not simpler for the user.

Slide 5 addresses some of the issues implicit with building 'just-in-time' complex aerospace systems. Because of the need to get aircraft into Service quickly, to gain revenue, or to close a military capability gap, there can be a high commercial penalty for not getting it 'right first time'. Thus the development must be assurance-based as far as possible. Physical testing should only be to confirm that it is right, not to find if it is wrong. Also pragmatism and cost constrain the number and scope of tests that are possible. Each physical ground or flight test should thus be backed by an assurance that success in the simulated operation, which is what a test is, infers that the related range of operations will also be successful. This usually demands extensive sensitivity studies, particularly in novel or complex situations. Passing a test is irrelevant if it is the wrong tests. Simulations can also play a very important role in facilitating inter-team communication – if of course they are right.

All this means that it is very desirable to 'test' in simulation before a complex aerospace system design is frozen. However, very few Aerospace enterprises have any comprehensive processes to ensure that such simulations are sufficiently valid for their results to be used for making critical design decisions.

Slide 6 continues this theme by giving high-level definitions of verification and validation. The important distinction is that verification is internal – does the item meet its specification. Thus this can usually be done through rig-testing an item alone, stimulated by specified interactions. Validation by contrast is external and is concerned with the item integrating into its context 'system' to satisfy higher-level requirements and constraints. Validation may be usefully separated into two parts, structural and behavioural. Structural validation is concerned with mapping all the right interactions from the real-world, whereas behavioural validation is concerned with comparison between simulated and real outputs given the same inputs. As discussed above, the most valuable contribution that simulation can make to reasoning about aerospace complexity is before a 'real' system is built – when there is clearly no 'real' with which to compare its behaviour. It also follows that behavioural validation without structural validation is illusory, since there can be no certainty that the inputs used to generate output comparisons are correct or complete. Thus the best that can be achieved for validating predictive simulations is to minimise the risk of erroneous assumptions about the interactions that an item has with its context / environment. Often these simulations will involve hierarchies (or networks) of interactions, because the systems have hierarchies (or networks) of components. Thus validating at each level or node is dependant on having consistently disaggregated contexts or environments.

Slide 7 outlines one technique that is particularly well suited to meeting this need. This is Problem Frames, developed by Michael Jackson (for a more complete overview of the technique, see references: Jackson, M., *Software Requirements & Specifications*, Addison-Wesley, 1995, ISBN 0-201-87712-0 and Jackson, M., *Problem Frames*, Addison-Wesley, 2001, ISBN 0-201-5967-X). There is currently a lot of academic research in this area and the British Computer Society Requirements Engineering Group recently ran a very successful workshop on the topic at the Open University. SE Validation Limited in conjunction with Adelard LLP has developed a Problem Frame application for Adelard's Assurance and Safety Case Environment (ASCE) and the example diagrams in the following slides are exported from this tool.

Slides 8, 9 and 10 cover an hypothetical example problem frame for an aircraft fuel system. The first of these slides shows a complex, yet still idealised, view of the interactions between the fuel system 'domain of interest' and other domains. The 'share' domains mark interfaces, since an interface exists in at least two domains. For example the garden wall between my and my neighbour's garden is 'part of' both gardens. However the fact that it exists in both domains does not imply that there is joint ownership of it or that the perception of the wall when viewed from my garden is the same as when viewed from his. Thus in Problem Frames there are equivalent concepts of one domain needing to make the specified interface properties 'true' and also of different views of the shared phenomena from different domains. The 'part-of' relationship can also be used to denote decomposition. For use in V&V of models and simulations a 'whole-of' relationship has been introduced to Problem Frames in order to facilitate different domain viewpoints – for example as in Professor Mike Moulding's 'Y' model for simulation validation. The dotted ellipses in the network represent behavioural requirements. One domain has a behavioural requirement on another if it relies on services from that domain in order to deliver its specified interface states. It should be noted that Problem Frames are not an extension of UML or SysML. Although these notations are being used together there are some very important philosophical differences. For example there is a lot of difference in viewing Users and the operating environment as extensions to the system being designed and viewing the system from the perspective of the User and his or her needs.

Although the network shown in slide 8 is complex, it is a starting point and there is no reason why it should not be developed and reused over time. For specific design issues a reduced problem frame can be produced that abstracts from the first by ignoring shared phenomena or requirements that are not significant to the analysis in question. Slide 9 is an example of this, where the domain of interest

has been reduced to the control of the fuel system in flight and many of the shared phenomena have been idealised. For example, here it has been assumed that the interaction between the fuel system and the external atmosphere can be represented by assuming the international standard temperate atmosphere and Mach-number related kinetic heating. This reduction and simplification represents a 'validation' risk, but if formal tracking of changes is used, this change is open to review and challenge and apparent to any engineer viewing the problem frame. Decisions to reduce, abstract, simplify or segment make more detailed verification analyses feasible, but all introduce validation risks. However once these decisions become explicit the risks involved can be managed. It also becomes much easier to introduce 'problem' patterns that are reusable, but for critical systems this must be done in the full knowledge of the risks involved.

Slide 11 shows a suggested high-level approach incorporating many of these factors. The overall methodology must support a number of different perspectives or representation spaces: requirements, artefacts, processes and decisions. It must link knowledge management techniques with tools to abstract, simulate and reason about engineering issues. The process may be initiated in two ways, either from empirical evidence of performance of an existing system or changes in needs identified through stakeholder analysis and engagement. Such techniques may identify needs, such as to increase range from 2000 Km to 3000 Km. Whilst there are many options to meet this need, it is essential that the overall approach permits traceability to meet concerns such as safety and design integrity.

Specific notes:

- simulation models are in fact abstractions of engineering data, their fidelity is contingent on the context and the problem to be overcome;
- requirements representation space may include statements of need and engineering requirements and specifications captured in advanced tools;
- artefacts are outputs of the engineering processes;
- to support exploration of the problem space and reuse, abstracts of engineering concepts may be captured in repositories.

Slide 12 The engineering designer may be faced with a number of contradictions posed by the problem context. Often these contradictions lead to complexity in the solution. The designer may need to adopt several different strategies in order to explore different approaches to overcoming these contradictions. These strategies may need to include :

1. Segmentation and division
2. Merging, combining and integration.

The principle of segmentation means that once engineering data has been abstracted, it is divided into component parts. In the example Fuel System Problem Frame shown in slide 8 the structure of the assumed segmentation is represented by the 'part-of linkages' shown and the component interfaces by the 'share' nodes. Simplifying the Problem Frame then represents structural abstraction by ignoring interactions that are judged unimportant to the specific context under consideration. However, the risks involved in both the process of segmentation and abstraction must be managed if the integration of the designed components is to meet the behavioural requirements and constraints. Often this means that the process of re-combining the segmented domains must be explicitly considered when the segmentation is done.

Merging, combining and composition may be used in the engineering domain to enable similar parts to perform parallel operations. In this case the problem frame dependencies may change.

Once the simulation artefact has been developed, it is essential that the position, or state of, the artefact in meeting the problem and sub-problem requirements are captured; this includes information on assumptions and design decisions. This will enable retrieval of important knowledge from the artefact and decision space for subsequent system engineering processes and for .

Here the choice of simulation environment is not important. Indeed it may be important to use simulation platform-independent models to ensure portability. The current focus with these has been on how you develop and execute the simulation models. Future challenges include the provision of tool support for the abstraction, merging and recombination of engineering data to permit radical improvements in the system engineering processes.

In order to support validation of the simulation and synthetic environment components it is important to capture knowledge in the problem, requirement, artefact and decision spaces, since it must reflect the trade-off decisions made. Lack of shared understanding of the engineering design space or misinterpretation of analysis results, though in the context of concurrent engineering across the supply chain, may itself lead to conflicts and increased complexity in the solution space.

Slides 13 and 14 conclude that:

1. This very brief overview has argued that simulation is key to reasoning about aerospace complexity.
2. Validation of simulations is vitally important, if the risks of using simulations as a key means of communication and of predicting the validity of system design solutions are to be managed.
3. In just the same way that system validation needs to be predictive, so does the simulation validation that supports it. Clearly assurance gained through simulation fails if you find out (later) that the simulation was fatally flawed. Since behavioural validation requires comparison with reference information about the behaviour of the real system, which is not available during predictive use, the emphasis must be on structural validation.
4. Structural validation needs context disaggregation to support reasoning about the adequacy of environment representation for component simulation.
5. Problem Frames appear an effective way forward and can be used with other methods such as decision networks or Goal Structuring Notation to provide means of linking the different information spaces.
6. Validation then needs to focus on justifying the assumed shared interface phenomena.
7. The suggested high-level process supports tracking both of design rationale and how solution artefacts solve the problem.
8. There is a need for these relationships to be codified to support downstream activities and problem-, decision- and solution-pattern reuse.
9. Further work is needed in all of these areas to allow the widespread, confident, use of simulation to support reasoning about aerospace complexity.