Proceedings of the 2006 Winter Simulation Conference
L. F. Perrone, F. P. Wieland, J. Liu, B. G. Lawson, D. M. Nicol, and R. M. Fujimoto, eds.

THE SIMULATION PROJECT LIFE-CYCLE: MODELS AND REALITIES

Robert G. Sargent
(Panel Co-chair)

Dept. of Electrical Engineering &
Computer Science
L.C. Smith College of Engineering
and Computer Science
Syracuse University
Syracuse, NY 13244
U.S.A.

Richard E. Nance
(Panel Co-chair)

ORCA Computer, Inc.
1800 Kraft Drive
Suite 111
Blacksburg, VA 24060-6370
U.S.A.

C. Michael Overstreet

Computer Science Department
Old Dominion University
Norfolk, VA 23529-0162
U.S.A.

Stewart Robinson

Warwick Business School
University of Warwick
Coventry
CV4 7AL
United Kingdom

Jayne Talbot

Virtual Technology Corporation
5510 Cherokee Avenue
Suite 350
Alexandria, VA 22312-2320
U.S.A.

ABSTRACT

This panel session will discuss various issues regarding simulation life-cycle models. Simulation life-cycles models have received little attention, and this panel session seeks to generate interest in this topic and stimulate new ideas for development, teaching, and use of these models.

1 OVERVIEW

Nance and Arthur (2006) state that the usage of formal Software Requirements Engineering activities in Modeling and Simulation (M&S) development and analysis is minimal, at best. If one analyzes the Nance and Arthur paper closely, one also notes the paucity of work on M&S life-cycle models. In contrast, life-cycle models are a subject of considerable interest in software engineering. The purpose of this panel is to bring attention to the topic of M&S life-cycle models. By doing so, we hope to generate interest in this topic within the research, application, and education communities.

The organization Project Management Institute is concerned with managing projects of all types. Their book *A Guide to the Project Management Body of Knowledge (PMBOK Guide)* (Project Management Institute 2004) provides considerable information that is applicable to managing simulation projects including information that is applicable to life-cycle models. In this paper we are discussing the project life-cycle of developing simulation models and not the life-cycle of the simulation models, which are products. Note that a project life-cycle is different from a product life-cycle. A project has a definite beginning and a definite end, and frequently produces a product that has a life-cycle of its own. Paraphrasing or using direct information from the PMBOK Guide, a project can be divided into project phases and collectively the phases are known as a project life-cycle. A project phase is a collection of logically related project activities, usually culminating in the completion of one or more major deliverables. At the conclusion of each phase a review is usually made of each key deliverable to (a) determine if the project should continue to the next phase and (b) detect and correct errors in a cost effective way. The number of phases is usually small, say less than ten, and each project phase can have sub phases. Different project life-cycle models can be developed for each application domain by how different project phases are defined and related to each other for a project and how the review of each phase is handled.

In their survey of the literature for the use of formal software requirements engineering in M&S, Nance and Arthur (2006) found only the textbook by Robinson (2004) discusses these requirements and that M&S life-cycle models do not clearly specify where such requirements are included. Another important observation in this survey is

that little exists in the literature on M&S life-cycle models. Nance and Arthur (2006) report on the Balci-Nance M&S life-cycle model (Balci and Nance 1987), the Kreutzer (1986) model for life-cycle of a simulation project, and the Sargent (2001) model. Note that the first two models are products of the 1980's and that the Sargent model, while not developed as a M&S life-cycle model, can be used as one as pointed out by Nance and Arthur (2006).

The co-chairs of this panel have developed a set of questions from which the panelists have formed their responses in this paper. Section 2 of the paper contains the questions and subsequent sections present the responses of the panelists. The last section is the Summary.

2 QUESTIONS FOR THE PANELISTS

1. The approaches to simulation model development in simulation textbooks and in M&S life-cycle models generally follow a "waterfall" approach. How might software life-cycle models; e.g., Boehm's Spiral Model (Boehm 1988), or new software engineering methodologies; e.g., Agile Development (Ambler 2002), Synchronize-and-Stabilize (2006) contribute to improvements in M&S life-cycle models?
2. Is a single life-cycle model applicable for all M&S development projects? Would multiple life-cycle models promote confusion and lack of uniformity? Would a "core life-cycle model" with tailoring provide some uniformity but permit differences based on budgets, schedules, and/or objectives?
3. What level of detail should be expected in an M&S life-cycle model? Where is the proper boundary between a life-cycle model and the model development methodology?
4. Verification and Validation (V&V), sometimes accompanied by "acceptance" or "certification," are activities extremely important in M&S. Are the representations of these activities essentially in a life-cycle model? Should a life-cycle model include the "Real World" and its relations with the "Simulation World" (Sargent 2001)?
5. Requirements specification in M&S projects or concepts from Software Requirements Engineering are generally not discussed in simulation textbooks or included in M&S life-cycle models. How and where should requirements be included—in life-cycle models or in modeling methodology?
6. Some M&S applications involve models intended to be used for decades. Should a life-cycle model include the long term adaptation and extension typically labeled the "maintenance or sustainment" phase in software engineering? Is the proper view one of model evolution rather than a focus on model development?
7. The differences between modeling and programming are becoming "blurred" in both M&S and in software engineering with the promulgation of software development approaches such as model driven development. Is this "blurring" a positive, negative, or neutral influence? Do recently developed techniques in software engineering contain helpful guidance to assist M&S development?

3 RESPONSE OF NANCE

Response to Question 4: V&V are such essential and critical activities that a M&S life-cycle model (LCM) must portray them. The level of detail is not an issue; what is important is to show their relationship to other activities throughout the creation of a model. That is why the "Real World" and "Simulation World" interplay is so revealing in the Sargent characterization (Sargent 2001). An organization dependent on M&S for enterprise management decision support must continually draw on the comparative analysis and instructive guidance of experimentation in the *system domain* and simulation results in the *model domain*.

Response to Question 6: Since M&S is considered to be most applicable for large, complex systems studies, model evolution is the proper focus of a LCM. This claim is not to discourage or disparage prototypes and "quick and dirty" applications of M&S. Such uses rarely require the directional support embodied in a LCM; but the systems that continue for years or decades; e.g., the B-52 aircraft, desperately need a methodology and a LCM that explicitly treat the evolutionary perspective. If I may be permitted to plagiarize myself, "Software don't break, but software do rot." Extension of this assertion to models seems warranted.

Response to Question 2: Multiple standards are never a good idea. A single standard applicable to all instances of M&S applications is insufficient *unless* that standard permits tailoring. Note that "tailoring" presumes a core set of activities that are present in all instances. This core set might be small, and might extend and expand through several stages depending on project needs.

Response to Questions 1 and 5: The absence of treatment of LCMs in simulation texts devolves from the diversity of academic disciplines in which M&S is taught and used. Despite the multiple "homes" in which M&S resides, the dominant use of the technique until the 1990's was for system analysis and acquisition. The overwhelming influence of operations research and management science on the teaching of (and research in) M&S cannot be denied. One teaches and one writes about that which appeals most to him or her. For most of those in the INFORMS community that neither includes LCMs, which are relegated to a project management course, nor requirements specifica-

tions, which are assigned to software engineering. The work of one of my colleagues on the panel (Robinson 2004) perhaps is a harbinger of a sea change, but I have doubts. Yet, the influences of M&S uses for training and entertainment purposes portend a future that is difficult to predict.

The ignorance or dismissal of advances in software engineering research and practice in M&S has already exacted a heavy toll. Concepts and techniques in software requirements engineering need to be adapted and integrated in model development and sustainment. Failure to do so will perpetuate the endorsement of the "simple matter of programming" myopia that should by now be relegated to antiquity.

Response to Question 7: Treating the production of code as the last step in a modeling process is not new to those in M&S. This depiction of the abstraction resolution activities in model development hark back to the 1980's (Balci and Nance 1987). This conceptual discovery, labeled "model driven development" and subsumed by "model driven architecture," is the latest *technology du jour* in commercial software engineering. While some new wrinkles are already evident, the conclusion is not established that the emperor has changed his costume.

Closing Remarks: The creation of instructive LCMs is accomplished through an insightful blending of technical and management guidance. The recognition, identification, organization, and level of description in a LCM and a derived methodology are dictated by mutual management and technical concerns. The extent to which the M&S community ignores or avoids the role of LCMs could have consequences for its long-term viability and relevance.

4 RESPONSE OF OVERSTREET

Question 2 for the panelists stimulated the following thoughts about the use of alternative life cycle models in simulation.

Members of the software engineering community advocate different life-cycle models for software development. The alternatives may reflect the needs of different approaches to software development or may match the varying characteristics of different software projects. Two highly visible and contrasting approaches to software development are CMMI (Software Engineering Institute 2006) and Extreme Programming (Extreme Programming 2006, Wikipedia 2006). Both approaches have experienced wide and successful use; each is well supported by documentation and training, and both have supporters and detractors.

CMMI is often viewed as a more traditional approach while Extreme Programming, easily the most visible example of the Agile Programming approaches, is seen by proponents as incorporating newer better ways of developing software.

This is not an appropriate place to evaluate the merits of the two approaches, a difficult and complex task, but some contrasting aspects of these approaches are relevant to the current discussion. McBreen (2003), in his critique of Extreme Programming, points out that CMMI was at least initially influenced by problems arising in the development of "traditional" software systems. Historically, many early software systems replaced existing manual systems (e.g., payroll or inventory). Users had much experience with the existing system so requirements for their replacements were generally relatively easy to identify. Conventional CMMI assumes the existence of reasonable complete and stable requirements.

In contrast with the past, many new systems provide capabilities not previously possible. Thus we often have limited or no experience in how the proposed application should function. This is true of many highly interactive web applications. Hence providing reasonable and stable requirements before coding begins can be challenging or impossible.

Advocates of Extreme Programming assert that it has evolved in part to address software applications where requirements are poorly understood at the beginning of a project and are thus likely to evolve substantially during implementation. Systems built using the Extreme Programming approach are intended to be "agile" during their development in the sense that as users gain experience with partial implementations of the new system and better understand their needs, the evolving system is more readily changed to meet the improved identification of the most important and most desirable features.

As one example of the difficulty in identifying requirements in new application domains, readers who were developing code during the transition from keypunches and decks of cards to time-sharing systems providing interactive editors may recall how long it took to identify desirable editor features. The earliest text editors were often referred to as "glass keypunches" because the available editor features basically replicated how one modified an existing program deck using a keypunch.

Identifying appropriate processes for use in developing simulations is one aspect of simulation that makes it an interesting problem area. This is due in part to the variety of purposes for which simulation models are built: some simulations are use once and thrown away; others see extensive use over relatively long periods. Additionally, in some simulations, the model on which the simulation is based is well understood at the beginning of a project and model changes are minimal during development. In others, development of an acceptable model is an integral part of the simulation process and repeated modification of a model or its implementation is key to the development process.

A central theme of simulation is education: sometimes a simulation is built to educate users (for example, to assist

in planning by allowing users to ask "what-if" questions, or to assist in training individuals for a new environment). At other times, it is the modelers who are seeking to be educated: they are building and evolving models to improve their understanding of a system of interest. Sometimes the insights gained through building a valid model of a system are more valuable to modelers than the functional simulation. This can be true for both planners and scientists evaluating proposed theories.

Different simulation projects have different software development needs; some simulation projects such as OneSAF (SAIC 2003) involve the development, use and reuse of large code bases (sometimes more than 1,000,000 lines of code). Success of such projects requires reliance on a sound, proven software development process. CMMI-like approaches seem appropriate for projects of this magnitude since the simulation is likely to have a relatively long life both for its intended use (training in the case of OneSAF) and through likely reuse of major software components that are part of the implementation.

Other uses of simulation have different development needs and use of more agile software development techniques seem appropriate. For example, when a simulation is used to assist in the development a new design, the simulation may be discarded after design decisions are made. In a project where simulation was used to develop and evaluate the specification of a parallel network protocol (Foudriat et al. 1991), it is only a sight exaggeration to state that the same simulation was never run twice. The simulation process used (somewhat simplified) was one of continual code change and consisted (in part) of the steps: run the simulation, examine the data produced by the simulation, identify changes to be made to the model or code based on this data analysis and then repeat. The model or implementation was changed in each iteration for one of three reasons: insight was gained into the behavior of the protocol and the model was modified to improve protocol performance (for example, to eliminate a bottleneck identified by the simulation), defects were found in the model or its implementation, or the output was confusing so that code was changed to provide more details about what happened during execution (since either the implementation was defective or our understanding of the protocol behavior was defective). So the ability to make repeated changes was key to the success of the simulation approach and one reason why simulation was the technique of choice for analyzing the new protocol.

While both CMMI and Extreme Programming can be tailored to some extent to meet the different needs of different simulation projects, the Extreme Programming approach seems more appropriate for some smaller quickly evolving simulation projects. CMMI seems better for large, multi-team, multi-site long-lived simulation projects.

One view in the software engineering community is that the various life-cycle models were developed based on different perceptions of the most important risks associated with software projects. It is admittedly an over-simplification to state that CMMI reflects concerns with budgets and deadlines, Extreme Programming with creating appropriate software solutions in new application domains.

It is our belief that since different simulation projects often have different risks, the choice of an appropriate life cycle model for a project should be based on that model's ability to address the most important risks. Thus multiple life cycle models are useful in simulation; the choice of which life cycle model is appropriate for a particular simulation depends on characteristics of that projects.

5 RESPONSE OF ROBINSON

The primary question I would like to address is question 2. In doing so, however, I will also touch upon issues related to question 1.

We should first recognize that simulation is not a uniform field. Simulation modellers adopt diverse approaches to developing and using simulation models, and therefore they follow (and require) quite different life-cycle models.

Robinson (2002) identifies three modes of simulation practice. The first is referred to as 'simulation as software engineering' and centers on the provision of a product, in this case a simulation model. This involves large models, whose prime motivation is representation of a real system. These models are often used over many years. The development of the model involves multiple modellers, developing code in a programming language, whose predominant skill is software development. It may take years (certainly many person years) to develop the model. Verification and validation is performed by the modellers and on some occasions by independent assessors. In this mode of practice there is much interest in model reuse and distributed simulation. Such an approach to modelling is commonplace in the military.

The second mode of practice is described as 'simulation as a process of organizational change'. In this mode the work centers on the provision of a service, with the prime motivation being to intervene in a problem situation. This involves small scale models that are used for a short period and then normally thrown away. The model is developed by a lone modeller, typically using a simulation package, and skilled in modelling. Model development requires only a matter of weeks. Assessment of the model is carried out jointly between the modeller and the client. There is some interest in model reuse (e.g. generic models) and distributed simulation, but this is only limited as model development and simulation are generally not onerous. This approach to simulation is common in business.

The final mode of practice, 'simulation as facilitation', is about understanding and provoking debate about a problem situation. These models could be described as 'quick-

and-dirty' with no expectation of being used in the long-term. Again the model is developed by a lone modeller using a simulation package, but now with the close involvement of the client throughout. The predominant skill required is process management. The model may be developed in a matter of hours and it is validated with respect to the extent that it aids understanding of the problem and debate. Indeed, even if the fidelity of such models is very low, they could still be considered valid if they are seen as useful. Model reuse is beneficial to the extent that it can aid rapid model development. Distributed simulation is almost certainly of no specific interest. Such an approach to simulation has become possible in recent years through the availability of visual interactive modelling systems, albeit that these are still far from enabling live model building in all but the simplest of situations.

Given these modes of practice we can reflect on the M&S life-cycle models that are implied. Simulation as software engineering, as the name implies, bears close correspondence to software engineering (although Balci (1994) identifies some significant differences). As a result, investigating the adoption of software engineering life-cycle models would seem a fruitful line of enquiry. Any such investigation would, of course, have to take account of the differences arising from the simulation arena. Just as software engineering has moved on from the original conceptualization of a waterfall model, so too, simulation as software engineering needs to do so. At present the model of Balci (1985) provides what is probably the most recognized description of the simulation life-cycle in the software engineering domain.

One of the critical issues in describing the life-cycle model in the first mode of practice is the extent of iteration between phases in the modelling process. Such projects are less amenable to iteration due to their complexity. With many participants, iteration would require a large investment in coordination and communication between all parties; something that the more recently developed software life-cycle models might help to create. On the other hand, iteration allows for a less well specified problem and model; something which is beneficial in some environments. Whichever, the benefits of iteration should be weighed against the cost. The conclusion may be that the waterfall model with its limited scope for iteration is appropriate, but this will not always be the case. As such, a number of simulation life-cycle models should be proposed, and their relative strengths and weaknesses identified.

Moving to simulation as a process of organizational change, software engineering probably has fewer direct lessons for simulation modellers. That said, ideas such as rapid prototyping, for instance, would certainly provide benefits (Powell 1995; Pidd 1999), enabling faster and more iterative model development. In this domain, I have found Sargent's original model (Sargent 1982), as adapted by Landry et al. (1983), the most convenient way of describing the simulation life-cycle (Robinson 2004). A particular strength is the emphasis on iteration and the parallel activities of verification and validation during model development and use.

In the third mode of practice, simulation as facilitation, a very different life-cycle model is required. In this domain the simulation model is becoming more incidental, acting as a catalyst for understanding and debate, but not necessarily deriving results directly about potential improvements and solutions to the real world problem. This mode of practice bears some resemblance to the ideas found in 'soft' operational research or problem structuring methods (Rosenhead and Mingers, 2001). These ideas are founded on the following principles:

- Ability to deal with unclear, multiple and conflicting objectives
- People are seen as active subjects in an intervention, not as passive objects
- Facilitating debate around a problem situation
- Acceptance of uncertainty
- Reduced data demands
- Focus on seeking improvements to a problem situation not a solution, since the goal of a solution is often unattainable

Robinson (2001) describes an example of simulation as facilitation. In doing so a life-cycle model is proposed based on the work of Lane and Oliva (1998) in system dynamics. This is shown in Figure 1. The key processes identified in the simulation life-cycle are: conceptualization, model development and facilitation. Under each of these are a number of sub-processes. Iteration between processes is shown through the double arrows. Validation is identified as a continuous process that is carried out throughout the life-cycle, albeit that there is a specific point where validation of the completed model takes place.

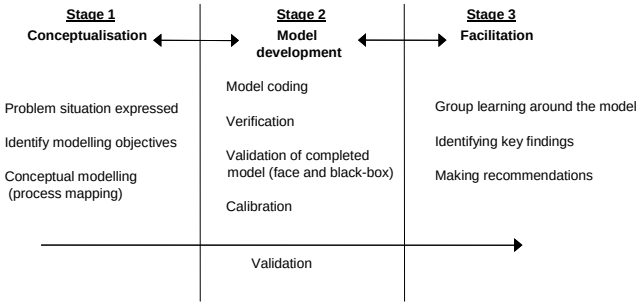

Figure 1: Life-Cycle Model for Simulation as Facilitation (Robinson 2001).

In describing these three modes of simulation practice we are able to see that one life-cycle model is unlikely to

be appropriate to all. Indeed, even within a mode of practice a range of life-cycle models may apply. This could be a result of personal preference, the modelling domain or the existence of a range of sub-modes of practice. Indeed, the modes of practice identified are not meant to be seen as discrete, but part of a continuum of practices from software engineering through to facilitation. As a result, I would suggest that a range of M&S life-cycle models should exist and that further, their relevance to a mode of practice should be explicitly identified.

6 RESPONSE OF SARGENT

Response to Question 1: I believe that the development of simulation models is, in general, iterative and as a result "waterfall" models are usually not appropriate. As discussed in the responses below, different M&S life-cycle models can and should be developed. Furthermore, M&S life-cycle models should be included in simulation textbooks.

Response to Questions 2 and 3: Because I believe in the KISS (Keep It Simple 'Sarge') Principle, I believe that M&S life-cycle models should be simple and each model should use a graph to show the relationships among the phases and any sub phases. This approach to M&S life-cycle models probably results in the development methodologies containing more information than those for complex life-cycle models. (I am of the opinion that life-cycle models should be developed first and than the development methodologies developed for each life-cycle model.)

I believe it is desirable to have a single M&S life-cycle model; however, I believe it is difficult to have a single model that is not complex. Large simulation projects usually use multiple teams of people to develop a simulation model over several months and these projects have many more requirements and activities than a small project that uses, e.g., only a single individual to develop a simulation model over a few weeks. Having a single model to handle the range of requirements and activities for the different size projects would in all likelihood require more complexity than individual life-cycle models developed for different size and types of simulation projects.

I believe a simple life-cycle model containing only a few phases can show the "high" level development of a simulation modeling project. However, I am of the opinion that such a "high" level model would not be adequate for large scale simulation projects because it would not contain the necessary phases (and perhaps sub phases) and their relationships, requirements, and deliverables that are needed for large size projects. Thus it might be better to have a few, say less than five, different simulation life-cycle models including one that contains only the high level phases.

Response to Question 4: I strongly believe that M&S life-cycle models should contain V&V. While at least one V&V phase should be included in any M&S life-cycle model, the methods and/or techniques used should be part of the methodology and not part of the life-cycle model. I further believe both the "real world" and the "simulation world" should be included in an M&S life-cycle model in some form. In both of the models that I use in my V&V work (Sargent 2005), I have the "real world" and the "simulation world" represented in some way.

Response to Question 5: High-level software requirements should be included in life-cycle models for large size simulation projects; however, they probably are not needed for small size simulation projects, especially for simulation projects where a simulation language that has modeling capability is being used.

Response to Question 6: One of the critical questions that needs to be addressed when starting a simulation project is (a) how long is the simulation model going to be used and (b) if appropriate, who is responsible for maintaining the model including keep it current with the system that it is representing. There is the management of developing a simulation model, a project that should have a definite ending, that results in a simulation model, a product. (In this panel session we are discussing life-cycle models of developing simulation models, which are projects.) Whether the model development team is responsible for maintaining or sustaining a simulation model for some period of time *or* a different team is responsible depends on the simulation project and the environment in which the simulation model is going to be used in. For example, if a simulation model is being developed for the planning of a new factory, then it may be most appropriate for the model development team (the project team) to handle the simulation model until the factory is under operation and then the simulation model turned over for maintenance to some other team (the product team) if the simulation model is going to be used in operations. (See Banks and Gibson (1998) for an example of case where the 'maintenance' of a simulation model was not performed during the design and start up of an industrial system resulting in the simulation model not being representative of the system as the system evolved.) Thus an M&S life-cycle model should allow for the 'maintenance or evolution' of a simulation model for a determined period.

7 RESPONSE OF TALBOT

I have spent my engineering career supporting the Department of Defense M&S community and have been a provider of M&S products that support a variety of DoD applications including analysis, concept exploration, system testing, and training. The following observations and comments are based on my collective experiences in providing M&S solutions to DoD.

My response to question 1 follows: The waterfall approach described in M&S textbooks as the traditional life-cycle model for M&S products is flawed because it fails to capture the iterative nature of the requirements specification phase. In my experience, detailed, actionable requirements for M&S projects are often not known at the onset of the project. A life-cycle model should reflect the process of gleaning and refining requirements. Agile Modeling (Ambler 2002) and the WinWin model (Boehm 1998) do a fair job at capturing what, in practice, turns out to be a collaborative and iterative process to elicit requirements.

It seems imperative and logical that requirements specification be well understood prior to starting development in an M&S project. However, it has been my experience in the DoD M&S field that complete and testable requirements are rarely well understood by the customer. At the onset of a project, customers approach developers with objectives or goals for M&S products rather than requirements. In addition, customers rarely comprehend the software complexities and limitations associated with their M&S project objectives. Lastly, to further complicate the requirements definition and capture, it is often acknowledged up front that the goals and objectives are subject to change or expected to evolve over the course of the development. The ability to support changing requirements therefore becomes a requirement.

In several successful M&S procurements that I have observed, a process to compensate for the lack of a formal requirements specification phase has been used. It consists of rapid prototyping to elicit customer requirements resulting in concrete customer objectives. The five steps described below are tightly coupled and iterative and are *not* inclusive of all steps required to build validated M&S products. They focus on meeting customer needs when a lack of formal requirements definition exists.

1. Discussion between customer and developer about objectives/goals/requirements: The objective of the discussion is get a shared understanding of a vision of the end product with an understanding that detailed requirements needed to complete the project may not be known at this time.
2. Story boarding: Taking from what is learned from the objectives discussion, design a concept, view or picture of the resulting product that can be reviewed by the customer and illicit more concrete requirements. In a course way, this becomes a requirements set.
3. Customer review: Review the concept, view or picture with the customer to gain concurrence that the project vision is on track and uncover issues or discrepancies with the addition of new features or functionality. Form an agreement as to what is to be implemented in the next phase.
4. Implementation: Develop the full functionality agreed to in the customer review creating a prototype of the final product, albeit incomplete.
5. Demonstration: Demonstrate the agreed upon functionality implemented in the previous phase to ensure it is in keeping with what was agreed to in the review phase.

Over time, the process is repeated and more requirements are unearthed. This process requires a regular communication rhythm between the customer and the developers to keep the process going and is thereby reliant on a committed customer. I have seen this process work well with a development team of five and iteration cycles lasting one week.

The challenges in this approach are several. This process depends on an underlying M&S architecture that is designed with flexibility and scalability. Without this, the final product has the potential to fail to expand to meet the emerging requirements as they are uncovered. Focus on the upfront phases, requires discipline in testing. Often end-to-end testing is not given proper concentration and must be accounted for in the overall process. Lastly, reliance on customer feedback and verbal communication to create requirements can be dangerous as compared to basing development on a formal set of written requirements. Tools to carefully document customers' communication and feedback are critical to this iterative process.

This process is not ideal for all M&S development efforts and I suspect does not scale well to large M&S product procurements. It has proven successful in a number of M&S procurements where the formal requirements were not well known at the onset of the project and hope that the customer would independently refine them over time was nil.

Interestingly, in spite of customers' inability to clearly specify requirements, I have come across very few customers that require any type of process or methodology for M&S development. Given that procuring M&S products have continued without adherence to an accepted process or methodology, I suspect that even if a process were promulgated and promoted, that DoD would not embrace it readily. In the end, customers are reliant on developers to "do a good job" and I suspect that conscientious developers will employ an M&S development process where it makes financial sense for them to do so. If they can "get away" without it, I suspect they will continue to do so.

8 SUMMARY

Let us first look at the questions. Question one was regarding types of life-cycle models and Question two was whether there should be more than one life-cycle model. The remaining five questions were regarding specific de-

tails of life-cycle models. Each panelist could select which questions and how many questions to response to.

It is interesting that the three panelists who were not the co-chairs choose to respond to only one question and that was either Question one or Question two. Perhaps this indicates the paucity of work on M&S life-cycle models. Both co-chairs responded to both Questions one and two and some of the other questions.

A few conclusions regarding the "answers" to the questions can be made.

- The panelists agree that "waterfall" life-cycle models are not appropriate except in some special cases.
- Most of the panelists believe that more than one life-cycle model is needed. One panelist believes that only one model (with tailoring allowed) should be used. This will perhaps lead to an interesting discussion during the panel session.
- Both co-chairs agree that V&V should be included in life-cycle models.
- Both co-chairs agree that the "Real World" and the "Simulation World" should be included in life-cycle models in some form.
- The two co-chairs have differences in their "answers" to the other questions that they both responded to.

Based on the panelists' responses, much work is needed on M&S life-cycle models by the research, application, and education communities.

REFERENCES

Ambler, S. W. 2002. *Agile Modeling: Effective Practices for XP and RUP*. New York: John Wiley & Sons.

Balci, O. 1985. Guidelines for successful simulation studies. Technical Report TR-85-2, Department of Computer Science, Virginia Tech, Blacksburg, VA.

Balci, O. 1994. Validation, verification, and testing techniques throughout the life cycle of a simulation study. *Annals of Operations Research* 53: 121-173.

Balci, O. and R. E. Nance. 1987. Simulation development environments: a research prototype. *Journal of the Operational Research Society* 38: 8, 753-763.

Banks, J. and R. Gibson. 1998. Simulation evolution, *IIE Solutions*, November 1998, 26-29.

Boehm, B. 1988. A spiral model of software development and enhancement. *IEEE Computer* 21: 6, 61-72.

Boehm, B. 1998. Using the win-win spiral model: a case study. *IEEE Computer* 31: 7, 33-44.

Extreme Programming. 2006. <http://www.extremeprogramming.org>. [accessed July 1, 2006].

Foudriat, E. C, K. Maly, C. M. Overstreet, S. Khanna, F. Paterra. 1991. A carrier sense multiple access protocol for high data rate ring networks. *Computer Communications Review*. pp.59-70.

Kreutzer, W. 1986. *System Simulation – Programming Styles and Languages*. New York: Addison Wesley.

Landry, M, J.L. Malouin, and M. Oral. 1983. Model validation in operations research. *European Journal of Operational Research* 14 (3): 207-220.

Lane, D.C. and R. Oliva. 1998. The greater whole: towards a synthesis of system dynamics and soft systems methodology. *European Journal of Operational Research* 107: 214-235.

Kreutzer, W. 1986. *System Simulation – Programming Styles and Languages*. New York: Addison Wesley.

McBreen, 2003. *Questioning Extreme Programming*, New York: Addison Wesley.

Nance, R. E. and J. D. Arthur. 2006. Software requirements engineering: exploring the role in simulation model development. In *Proceedings of the 2006 Operational Research Society Simulation Workshop (SW06)*, eds., J. Garnett, S. Brailsford, S. Robinson and S. Taylor, 117-127. The Operational Research Society, Birmingham, United Kingdom.

Pidd, M. 1999. Just modeling through: a rough guide to modeling. *Interfaces* 29 (2): 118-132.

Powell. S.G. 1995. Six key modeling heuristics. *Interfaces* 25 (4): 114-125.

Project Management Institute. 2004. *A Guide to the Project Management Body of Knowledge (PMBOK Guide)*, 3rd edition, Project Management Institute, Four Campus Boulevard, Newtown Square, PA 19073-3299.

Robinson, S. 2001. Soft with a hard centre: discrete-event simulation in facilitation. *Journal of the Operational Research Society* 52: 905-915.

Robinson, S. 2002. Modes of simulation practice: approaches to business and military simulation. *Simulation Modelling Practice and Theory* 10: 513-523.

Robinson, S. 2004. *Simulation: The Practice of Model Development and Use*. Chichester, West Sussex, England: John Wiley.

Rosenhead, J. and J. Mingers. 2001. *Rational Analysis for a Problematic World Revisited, 2nd ed*. Wiley, Chichester, UK.

SAIC. 2003. The future of simulation. *SAIC Magazine*, Summer 2003. Available at <http://www.aic.com/news/saicmag/2003-summer/simulation.htm>. [accessed July 1, 2006].

Synchronize-and-Stabilize. 2006. <http://searchwebservices.techtarget.com/sDefinition/0,,sid26_gci922408,00.html>. [accessed July 13, 2006].

Sargent, R.G. (1982). Verification and validation of simulation models. In: *Progress in Modelling and Simulation* (Cellier, F.E., ed.). Academic Press, London: 159-169.

Sargent, R. G. 2001. Some approaches and paradigms for verifying and validating simulation models. In *Proceedings of the 2001 Winter Simulation Conference*, eds. E. Yucesan, C.-H. Chen, J. L. Snowdon, and, J. M. Charnes, 106-114. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Sargent, R. G. 2005. Verification and validation of simulation models. In *Proceedings of the 2005 Winter Simulation Conference*, eds. M. E. Kuhl, N. M. Stieger, F. B. Armstrong, and J. A. Jones, 130-143. Piscataway, NJ: Institute of Electrical and Electronics Engineers.

Software Engineering Institute. 2006. <http://www.sei.cmo/cmmi>. [accessed July 1, 2006].

Wikipedia. 2006. <http://en.wikipdia.org/wiki/Extreme_programming>. [accessed July 1, 2006]

AUTHOR BIOGRAPHIES

RICHARD E. NANCE is an Emeritus Professor of Computer Science at Virginia Tech and a Faculty Fellow in Systems Engineering at Stevens Institute of Technology. Dr. Nance is also Chief Scientist, Orca Computer, Inc. He received B.S. and M.S. degrees from N.C. State University in 1962 and 1966, and the Ph.D. degree from Purdue University in 1968. He has served on the faculties of Southern Methodist University and Virginia Tech, where he was Department Head of Computer Science, 1973-1979. Dr. Nance held research appointments at the Naval Surface Weapons Center (1979-80) and at the Imperial College of Science and Technology (UK). Within ACM, he has chaired two special interest groups: Information Retrieval (SIGIR), 1970-71 and Simulation (SIGSIM), 1983-85. He has served as Chair of the External Activities Board and several ACM committees. He is the author of over 150 papers on discrete event simulation, performance modeling and evaluation, computer networks, and software engineering. Dr. Nance has held several editorial positions and was the founding Editor-in-Chief of the *ACM Transactions on Modeling and Computer Simulation*, 1990-1995. He served as Program Chair for the 1990 Winter Simulation Conference. Dr. Nance has received several awards for his editorial and professional contributions. In 1996, he was named an ACM Fellow. He is a member of Sigma Xi, Alpha Pi Mu, Upsilon Pi Epsilon, ACM, and INFORMS. His e-mail address is <nance@vt.edu>.

C. MICHAEL OVERSTREET is an Associate Professor of Computer Science at Old Dominion University. A member of ACM and IEEE/CS, he is a former chair of SIGSIM, and has authored or co-authored over 80 refereed journal and conference articles. He received a B.S. from the University of Tennessee, an M.S. from Idaho State University and an M.S. and Ph.D. from Virginia Tech. He has held visiting appointments at the Kyushu Institute of Technology in Iizuka, Japan, and at the Fachhochschule fűrs Technik und Wirtschaft in Berlin, Germany. His current research interests include analysis of simulation models to enhance model understanding and static code analysis. Dr. Overstreet's home page is <www.cs.odu.edu/~cmo>. He can be reached by e-mail at <cmo@cs.odu.edu>.

STEWART ROBINSON is Professor of Operational Research at Warwick Business School. He holds a BSc and PhD in Management Science from Lancaster University. Previously employed in simulation consultancy, he supported the use of simulation in companies throughout Europe and the rest of the world. He is author/co-author of three books on simulation. His research focuses on the practice of simulation model development and use. Key areas of interest are conceptual modelling, model validation, output analysis and modelling human factors in simulation models. His email address is <stewart.robinson@warwick.ac.uk> and his Web address is <www.btinternet.com/~stewart.robinson1/sr.htm>.

ROBERT G. SARGENT is a Professor Emeritus of Syracuse University. He received his education at The University of Michigan. Dr. Sargent has served his profession in numerous ways including being the General Chair of the 1977 Winter Simulation Conference, serving on the WSC Board of Directors for ten years and chairing the Board for two years, being a Department Editor for the *Communications of the ACM*, holding the Presidency and other offices of what is now the INFORMS Simulation Society, and serving as President of the WSC Foundation. He has received several awards for his professional contributions including the INFORMS Simulation Society Lifetime Professional Achievement Award and their Distinguished Service Award, and is a Fellow of INFORMS. His current research interests include the methodology areas of modeling and of discrete event simulation, model validation, and performance evaluation. Professor Sargent has published extensively and is listed in Who's Who in America and in Who's Who in the World. His e-mail is <rsargent@syr.edu>.

JAYNE E. TALBOT attended the University of Virginia and was trained as an electrical engineer. She started her career in 1985 with the Army's Night Vision Lab and the Environmental Research Institute of Michigan working on the physics-based modeling of infrared systems. She joined MITRE in 1992 and worked on a large variety of M&S programs supporting the training and analysis communities. Jayne is currently employed by the Virtual Technology Corporation, a distributed simulation company, in 2002 as a Group Manager. She has responsibilities for M&S programs that support hardware and software joint testing, system concept exploration and analysis, and military staff training.