# Practical Application of Formal Methods in Modeling and Simulation

**D. Richard Kuhn**
**National Institute of Standards and Technology**
**kuhn@nist.gov**

**Dan Craigen, Mark Saaltink**
**ORA Canada**
**dan@ora.on.ca**
**mark@ora.on.ca**

**Keywords:** Formal specifications, formal verification, lightweight formal methods, network analysis

## Abstract

This paper provides an introduction to applying formal methods to modeling and simulation problems at reasonable cost. Two approaches are discussed. First, *lightweight formal methods* combine simplified specification approaches with automated analysis, making it possible to analyze requirements and designs early in the development cycle. Second, by *hiding the complexity of the formal models* and providing the analytical results solely in terms understood by domain experts, new formal tools provide rapid feedback to requirements teams and developers.

## LIGHTWEIGHT FORMAL METHODS

What are "lightweight formal methods" and how can they be useful for the practicing engineer? Traditionally, formal (i.e., mathematical) software development methods have been promoted as the only approach that can guarantee the absence of certain classes of errors. By specifying software behavior formally, rigorous proofs of its properties can be developed. But the great expense of these methods has confined their use to a handful of high-security or safety critical systems. "Lightweight" methods may involve both greater automation of formal analysis, and more focused application of formal techniques. As a result they can be cost effective for a broad range of problems.

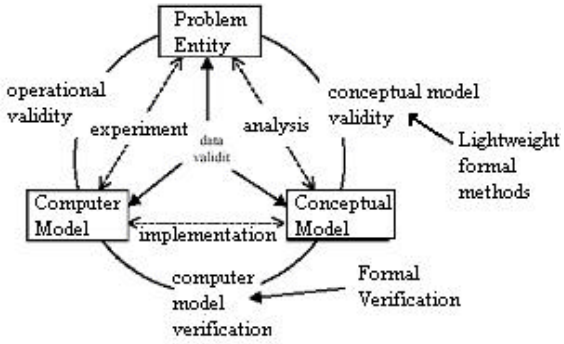


**Figure 1.** Formal techniques in M&S

To frame the discussion, consider the view of the modeling process shown in Figure 1 [1; 2] annotated to illustrate the applicability of formal techniques. Traditionally, formal techniques have been applied to the software *verification* problem. Given a formal definition of requirements, *R*, and a formal specification of the software (although typically only a top level specification), *S*, theorem proving tools can be used to assist in proving that the specification meets the requirements, i.e., that $S \Rightarrow R$.

While the verifications shown in Figure 1 can be conducted semi-automatically, and proofs checked mechanically, *validation* is a different problem. A succinct distinction between verification and validation is that verification is "building the system right," while validation is "building the right system." If we have a set of requirements, we can verify, formally or informally, that the system implements the requirements. But validation is necessarily an informal process. Only human judgment can determine if the system that was specified and built is the right one for the job.

Despite the necessity of using human judgment in the validation process, formal methods do have a place in validation, particularly in large complex applications such as M&S. One of the most promising applications of formal techniques is the "lightweight" application of formal methods for requirements modeling [3]. By stating requirements formally, theorem-proving tools can be used to explore properties, often detecting conflicts between different requirements or missing assumptions. This approach does not replace human judgment, but can aid in determining if the "right system" has been specified by making it easier to determine if desired properties hold. A particular advantage to using lightweight formal methods in requirements engineering is that ambiguities and inconsistencies in requirements are discovered early, when they can be corrected with much less expense than after code has been developed.

A significant difference between the validation problem for M&S systems and for software designed for control or calculation is that M&S systems have two types of validation requirements. By definition, the M&S system must model and predict behavior of some real world entity. This problem has been called "operational validation." A second aspect to the validation problem for M&S systems is "conceptual model validation," which is concerned with ensuring that the assumptions underlying

the conceptual model are correct and that the logic and structure of the model are suitable for the model's intended purpose [1]. Where formal methods have been applied to validation, their use corresponds most closely to the problem of conceptual model validation. Figure 1 shows a view of the modeling process [1;2], annotated to illustrate the applicability of formal techniques.

Because the conceptual model describes what is to be represented by the simulation, it must include assumptions about the system and its environment, equations and algorithms, data, and relationships between model entities. Although algorithms and equations are necessarily formal statements, the assumptions and relationships are most often described using natural language, which introduces the potential for ambiguities and misunderstandings between developers, users, and subject matter experts. An increasingly popular trend in formal techniques, lightweight formal methods have shown a potential for detecting major errors in requirements statements, without the expense of a formal design verification, by applying formal analysis to earlier products of the system design process.

The basic premise of this approach is to use formal techniques in analyzing the assumptions, relationships, and properties of requirements stated in a requirements statement or conceptual model. An advantage of the approach is that it can be applied to partial specifications, or to a limited segment of a complete specification. The analysis is done in three phases [4]:

1. Restate the requirements and conceptual model in a formal (or semi-formal) notation, typically a state table description.
2. Identify and correct ambiguities, conflicts, and inconsistencies.
3. Use a model checker or theorem prover to study system behavior, demonstrate properties, and produce traces of system behavior.

Developers, users, and subject matter experts can then use these results to improve the conceptual model.

The representational abstraction phase of conceptual model design uses a variety of modeling methods to represent simulation elements and their relationships [5;6]. Notations such as those provided in the Unified Modeling Language are often effective. While standard UML does not contain sufficient formality to map directly to formal representations used by model checkers, some more recent additions to UML may make this possible. The Object Constraint Language [7] includes elements of first order logic that, when combined with some of the state machine representations of UML, can provide a rigorous system specification. Because popular model checkers use state machine representations as input, a conceptual model defined with OCL would appear to have the potential for efficient translation into the input notations of either theorem proving tools or model checkers such as the popular SMV and SPIN tools. However, we must admit to some uncertainty that a rigorous foundation for UML/OCL can be developed; reverse engineering a rigorous foundation into an existing language is generally exceedingly difficult. Jackson's Alloy [8] uses a well-founded UML/OCL-like notation to perform formal modeling and analysis.

A more rigorous notation, the Z formal specification language [9] has primarily been used to model system requirements. ORA Canada's Z/EVES system [10], which has been used in 62 countries, uses state-of-the-art formal methods techniques,, integrating the Z specification notation with a leading automated deduction capability. The resulting system supports the analysis of Z specifications in several ways:

- Syntax and type checking.
- Schema expansion.
- Precondition calculation.
- Domain checking.
- General theorem proving.

From a lightweight formal methods perspective and technology transfer perspective, it is important to note that users can be introduced to Z/EVES capabilities in steps. For example, little knowledge of the theorem prover is required for syntax and type checking, schema expansion, and precondition calculation. Even with domain checking, many of the proof obligations are easily proven. In more difficult cases, generating the proof obligation is often a substantial aid in determining whether a specification is meaningful. The use of engineering judgement in what analyses to perform and to what detail is crucial to the beneficial use of Z/EVES, in particular, and formal methods, in general. Example applications of Z/EVES include security and safety systems [11, 12,13].

## Example Applications

A particularly interesting aspect of the lightweight approach to formal methods is that it has been used to model and analyze the behavior of software, hardware, and humans acting together in systems. Agerholm and Larsen [14] describe the application of formal modeling to a NASA extravehicular activity system. Using the PVS theorem-proving tool, the authors were able to model the EVA system and study its properties using a model that is essentially executable. Lutz [15] describes requirements validation of onboard fault monitors for a spacecraft. An

interesting aspect of this project is that developers were able to reuse the requirements model for a second project that evolved from the first in a series of builds.

Janssen et al. [16] describe the application of model checking to the analysis of automated business processes, such as insurance claim processing. The formal model is used to ensure that processes maintain desired properties, such as ensuring that the proper sequence of processing is maintained, that two mutually exclusive outcomes are prevented by the system, or that particular events always lead to the correct outcome. The formal analysis helps to prevent unexpected failures that can occur in large distributed systems where processes occur with partial human intervention. By modeling processes at the requirements stage, developers can identify problems that might require major rework if not detected until the system is built and tested.

## Benefits, Costs, and Success Characteristics

The true test of any method, of course, is whether its benefits outweigh its expense in time and materials. Three case studies reviewed by Easterbrook et al. [4] describe the use of lightweight formal methods in modeling systems for the International Space Station, detailing the effort involved:

- High level Fault Detection, Isolation, and Recovery requirements – This effort formalized 18 pages of text requirements, then used the PVS theorem proving tool to analyze FDIR properties. A total of 15 ambiguities and inconsistencies were discovered, at a cost of two staff-months.
- Bus controller FDIR requirements – This study analyzed requirements for the controller for the main communications bus on the space station. A 15-page detailed requirements document was formalized using the SCR methodology. A large number of ambiguities in the original English language requirements document were discovered, using approximately 1.5 staff-months effort.
- Cassini deep space probe fault protection – A total of 85 pages of English language requirements were formalized using state tables followed by PVS specifications. The analysis detected 37 problems with the requirements, including 10 cases of inadequate handling of off-nominal/failure conditions. Approximately 12 staff-months were required for this effort.

Like other successful applications of lightweight formal methods, these projects shared characteristics that appear to be important for success from a cost/benefit standpoint:

- Formal methods were only used where existing, informal requirements review techniques had been inadequate. Experience had shown that extensive review of requirements by experts still resulted in flaws carried forward to implementations.
- Selective use of formal methods: only the most critical or complex aspects of requirements were formally analyzed.

## HIDING COMPLEXITY – CASE STUDY

Hiding complex technologies is one well-recognized means of successful technology transfer. ([17] and [18] discuss in-depth formal methods technology transfer issues and approaches.) For example, we do not need to fully understand the technologies underlying our DVD players and Digital Cameras to enjoy the benefits of such products. In a similar manner, the benefits of formal methods modeling and analysis can be brought to various domains where clear value is added, yet the complexity of the technology is hidden from the domain experts. This approach provides one means of introducing logic-based models into the M&S world and, while providing M&S capabilities, supports the formal analysis of such models to determine properties of interest. We concretize the discussion by using a formal logic-based model to capture various aspects of IP networks.

The Cayenne Network Analyzer (CNA) works on models of networks; this data can correspond to a deployed network, a planned network, or a prior version of an existing network. The analyses provided, particularly change impact analysis, can help the network manager assess the *functionality* or *security* of a network and the impact of changes.

The CNA can model network devices such as hosts, gateways, and routers; physical networks such as LANs or dialup lines; routing tables; access control and filtering rules; and services offered by hosts. The CNA offers several analyses:

- Reachability of one host from another.
- Differences in reachability between two configurations of a network.
- Accessibility of services; and changes to accessibility of services in two networks.

The CNA analysis is *exhaustive*. Every possible packet is accounted for in the results, so there is no need to make guesses about suitable test cases or to measure the coverage of tests. The exhaustive analysis can be constrained by, for example, specifying a source or destination host or by specifying some other aspect of the packet.

The following example shows how a network manager might use the CNA to assess the impact of a change to access rules. Figure 2 shows the network involved. A

backbone connects smaller LANs for three divisions of a business, each protected by a gateway/firewall. Hosts Sales WS and Admin WS have symbolic addresses, and are used to represent *any* of the workstations on their LANs. The only constraint is that their addresses lie within the block of network addresses assigned to the LAN, and are not the same as the gateway or server address.

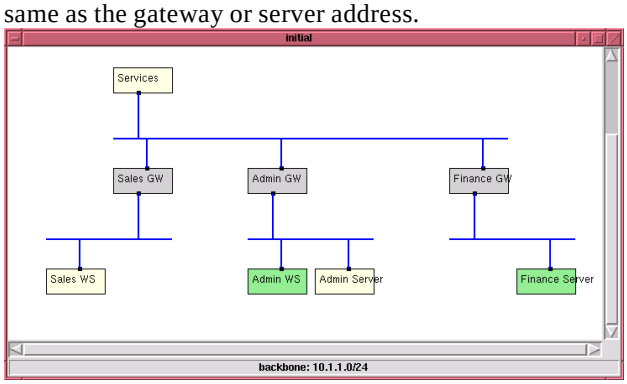


**Figure 2:** The Network Topology

One simple analysis is to show the services available to a host. Figure 3 shows the resulting table of available services.

| Source Host | Prot. | Source IP | Dest. IP | Dest. Port | Dest Host | Service |
|---|---|---|---|---|---|---|
| Sales WS | tcp | sales-ws-address | 10.1.1.2 | smtp | Services | mail |
| Sales WS | tcp | sales-ws-address | 10.1.1.2 | www | Services | web |
| Sales WS | udp, tcp | sales-ws-address | 10.1.1.2 | domain | Services | DNS |

**Figure 3:** Query Result: Services Available to Sales WS

In our example, we suppose that a user in the sales department wants to be able to log in to the administration server using telnet, but is unable to do so in the existing configuration as all telnet traffic is blocked leaving the Sales gateway (as shown in Figure 4). In order to allow the requested traffic, a new entry is made, allowing outgoing telnet connections. Figure 5 shows the revised rules.

| Interface | Protocol | Dst port | Action |
|---|---|---|---|
| eth0 | tcp | smtp | ACCEPT |
| eth0 | tcp | www | ACCEPT |
| eth0 | | domain | ACCEPT |
| eth0 | | | DENY |

**Figure 4:** Initial Output Filter of Sales GW

| Interface | Protocol | Dst port | Action |
|---|---|---|---|
| eth0 | tcp | telnet | ACCEPT |
| eth0 | tcp | smtp | ACCEPT |
| eth0 | tcp | www | ACCEPT |
| eth0 | | domain | ACCEPT |
| eth0 | | | DENY |

**Figure 5:** Revised Output Filter/Sales GW

We can use the CNA to discover what services are now accessible that were not previously. Figure 6 shows the results of this query.

| Source Host | Prot. | Source IP | Dest. IP | Dest. Port | Dest Host | Service |
|---|---|---|---|---|---|---|
| Sales GW | tcp | 10.1.1.10 | 10.1.3.2 | telnet | Admin Server | Remote login |
| Sales GW | tcp | 10.1.1.10 | 10.1.4.2 | telnet | Finance Server | Remote login |
| Sales WS | tcp | sales-ws-address | 10.1.3.2 | telnet | Admin Server | Remote login |
| Sales WS | tcp | sales-ws-address | 10.1.4.2 | telnet | Finance Server | Remote login |

**Figure 6:** Query Result: Services Newly Available After Final Filter Revisions

As can be seen, the change has had some unintended effects: now any workstation in the sales network can not only login to the administration server, but also the finance server.

This analysis might prompt a more elaborate revision, where the input filters for the administration and finance networks are strengthened. When these changes are made, and we again query what services are newly available, the result, shown in Figure 7, is exactly what we want: one user (at address 10.1.2.5 in the Sales network) has access to the administration server.

| Source Host | Prot. | Source IP | Dest. IP | Dest. Port | Dest Host | Service |
|---|---|---|---|---|---|---|
| Sales WS | tcp | 10.1.2.5 | 10.1.3.2 | telnet | Admin Server | Remote login |

**Figure 7:** Query Result: Services No Longer Available After Final Filter Revisions

We can also see which services are no longer available (by querying what is newly available when changing from the new to the old configuration). Figure 8 shows that we have removed access to remote login from a few servers and a gateway.

| Source Host | Prot. | Source IP | Dest. IP | Dest. Port | Dest Host | Service |
|---|---|---|---|---|---|---|
| Services | tcp | 10.1.1.2 | 10.1.4.2 | telnet | Finance Server | Remote login |
| Admin GW | tcp | 10.1.1.11 | 10.1.4.2 | telnet | Finance Server | Remote login |
| Admin Server | tcp | 10.1.3.2 | 10.1.4.2 | telnet | Finance Server | Remote login |
| Services | tcp | 10.1.1.2 | 10.1.3.2 | telnet | Admin Server | Remote login |

**Figure 8:** Query Result: Services Newly Available After First Filter Revision

## Change of Topology

The next example is derived from an actual situation, where we restructured our network to include a DMZ and restricted access to our LAN. In the initial configuration a router with some simple firewall rules connected the LAN directly to the Internet. The LAN had several workstations, a printer, and a main server with dial-up PPP access. We have a class C address block 206.191.58.0/24 and do not use any address translations.

In this model, there are three symbolic addresses used: the external host has any address outside our address block; the workstation has any address in the range 206.191.58.5 to 206.191.58.39, and the dialup has any address in the range 206.191.58.40 to 206.191.58.49. The actual network has a number of workstations and dialup hosts.

The new configuration inserted a new bastion host between the router and LAN. Web pages and an ftp server were moved from the internal server to the bastion, and the bastion also functioned as a firewall for access to and from the LAN. In order to minimize disruption to external users of the services, the new bastion host used the IP number (206.191.58.2) that had originally been assigned to the server. The internal server received a new, previously unused, address.

| Source Host | Source IP | Dest. IP | Fate 1 | Fate 2 |
|---|---|---|---|---|
| Workstation, gw, dailup Outside | | 206.191.58.4 | LOST | DELVD |
| Workstation, gw, dialup | ~206.191.58.* | 206.191.58.4 | LOST | DELVD |
| Outside | ~206.191.58.* | 206.191.58.19 | LOST | DELVD |
| Gw | | 206.191.58.19 | LOST | DELVD |
| Outside | ~206.191.58.* | 206.191.58.2 | DELVD | DELVD |
| Gw, dialup | | 206.191.58.2 | DELVD | DELVD |
| Outside | ~206.191.58.* | 206.191.58.19 | DELVD | DELVD |
| Workstation, dialup | | 206.191.58.19 | DELVD | DELVD |
| Workstation, dialup | | 206.191.58.2 | DELVD | LOST |
| Workstation, dialup | | 206.191.58.1 | DELVD | LOST |

**Figure 9:** Different Fates in the Old and New Configurations

Rather than renumber all our internal hosts, we decided to divide our class C address space into two blocks, the first, 206.191.58.0-206.191.58.3 for the DMZ (giving us exactly two usable host addresses), and the remainder, 206.191.58.4-206.191.58.255, for the internal LAN. We used the CNA to determine whether we had correctly revised our routing rules and the results appear in Figure 9. This shows all packets, including possibly spoofed ones, sent by any host in the model, having a different fate in the two configurations. There are four kinds of difference:

- Packets with a destination of 206.191.58.4 used to be lost; they are now delivered to the bastion host. This is expected, as that address was unused in the original configuration.
- Packets with a destination of 206.191.58.19 used to be lost; they are now delivered to the internal server host. This is also expected, as that address was unused in the original configuration. (In some rows, these packets are shown as being delivered to the workstation. A side condition, not visible in the figure, shows that this covers the case where the workstation address variable has this .19 address as its value.)
- Packets with a destination of 206.191.58.2 are delivered to the bastion rather than the original server.
- Packets from a workstation or dialup with a destination of 206.191.58.1 are lost, but used to be delivered to the gateway.

All but the last difference was expected. The last difference shows we did not quite restore connectivity.

## Discussion

The above analysis was presented entirely in terms understandable to a network manager. Yet, the underpinning for the analysis is that of formal modeling and analysis. The benefits, especially that of comprehensive analysis of a network model, are clear, yet the complexity hidden. Cayenne's ability to make use of "symbolic simulation" is of particular note as it provides a basis for the comprehensive analysis and the ability to prove that networks are compliant with policy.

In an abstract for a presentation that J Moore (U. T. Austin) recently gave at the University of Pennsylvania, Dr. Moore succinctly described some of the benefits of formal modeling and analysis:

"*Computer hardware and software can be modeled precisely in mathematical logic. If expressed appropriately, these models can be executable. This allows them to be used as simulation engines or rapid prototypes. But because they are formal they can be manipulated by symbolic means: theorems can be proved about them, directly, with mechanical theorem provers. But how practical is this vision of machines reasoning about machines? In turns out that researchers in academia and industry are using mechanical theorem provers to prove important theorems about commercial microprocessor designs, including processors by AMD, Motorola, IBM, Rockwell-Collins and others. Some of these microprocessor models execute at 90% the speed of C and have had important functional properties verified. In addition, we are modeling the Java Virtual Machine and are proving theorems about JVM methods.*"

What is of particular note to the M&S community is that it is possible to have the benefits of formal models and yet still have these models nearly as efficient as simulations written in C or the like.

## CONCLUSION

This brief paper has introduced two means through which formal methods could be successfully introduced into the M&S field: lightweight formal methods and hiding complexity. Both approaches can provide significant benefits, yet reduce impediments to technology adoption.

## REFERENCES


1 R.G. Sargent, “Validation and Verification of Simulation Models”, *Proceedings, 1999 Winter Simulation Conf.*

2 D.E. Stevenson, "A Critical Look at Design, Verification, and Validation of Large Scale Simulations ", *IEEE Computational Science and Engineering* (to appear).

3 D. Jackson, “Lightweight Formal Methods”, *International Symposium of Formal Methods Europe*, Berlin, Germany, March 12-16, 2001, Proceedings.

4 S. M. Easterbrook and J. R. Callahan, "Formal Methods for Verification and Validation of partial specifications: A Case Study," *J. of Systems and Software*, vol. 40, (3), 1998.

5 D.K. Pace, “Conceptual Model Development for C4ISR Simulations”, 5th International International Command and Control Research and Technology Symposium, Dept. of Defense, 2001. http://www.dodccrp.org/2000ICCRTS/cd/papers/Track2/059.pdf

6 Defense Modeling and Simulation Office, “Conceptual Model Development and Validation”, www.msiac.dmso.mil/vva/Special_Topics/Conceptual/conceptual-pr.PDF, Nov. 30, 2000.

7 J.B. Warmer, A.G. Kleppe, The Object Constraint Language: Precise Modeling With UML, Addison-Wesley, 1998.

8 Daniel Jackson. Alloy: A Lightweight Object Modelling Notation. ACM Transactions on Software Engineering and Methodology (TOSEM), Volume 11, Issue 2 (April 2002), pp. 256-290

9 J. M. Spivey. The Z Notation: a reference manual. http://spivey.oriel.ox.ac.uk/~mike/zrm/

10 Dan Craigen, Irwin Meisels, and Mark Saaltink. Analysing Z Specifications with Z/EVES. In Industrial-Strength Formal Methods in Practice, J.P. Bowen and M.G. Hinchey (Editors), September 1999. Available through Springer-Verlag FACIT series.

11 Simon N. Foley. A Kernelized Architecture for Multilevel Secure Application Policies. In Computer Security - ESORICS 98, 5th European Symposium on Research in Computer Security, Louvain-la-Neuve, Belgium, September 16-18, 1998. Jean-Jacques Quisquater, Yves Deswarte, Catherine Meadows and Dieter Gollmann (Editors). Lec. Notes in Computer Science, Volume 1485, Springer, 1998.

12 F. Fung and D. Jamsek. Formal Specification of a Flight Guidance System., NASA/CR-1998-206915, January 1998.

13 Irfan Zakiuddin, Jim Woodcock, Michael Goldsmith and Jason Hulance. Formal Verification for Survivable Key Management Systems. Position Paper for the Third Information Survivability Workshop -- ISW-2000 October 24-26, 2000. Available through http://www.cert.org/research/isw/isw2000/index.html

14 S. Agerholm and P.G. Larsen, “Modeling and Validating SAFER in VDM-SL”, Proc., Fourth NASA Langley Formal Methods Workshop, Sept. 1997.

15 R.R. Lutz, “Reuse of a Formal Model for Requirements Validation”, Proceedings, Fourth NASA Langley Formal Methods Workshop, Sept. 1997.

16 W. Janssen, R. Mateescu, S. Mauw, P. Fennema, P. v.d. Stappen, “Model Checking for Managers”, *6th Int.l SPIN Workshop on Practical Aspects of Model Checking*, Toulouse, France, 21-24 September 1999.

17 Formal Methods Diffusion: Past Lessons and Future Prospects. R. Bloomfield, D. Craigen, F. Koob, M. Ullmann, and S. Wittmann. Proc. SAFECOM P 2000, the 19th International Conference on Computer Safety, Reliability and Security, Rotterdam, October 2000.

18 Formal Methods Technology Transfer: Impediments and Innovation. Dan Craigen, Susan Gerhart and Ted Ralston. In Applications of Formal Methods. M. G. Hinchey and J. P. Bowen (editors). Prentice-Hall International Series in Computer Science, September 1995.