

SOFTWARE REQUIREMENTS ENGINEERING: EXPLORING THE ROLE IN SIMULATION MODEL DEVELOPMENT

Richard E. Nance
Orca Computer, Inc.
Blacksburg, VA 24060
nance@vt.edu

James D. Arthur
Department of Computer Science
Virginia Tech
Blacksburg, VA 24060
arthur@vt.edu

ABSTRACT:

The utility and benefits of a well-defined requirements engineering process are cited in many articles describing software engineering research and similar industry studies. In this paper we consider to what extent the M&S community employs software requirements engineering (SRE) in the model development process. We motivate the desirability of having an SRE component by first outlining current SRE models and their touted benefits. To help formulate a reasoned conclusion, we examined several recent M&S books and three M&S Life-cycle models for references to requirements in the M&S activities. Our findings, also outlined in this paper, indicate that the use of any formal SRE activities within M&S development and analysis is minimal, at best. Nonetheless, our investigation has also uncovered some recent changes that point to a rise in the awareness of the need for a more formal, well-defined approach to software requirements engineering in the M&S model development process. We note those changes and some additional opportunities for increased SRE use.

Keywords: Requirements, Conceptual
Modeling, Model Development

1. INTRODUCTION

The objective of this paper is to explore the following question:

Do modeling and simulation (M&S) studies employ software requirements engineering (SRE), implicitly or explicitly, in the model development process?

By “explore” we do not mean simply to answer the question. In fact, a categorical answer is not possible. What is sought here is to examine what

is entailed in the engineering of software requirements, to identify where such activities are performed or not in evidence in M&S model development methodology, and to note changes that might ensue from the increased use of SRE. As a precursor, we address the motivation for producing an answer to the question and the natural follow-up question: Why is SRE of importance to individual M&S projects or to the community in general?

1.1 BACKGROUND

No area within software engineering or software systems development has seen more research activity within the last decade than the software requirements phase. Interestingly, the emphasis in software systems research appears to follow a reverse of the development path. Software systems research had an early focus on the testing phase, then a transition is seen to code generation, followed by program organization and design methodology, and for the past ten or more years the sights have centered on the requirements phase. A question prompted by this observation is: To what degree have the simulation modeling phases followed a similar pattern?

Explanations of the reverse path of research emphasis are not difficult to produce.

1. The path reflects the movement from more concrete (possibly simpler) representations of a problem solution to the more abstract (more difficult).
2. Progress in the latter stages continues to affirm the importance in time and cost of errors in the early stages.
3. Increasingly, the data from practical project experience, captured through the adoption of development methodologies, reveals the high cost incurred by incomplete and incorrect

requirements and the impact of requirements changes (requirements volatility).

The co-dependency of simulation modeling methodology and software engineering is an acknowledged fact that can be traced to the infancy of both, which is generally about 1958 for the former and 1968 for the latter. Tocher's General Simulation Program (GSP), a collection of routines recognized by him as used with parameter specifications or minor modifications in almost every application of the technique, is a very early example of code reuse [1]. Conceptual advances in modeling methodology have exerted major influences in software engineering, e.g. the use of abstract data types, and the object-oriented paradigm in software development that are traceable to SIMULA 67. Likewise, progress in software engineering is quickly reflected in simulation, e.g. external access compilers, the adoption of programming language control constructs, and graphical interfaces, to name but a few.

The interplay between simulation modeling methodology and software engineering could be characterized as mutual influences affecting different domains of the abstraction continuum.

- The influence of software engineering is evident in the more concrete areas of implementation – the programming to produce an executable simulation model representation.
- In contrast, simulation modeling methodology has contributed to raising and resolving issues surrounding the more abstract domain of capturing and expressing the architecture and design of software systems.

Evidence supporting the first claim is provided in the genealogical tree for simulation programming languages (SPLs) [2] that shows the emergence of variants (or dialects) for the original SPLs in the branches for GPSS, SIMULA, and GASP. The second claim is supported by the slow but sure emergence of models and modeling as serving a major function in software development. Interesting speculations regarding the relationship between modeling and programming dot the computing landscape over the past 25 years (see [3, 4]), and apparently Model Driven Development (MDD) is now one of the “hot” topics on the software horizon [5].

1.2 ORGANIZATION

In Section 2 a brief overview of SRE is given, noting the effect of the emphasis in delineating the activities comprising the requirements phase. The role of requirements in current M&S development is examined from multiple perspectives in Section 3. Section 4 offers a re-examination of the M&S development process giving attention to areas where SRE activities could or should be employed. A concluding summary is provided in Section 5.

2. SOFTWARE REQUIREMENTS ENGINEERING

The engineering of requirements is a necessary function in the production of a software system. While requirements generation (earlier termed “analysis”) is identified from the beginning as a part of the software development life cycle (SDLC), its importance is slow to draw recognition. An explanation of this observation follows, and the emergence of the recognition of its importance is briefly traced.

2.1 A BRIEF REVIEW OF SOFTWARE PROCESS DEVELOPMENT

The sequential version of the Waterfall model [6], shown in Figure 1, is the earliest description of the software development life cycle.

The objectives of each phase are described by Lobo [7] as:

- **Requirements Analysis:** Entailing understanding of the function, behavior, performance and interfaces of the proposed system. In addition, the software requirements are documented and reviewed by the customer.
- **Design:** Translating requirements into a system representation that can be assessed for quality before coding begins.
- **Coding:** Transforming design into machine-readable (executable) representation.
- **Testing:** Conducting tests to uncover errors in code and to ensure that the results produced are correct for a given input.
- **Maintenance:** Accommodating and incorporating necessary changes in the software.

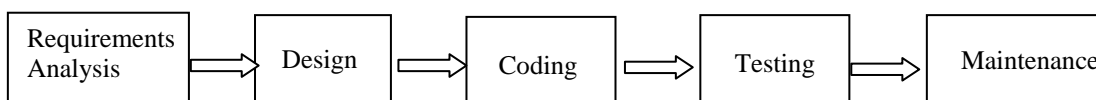


Figure 1: The Sequential Waterfall Model

Numerous modifications to the waterfall model follow: (1) incorporating feedback both within and among the phases, (2) recognizing that phase overlaps are necessary, and (3) prescribing the verification activities that assure the phase-to-phase transformations are made properly. While inadequacies lead to continuing modifications, the identification of the major software development phases forms the core of the subsequent model variations that include the iterative enhancement model [8], the prototyping model [9], spiral model [10], and extreme programming [11]. Despite the numerous refinements leading to model variations, the failure to adequately address requirements creep (the addition of or change in requirements as the project progresses) remains a persistent problem [12].

With the advent of the SDLC models, the phases in the development process become more apparent. However, the activities and techniques defining each phase remain only vaguely clarified, resulting in the software engineer having difficulties in carrying out the process. Hence, refining the description of activities comprising the SDLC phases provides essential guidance on performing these phases during the development process.

Although logic suggests that attention should be directed towards the first SDLC phase (requirements analysis), for the reasons explained above, the examination emphasis begins with the end product, and phase refinement historically proceeds in reverse order [13]. Initiating with maintenance and testing also reflects an economic influence since studies show that 60-70% of the product life cycle is spent in the maintenance phase [14]. The emphasis on testing produces techniques for black box and white box testing that are used extensively for the detection on errors in code [15]. Tools to automate testing methods are also developed.

The shift in attention to the programming phase occurs with subtle effects. Code and process analysis, sometimes conducted during testing, reveals the ad-hoc nature of the development processes, making the code difficult to comprehend. Hence, coding guidelines are produced to improve the readability, understandability and testability of the code. Integrated Development Environments (IDEs)¹ emerge to assist programmers in writing and debugging the code.

¹ IDEs provide an integrated set of tools and artifacts supporting a process; e.g. Microsoft's Visual Studio

Following the period of emphasis on the coding phase, the attention shifts to the design phase. The design phase is attacked by dividing the high-level activities into low-level components in the application of "step-wise" refinement, (solving complex problems by decomposition into smaller units). Additionally, new design approaches, such as the object-oriented paradigm, are proposed as adding better structure to the design and the code, increasing understandability and maintainability. Notations, such as those proposed by the Unified Modeling Language (UML), and supporting tools are advanced to improve representations utilized during the design phase [16].

Requirements analysis is the final phase to receive the focused attention of the software engineering community. In all likelihood, the somewhat inappropriate name "Requirements Analysis" has contributed to the delayed attention, creating a perception that activities such as needs analysis and requirements elicitation are minor [17].

2.2 THE FOCUS ON REQUIREMENTS

The examination of the requirements analysis phase has produced a flurry of activity, and a number of techniques and methods are currently drawing some attention. The brief descriptions that follow are taken with minor changes from [18].

2.2.1 SOFTWARE PRODUCT PLANNING THROUGH REQUIREMENTS TRIAGE

Davis [19] contends that "knowing what not to build is as important as knowing what to build." For a software development effort to be successful, he advocates "software product planning" through Requirements Triage to help elicit the "right" product features. Requirements Triage considers factors such as price, market, cost, time-to-market, market size, feature mix, market penetration, revenues, profits and return on investment as keys to project success. Its principal activities include Risk Analysis, Cost & Schedule Estimation, Price Analysis, Market Analysis and a "Feature Triage."

The triage process is iterative and begins by eliciting potential product features that, in turn, are categorized and prioritized based on their relative importance, cost and risk. Using the set of available resources as a guide, decisions are then made as to which features are to be included in the product and which are to be excluded. If an acceptable compromise is achieved, then software product planning is accomplished. If, on the other hand, a constraint is exceeded or a conflict

arises, one or more of the evaluation criteria must be adjusted to find a satisfactory balance.

2.2.2 KNOWLEDGE-LEVEL PROCESS MODEL OF REQUIREMENTS ENGINEERING

The Knowledge-Level Process Model of Requirements Engineering is a set of abstractions that represents requirements elicitation, manipulation, and maintenance processes [20]. For each of these processes, the corresponding third layer of abstraction is most instructive. More specifically, the third layer is a decomposition of a higher-level abstraction, and depicts a set of sub-processes related through input/output specifications. The “elicitation” process stresses gathering initial problem descriptions, domain knowledge, and requirements and scenarios. The “manipulation” process emphasizes requirements and scenario quality, e.g., consistency and necessity. Within this process, requirements and scenarios are reformulated to reflect a (semi-) formal representation. The relationships among requirements and scenarios are also established. The principal goal of the “maintenance” process is to produce documents in which requirements and scenarios are described, as well as the relationships among them. The Knowledge-Level Process Model provides a useful approach to process modeling through (a) its decomposition based on “separation of concerns” (facilitating a deeper understanding of the process as a whole), and (b) its realization of a generally applicable characterization.

2.2.3 AGILE MODELING

Agile Modeling (AM) is a practiced-based approach to modeling and documenting software-based systems [21]. It stresses the development of multiple representations (or models) of system components. Those “viewpoints” are based on the perceived needs of the stakeholder(s). AM embraces a collection of principles, such as assuming simplicity and embracing change, because requirements do, in fact, change throughout the modeling process. A set of practices that support those principles is also defined within AM.

The AM can be applied to requirements, analysis, architecture and design. From a requirements perspective, AM recognizes project stakeholders as the definitive source of requirements. In collaboration with the developer, the stakeholders plays an *active* role in identifying, defining and prioritizing requirements, and to accept (or reject) suggested ones from the developer. Iteration is the practice that is used to refine the high-level requirements within the bounded project scope.

2.2.4 THE WINWIN SPIRAL MODEL

The WinWin Spiral Model extends Boehm’s original Spiral Model to include elements that promote “Theory W” [22]. Theory “W” maintains that making winners of the system’s key stakeholders is a necessary and sufficient condition for a project’s success. A prominent element of the WinWin Model is the emphasis it places on negotiations–base approach to defining software requirements, architecture, development products and management strategies. The WinWin Model extends the original model by including three activities at the beginning of each spiral:

- Identifying the system’s key stakeholders – these include the developer, user and customer;
- Identifying the stakeholders’ “win” conditions for the system – win conditions for the customer include schedule and cost; win conditions for the user include functionality; for the developer, win conditions include schedule, profit, and adequate resources; and
- Negotiating “win” conditions so that all key stakeholders are satisfied.

A major strength of the WinWin Spiral Model lies in explicitly recognizing that each stakeholder has a set of “win conditions” that need to be addressed.

2.2.5 A PREVIEW APPROACH TO MANAGING VIEWPOINTS

The Viewpoints approach is introduced through the CORE system in 1979 [23]. The central tenet of Viewpoints is that no one stakeholder has a complete understanding of what the target system should do. From the Viewpoints perspective then, we are encouraged to elicit multiple viewpoints, and to reconcile their perspectives into a single, comprehensive view of the system. Sommerville, Sawyer and Viller [24] propose a generic process model, PREview, which is spiral in nature and emphasizes the viewpoint approach. The model’s three basic activities are:

- Requirements Elicitation – guided by organization concerns, requirements elicitation focuses on understanding the problem and the application domain, and then evolving corresponding requirements.
- Requirements Analysis – requirements are collected and then analyzed for completeness, inconsistencies and potential conflicts.
- Requirements Negotiation – analysts and stakeholders examine inconsistent or conflicting requirements, and formulate resolutions that promote consensus and acceptable “win” conditions for all.

Each of the above activities stresses the Viewpoint perspective. That is, viewpoints are identified during elicitation; their interactions are examined during the analysis activity; and finally, during the negotiations viewpoints are accepted, modified or tagged for additional examination during the next cycle of the spiral.

2.2.6 THE REQUIREMENTS GENERATION MODEL (RGM)

Within the conventional Waterfall Model (or one of its many variants), five development phases are most prominently represented: requirements analysis, design, implementation, testing, and maintenance. Additionally, one or more of these phases is further decomposed to emphasize important sub-components. For example, it is common to find design split into a high- and a low-level design phase; similarly, implementation is often divided into code and unit-testing sub-phases.

Like design and implementation, the evolution of requirements is based on two fundamentally distinct sets of activities: those focused on requirements generation and those supporting requirements analysis. Historically, however, requirements generation is included as an activity within requirements analysis. This view, unfortunately, promotes the misconception that requirements generation is only a minor activity [25]. Clearly, such implications run counter to the established importance of requirements generation.

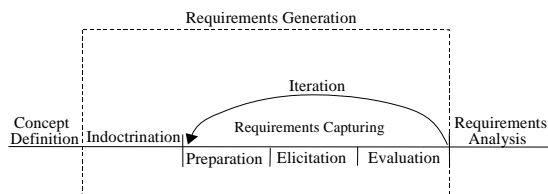


Figure 2: The Requirements Generation Framework

To address the misconception perpetuated by historical practices and to bring into focus the importance of the requirements generation “phase”, the Requirements Generation Model (RGM) refines the conventional requirements analysis phase. As illustrated in Figure 2 the refinement is a structured framework that distinguishes between concept definition, requirements generation and requirements analysis.

Concept definition provides an initial problem description and a preliminary set of needs. Requirements analysis focuses on examining the

individually generated set of requirements *as a collective whole*. Principal activities focus on (a) verification and validation, (b) confirming set completeness, and (c) producing a requirements specification document.

Requirements generation, the most prevalent process component, is composed of four distinctively different activities (or sub-phases): indoctrination, preparation, elicitation and evaluation. Indoctrination focuses on educating the stakeholders; preparation prescribes pre-meeting tasks that help increase the potential of conducting a successful meeting; elicitation activities center on the identification and recording of requirements during meeting with the stakeholders; and finally, evaluation is concentrated on organizing, verifying and validating the evolved requirements set. The latter three activities are placed within an iterative process whose exit criterion is the realization of a complete and sufficiently well defined set of requirements.

2.3 SUMMARY OF POINTS OF EMPHASIS

The review in the prior section is necessary to provide a sufficient understanding of the diverse yet related approaches to overcoming the “requirements problem.” The following summary of comparative and contrasting points is intended to assist the reader in noting the presence or absence of SRE techniques and methods in simulation modeling methodology.

Excepting requirements triage, requirements elicitation and analysis are prominent activities in all of the models outlined above. Requirements Triage, like the WinWin model, stresses risk analysis. The scope of the WinWin and Agile Modeling extend beyond requirements to include activities focused on architecture, design, and implementation; the RGM, on the other hand, focuses exclusively on the generation of necessary and sufficient requirements.

Distinctive differences are seen in the extent to which a model: (1) structures the requirements generation process and (2) advocates the use of specific methods and techniques to achieve stated objectives. The RGM provides an iterative framework and a monitoring methodology that define structure, activities and methods to support requirements generation. The Knowledge-Level Process Model provides a well-defined process structure for requirements elicitation, management and documentation, but suggests no supporting methods or techniques. Agile Modeling, on the other hand, defines 17 principles, 21 practices and four values, but offers

no real structure within which to employ those practices. PREview appears to fall somewhere between the Knowledge-Level Process Model and Agile Modeling, and like the RGM, emphasizes similar activities within an iterative (spiral) process.

3. THE PERCEIVED ROLE OF REQUIREMENTS IN MODELING AND SIMULATION

The expansion in understanding of both the importance and the complexity of the requirements phase in general software development is not reflected in M&S process descriptions. This assertion is supported by the paucity of treatment given the requirements specification function in recent M&S textbooks, the lack of identification of the activities identified for general software in M&S life-cycle models, and the relative lack of attention in publications in the M&S literature. With the exception of military applications, the advances in requirements engineering described in Section 2 are not evident in model development for simulation analysis.

The obvious counter to the negative implications of this assertion is that requirements engineering is a software engineering activity not a M&S activity. However, this partitioning that did not serve well in the past is even less workable today.

3.1 TREATMENT OF REQUIREMENTS IDENTIFICATION IN M&S BOOKS

Early textbooks for M&S courses typically describe the steps or phases in developing a model in rather general terms, reflecting the “six ‘major phases’ of an operations research study” attributed to Churchman *et. al.* [26] by Mize and Cox [27]:

1. Formulating the problem.
2. Constructing a mathematical model to represent the system under study.
3. Deriving a solution from the model.
4. Testing the model and the solution derived from it.
5. Establishing controls over the solution.
6. Putting the solution to work: implementation.

Early courses in M&S appear in a number of academic departments; thus the claim might be made that books targeted for students in mathematics or statistics might not view model development as an essential element in learning the subject. However, an informal survey of books spanning the last three decades reveals no direct treatment of topics such as “model

requirements,” “requirements specification,” and “simulation requirements.” This survey included books emphasizing the learning of a simulation programming language or environment, books including multiple languages or environments, and those stressing M&S concepts and the management of simulation projects.

Recent books in M&S or updates of earlier texts, with one exception, continue to give little or no attention to model requirements. That exception is the work of Robinson [28, pp. 66-67], who references the five general qualities of an effective model identified by Willemain [29]: validity, usability, value to client, feasibility, and aptness for clients’ problem. Robinson emphasizes the importance of the conceptual modeling phase and identifies the four main requirements of a conceptual model as *accuracy*, *credibility*, *utility* and *feasibility*. He proceeds to describe the product of the conceptual modeling phase as a *simulation project specification* that includes [28, p. 69]:

- Background to the problem situation.
- Objectives of the simulation study.
- Expected benefits.
- The conceptual model: inputs, outputs, content (scope and level of detail), assumption and simplifications.
- Experimentation: scenarios to be considered.
- Data requirements.
- Time-scale and milestones.
- Estimated study cost.

Robinson does not delve into the process for generating, analyzing, and verifying functional requirements.

While the criticism is pointed at M&S, early programming texts are equally unconcerned about the activities in progressing from problem recognition to solution program. As noted above, a change is obvious over the last 15 years.

3.2 TREATMENT OF REQUIREMENTS IN MILITARY AND GOVERNMENT M&S STUDIES

Among contemporary applications of M&S, the military and government agencies form a notable contrast in their attention to requirements. These studies could be classed as “contract driven,” and the concern for having test criteria that support an acceptance decision is apparent. The in-depth discussion of general requirements for simulation models used in waste management [30] is a notable example. The development of the Global Systems Simulation Program is another [31], albeit one that deals primarily with continuous simulation models.

3.3 TREATMENT OF REQUIREMENTS IN M&S LIFE-CYCLE MODELS

The attention to requirements in M&S life-cycle models is difficult to examine based on explicit references. One approach is to consider an M&S study as a software project and to utilize general project management software that imposes a pre-defined set of activities [32]. Such a generic description of each activity renders the identification of SRE actions and techniques almost impossible. Consequently, we restrict our attention to three life-cycle models specifically intended to describe M&S studies.

3.3.1 THE KREUTZER MODEL

Kreutzer [33, p. 7-9] describes “the life cycle of a simulation project” with the activities and information flow shown in Figure 3. While similar descriptions of the steps in a simulation study can be found in other simulation books, this source admits to emphasizing the modeling methodology issues that are referred to as the “nine aspects of simulation methodology” [33, p. 9].

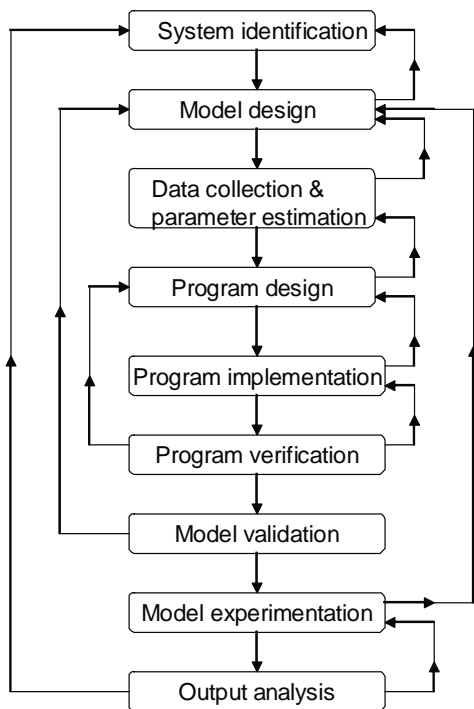


Figure 3. The Kreutzer Life-cycle Model of a Simulation Project

Kreutzer’s treatment of the conduct of a simulation study occurs at a junction point in modeling methodology: the transition from emphasis on *programming language* specification to *conceptual model* specification. Model

development environments are the subject of research investigation and discussions [34, 35]. In his own words, the focus of treatment is “modeling styles and programming techniques.”

Interesting perspectives are shared regarding conceptual understanding and the languages or notations used to represent such, but no explicit references to model requirements, requirements specification, or similar terms are evident. The recognition of the importance of SRE remains in the future.

3.3.2 THE BALCI-NANCE MODEL

The simulation project life cycle introduced by Balci and Nance [36] is touted as a comprehensive identification of the activities beginning with problem recognition and concluding with use of the results in decision support. Three groupings of phases are shown in Figure 4: (1) Problem Formulation, (2) Model Development, and (3) Integrated Decision Support. An emphasis is placed on the verification and validation (V&V) actions required as the transformation from each phases to its successor.

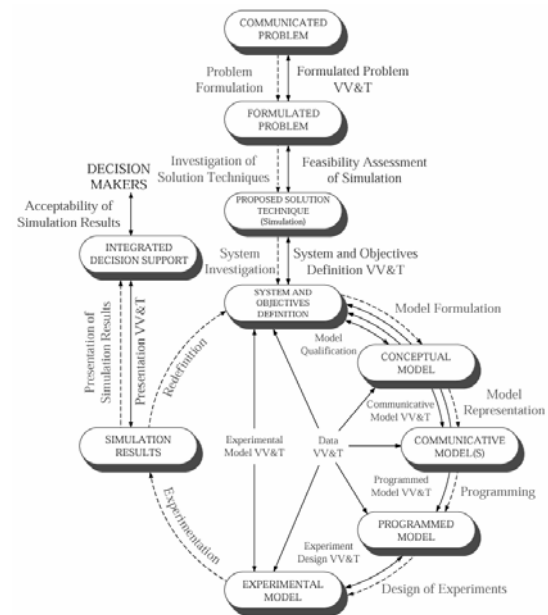


Figure 4: The Balci-Nance Life Cycle Model of a Simulation Project

The first four phases (Communicated Problem to System and Objectives Definition) are categorized as the Problem Definition phases. (Actually, the System and Objectives Definition serves as a transition to the Model Development phases, and a similar role is filled by the Simulation Results phase in transition to Integrated Decision Support.) Problem

Formulation, sometimes described in the literature as “Problem Structuring” or “Problem Definition,” is the process by which the initially communicated problem is translated into a formulated problem sufficiently well defined to enable specific actions toward a solution. The difficulties in this essential translation can be traced to:

- Insufficient attention is given to these initial phases in the life cycle.
- The problem is educational; few courses devote much attention to the issues because instructors either do not understand what is involved in this “art form” or feel inadequate to convey it to students.
- The accuracy of problem formulation greatly affects the acceptability and credibility of simulation results.

Prescribed actions within the Problem Definition phases might include some elements of SRE, but the existence of project requirements appears to be assumed. The mention of requirements occurs in the following question:

Can all of the specific requirements of the simulation project be satisfied (e.g., access to pertinent classified information)?

However, no explicit guidance or direction is given for developing, analyzing or using such requirements. The potential for employing SRE techniques within Model Formulation and Model Qualification appears to exist, but no explicit discussion is given as to how this might be done.

3.3.3 THE SARGENT MODEL

Figure 5 characterizes the on-going and evolving relationships between the simulation model and the system under study [37]. While not originally offered as such, the illustration provides an effective life-cycle model of an evolving simulation model of a continually changing complex system.

This model permits questions to be formulated as system experiment objectives that can be examined through the simulation model behavior or the converse: the simulation model behavior can be validated by examining the system behavior under the modeled conditions. An interesting interplay of validation is shown in three forms: theory, the conceptual model, and operational results. Experimentation is conducted on the system as well as the simulation model. Like the Balci-Nance Model, the description of activities does not advance to the details of specifying model requirements.

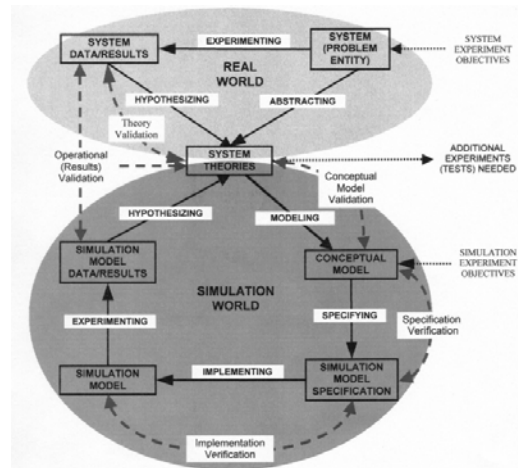


Figure 5. The Sargent Life-cycle Model of the Evolving System/Model Relationship

4. INCORPORATING EXPLICIT SRE ACTIVITIES TO IMPROVE M&S CREDIBILITY

None of the M&S life-cycle models address requirements at the level of detail permitting SRE actions, for example following the RGM approach, to be introduced. No doubt, the avoidance of this level of details reflects the past division between *modeling* and *programming*, where the former was the domain of the simulation analyst and the latter, that of the programmer. This division has experienced a blurring over the same period as SRE has received greater emphasis. Current thinking in the software domain advances terms such as “model-driven design” and “model-driven development” (MDD) [5]. Some software sages cite MDD as the next step in the evolutionary influences of object-oriented design and rapid application development (RAD). Clearly, efforts such as the systems modeling language (SysML) are eradicating the prior distinctions between modeling and programming.

Extending the SRE process changes into M&S development is likely to affect the problem definition phases in the Balci-Nance life-cycle model and the conceptual and communicative modeling activities in both the Balci-Nance and Sargent models. The explicit production of model requirements, a common artifact in military M&S applications, is likely to be adopted as an industry standard. A consequence of this outcome is to improve model testing and the verification and validation activities applied in the communicative and conceptual modeling phases. The precise definition of methods revisions remains dependent on other aspects of the model development methodology employed.

5. Summary and Conclusions

In this paper we have presented some observations and conjectures regarding the role of software requirements engineering in M&S model development. Within the software engineering arena both the theories and practices of requirements engineering are current areas of exploration. The RGM, PREview and Win-Win Spiral Model all reflect the importance of requirements engineering in software development, as well as the derived benefits. From the perspective of Modeling and Simulation, the emphasis on requirement specification during model development and analysis is somewhat less. With the notable exception of Robinson's book, current modeling and simulation books tend to ignore, or only slightly mention, the role that requirements can (or should) play in model development. Similarly, the three M&S Life-cycle models discussed in Section 3.3 lack explicit references to activities supporting the evolution of model requirements. Their decomposition and enunciation of distinct M&S activities, however, do provide "gaps" for the introduction of complementary requirements engineering activities.

Robinson's book and "requirements friendly" M&S models are positive indicators that requirements specification has potential utility in the M&S development life-cycle. An additional indicator is DoD's encouragement (or insistence) that requirements specification be an integral component of its M&S contracts for large-scale systems. It is the authors' shared opinion that these are moves in the right direction. We conjecture that, although development models differ between the Software Engineering and M&S disciplines, the derived benefits of a formal requirements specification process in either domain are similar: a higher quality product.

REFERENCES

- [1] Tocher, K.D. and D.G. Owen, "The Automatic programming of Simulations," In *Proceedings of the Second International Conference on Operational Research*, 1960, pp. 50-68.
- [2] Nance, R.E., "Simulation Programming Languages: An Abridged History," *Proceedings of the 1995 Winter Simulation Conference*, Arlington, VA, December 1995, pp. 1307-1333.
- [3] Weinberg, G.M., "Overstructured Management of Software Engineering," In *Proceedings of the 6th International Conference on Software Engineering*, Tokyo Japan, September 1982, pp. 2-8.
- [4] Nance, R.E., "Modeling and Programming: The Evolutionary Convergence," Presentation at Syracuse University, 1989.
- [5] Overington, M. Model-Driven Development Today, http://www.builderau.com.au/architect/soa/Model_Driven_Development_today/0,30024564,39205342,00.htm, April 26, 2005.
- [6] Royce, W.W., "Managing the Development of Large Software Systems: Concepts and Techniques," *Proceedings of the Western Electronic Show and Convention (WesCon)*, Los Angeles, August 1970, pp. 1-9 (Reprinted in the *Proceedings of 9th International Conference on Software Engineering*, March 1987, Monterey CA, pp. 328 – 338.)
- [7] Lobo, L.O., Analysis and Evaluation of Methods for Activities in the Expanded Requirements Generation Model (x-RGM), Masters Thesis, Department of Computer Science, Virginia Tech, URN: etd-07272004-133607, 2004.
- [8] V.R. Basili, A.Turner, "Iterative Enhancement - a Practical Technique for Software Development," *IEEE Transactions on Software Engineering*, SE-1(4), Dec 1975
- [9] Gomma, H. and Scott, D., "Prototyping as a tool in the specification of user requirements", In *Proceedings of the Fifth International Conference on Software Engineering*, San Diego CA, March 1981, pp. 333-342.
- [10] Barry Boehm, "A Spiral Model of Software Development and Enhancement," *IEEE Computer*, pp 61-72, May 1988.
- [11] Beck, K., *Extreme Programming Explained: Embrace Change*, Addison-Wesley, 1999.
- [12] Carter, R.A., Anton, A.I, Dagnino, A. and L. Williams, "Evolving Beyond Requirements Creep: A Risk-Based Evolutionary Prototyping Model," IEEE 5th International Symposium on Requirements Engineering (RE'01), Toronto Canada, August 2001, pp. 94-101.
- [13] Markus K. Gröner, *Capturing Requirements Meeting Customer Intent: A Structured Methodological Approach*, PhD

- dissertation , Virginia Tech, http://scholar.lib.vt.edu/theses/available/etd-5232002234024/unrestricted/Markus_K_Groener_Dissertation.pdf, 2002.
- [14] E. Bravo, "The hazards of leaving out the users," in *Participatory Design: Principles and Practices*, D. Schuler and A. Namioka, Eds. Hillsdale, New Jersey: Lawrence Erlbaum Associates, Inc. Publishers, 1993, pp. 3-11.
- [15] IPL Information Processing Ltd., *Designing Unit Test Cases*, 1996.
- [16] Rumbaugh, James, Ivar Jacobson, and Grady Booch, *The Unified Modeling Language Reference Manual*. Reading, MA: Addison-Wesley, 1999.
- [17] Alan Davis et al, "Identifying and measuring quality in a software requirements specification," *Proceedings of the 1st International Software Metrics Symposium*, pp 141-152, 1993.
- [18] Arthur, J.D. and Groener, M.K., "An Operational Model For Structuring the Requirements Generation Process," *The Requirements Engineering Journal*, January 2005, Vol. 10, No. 1, pp. 45-62.
- [19] Davis, A.M., "The Art of Requirements Triage," *IEEE Computer*, Vol. 36, No. 3 March 2003, pp. 42-49.
- [20] Herlea, D.E., C.M. Jonker, J. Treur, N.J.E. Wijngaards (2002), "A Compositional Knowledge Level Process Model of Requirements Engineering," *The International Journal of Software Engineering and Knowledge Engineering*, Vol 12, No. 1, February 2002, pp. 41-75.
- [21] Ambler, S.W. (2002). *Agile Modeling: Effective Practices for XP and RUP*. New York: John Wiley & Sons
- [22] Barry Boehm, "Using the Win-Win Spiral Model: A Case Study," *IEEE Computer*, Vol 31, No. 7, July 1998, pp 33-44.
- [23] Mullery, G. (1979) "CORE – A Method for Controlled Requirements Specifications," *Proceedings of the 14th International Conference on Software Engineering*, Munich Germany, September 1979, pp. 126-135.
- [24] Sommerville, I., P. Sawyer and S. Viller, "Viewpoints for requirements elicitation: A Practical Approach," *The Third IEEE International Conference on Requirements Engineering, Colorado Springs CO*, April 1998, pp. 74-81.
- [25] Alan Davis, *Software Requirements: Objects, Functions, & States*, Prentice-Hall, Upper Saddle River, New Jersey, 1993.
- [26] Churchman, C. W., R.L. Ackoff and E.L. Arnoff, *Introduction to Operations Research*, John Wiley & Sons, Inc. New York, 1957.
- [27] Mize, J.H. & Cox, J.G., *Essentials of Simulation*, Prentice-Hall, 1968.
- [28] Robinson, S., *Simulation : The Practice of Model Development and Use*, John Wiley, Chichester, West Sussex, England.
- [29] Willemain, T.R., "Insights on modeling from a dozen experts," *Operations Research*, Vol. 42, No. 2, pp. 213-222.
- [30] Miller, I., Kossik, R. and C. Voss, "General Requirements for Simulation Models in Waste Management," Waste Management 2003 Symposium, February 2003, Tucson AZ., <http://www.wmsym.org/abstracts/2003/pdfs/612.pdf>.
- [31] Drake, J., Foster, I., Malone, B., Williams, D., and D. Bader, "Global Systems Simulation Software Requirements: An Outline for CSET-ACPI Leveraging," http://www.csm.ornl.gov/ACPI/Documents/ACPI_CSET.htm
- [32] Chung, C. *Simulation Modeling Handbook: A Practical Approach*, CRC Press, Boca Raton, Florida, 2003.
- [33] Kreutzer, W., *System Simulation – Programming Styles and Languages*, New York: Addison Wesley, 1986.
- [34] Nance, R.E., Model Development Revisited, *Proceedings of the 1984 Winter Simulation Conference*, Dallas, Texas, 28 – 30 November 1984, pp. 75-80.
- [35] Nance, R.E., A Tutorial View of Simulation Model Development, *Proceedings of the 1983 Winter Simulation Conference*, Washington, D.C., 12-14 December 1983, pp. 325-331
- [36] Balci, O. and R.E. Nance, "Simulation Model Development Environments: A Research Prototype", *Journal of the Operations research Society*, Vol. 38, No.8, pp. 753-763.

- [37] Sargent, R.G., "Some Approaches and Paradigms for Verifying and Validating Simulation Models," Proceedings of the 2001 Winter Simulation Conference, Arlington VA, December 2001, pp. 106-113.

AUTHOR BIOGRAPHIES

RICHARD E. NANCE is Chief Scientist and Chairman of the Board for Orca Computer, Inc. He is also Professor Emeritus in Computer Science at Virginia Tech. He was the RADM John Adolphus Dahlgren Professor of Computer Science and the Director of the Systems Research Center at Virginia Tech. He held a distinguished visiting honors professorship at the University of Central Florida for the spring semester, 1997. Dr. Nance has held research appointments at the Naval Surface Weapons Center and at the Imperial College of Science and Technology (UK). He has held a number of editorial positions and was the founding Editor-in-Chief of the *ACM Transactions on Modeling and Computer Simulation*, 1990-1995. Currently, he is a member of the Editorial Board, Software Practitioner Series, Springer. He served as Program Chair for the 1990 Winter Simulation Conference. Dr. Nance received a Distinguished Service Award from the TIMS College on Simulation in 1987. In 1995 he was honored by an award for "Distinguished Service to SIGSIM and the Simulation Community" by the ACM Special Interest Group on Simulation. He was named an ACM Fellow in 1996.

JAMES D. ARTHUR is an Associate Professor of Computer Science at Virginia Tech. His research interests are in Software Engineering domain and include Requirements Engineering, Independent Verification and Validation, Software Quality Prediction and Assessment. Dr. Arthur is the author of over 35 papers in those areas. He has served as: participating member of IEEE Working Group on Reference Models for V&V Methods; Chair of Education Panel for National Software Council Workshop; and Co-Guest Editor for *Annals of Software Engineering* special volume on Process and Product Quality Measurement.