# Ideas About Simulation Conceptual Model Development

*Dale K. Pace*

The conceptual model for a simulation addresses the simulation's context, how it will satisfy its requirements, and how its entities and processes will be represented. The conceptual model is key to (1) assessing a simulation's validity for any situation not explicitly tested and (2) determining the appropriateness of a simulation (or its parts) for reuse or use with other simulations in a distributed simulation. There are no widely accepted approaches for decomposing the representation of the simulation subject into the entities and processes of a simulation's conceptual model, for abstracting such representation from available information about the subject, or for describing and documenting the simulation's conceptual model. This article discusses the development of a conceptual model for both unitary and distributed simulations, focusing on approaches that enhance model completeness, consistency, coherence, and correctness. More than a decade of work at APL in this area is represented in the ideas presented here. (Keywords: Conceptual model, Fidelity, Simulation, Validation.)

## INTRODUCTION

Contemporary software developments, including those for simulations, are limited more by methods for conceptual model development than by implementation capabilities. According to Cook,[1]

> We are not really having a problem coding a solution—we are having a problem understanding what solution to code. . . . If you focus on requirements and verification and validation, the coding will take care of itself.

To compound the problem, there are no generally accepted guidelines to help a simulation developer determine which attributes of a subject should be represented in a simulation or what level of fidelity is required for their representation. As explained later, guidelines and formalisms have been developed, but they have either limited applicability or lack wide acceptance and use. A format for describing conceptual models for simulations has been recommended[2] and is expected to become part of the DoD *Recommended Practices Guide for Verification, Validation, and Accreditation (VV&A)*.[3] Despite efforts to create an integrated collection of standards,[4] divisiveness over software standards has caused many critical factors (e.g., data item descriptions) to be omitted in some contracts.[5] This, in turn, reduces the likelihood of quality conceptual model development.

Proper development of a conceptual model is critical. A simulation's conceptual model describes how a developer intends an implementation to satisfy its

requirements. This model is the primary mechanism for transforming simulation requirements into specifications that can guide simulation development and implementation. Therefore, the conceptual model is the only rational basis for judging a simulation's capabilities under circumstances other than those specifically tested. It is also the basis for judgments about the appropriateness of the simulation (or its parts) for reuse in other simulations, an important consideration in an era of simulation-based design and acquisition. Conceptual models of simulations also provide the primary basis for judgment about the consistency and compatibility of simulations in a distributed simulation application such as a High Level Architecture (HLA) federation.

Guidance about conceptual model development, evaluation, and characterization has still not been generalized coherently. For example, in the early 1990s, Mayhew[6] provided guidelines for conceptual models of software user interfaces. The DoD Joint Technical Architecture[7] prescribed Integrated Computer Aided Manufacturing Definition (IDEF) models for data and process descriptions of systems that exchange information, but that descriptive methodology is not yet used for all defense software projects. However, it is being applied specifically to object-oriented systems in a new set of IEEE standards.[8]

Soft systems methodology, knowledge-based system development, formal methods, and knowledge engineering have approaches for abstraction and expression pertinent to simulation conceptual modeling. These methods have been used successfully in many applications. For example, an intelligent brokering service for knowledge-component reuse on the Web[9] is envisioned as allowing one to enter the specification of the knowledge-intensive problem (e.g., an engineering design problem) and then enabling the broker (using shared ontologies to support such component reuse) to examine available libraries of software components so as to configure a suitable problem solver. This example shows the kind of benefit possible when development discipline (such as shared ontologies) is employed extensively. Unfortunately, because these methods have not yet matured to general applicability, each researcher develops or modifies existing methods so that he or she can address a particular problem. This is reflected in the many articles in the *Journal of Conceptual Modeling*[10] about the capabilities and limitations of descriptive and development formalisms, especially those in the object-oriented arena, for conceptual modeling.

Likewise, there is little consensus about how to evaluate conceptual models. Teeuw and van den Berg,[11] as well as Lindland et al.,[12] identify quality criteria for conceptual models: completeness, propriety (pertinence), clarity, consistency, orthogonality (modularity, the independence of aspects of the subject represented), and generality (implementation independence). I focus on four quality attributes for a simulation conceptual model:

1. Completeness: The simulation conceptual model identifies all representational entities and processes of the problem domain (sometimes called the "mission space" in DoD parlance) and all control and operating characteristics of the simulation, which I will call "simulation space," needed to ensure that specifications for the simulation fully satisfy simulation requirements.
2. Consistency: Representational entities and processes within the conceptual model are addressed from compatible perspectives in regard to such features as coordinate systems and units; levels of aggregation, deaggregation, precision, accuracy, etc.; and descriptive paradigms.
3. Coherence: The conceptual model is organized so that all elements of both mission space and simulation space have function (i.e., there are no extraneous items) and potential (i.e., there are no parts of the conceptual model which are impossible to activate).
4. Correctness: The simulation conceptual model is appropriate for the intended application and has potential to perform in such a way as to fully satisfy simulation requirements.

This article begins with a definition for the simulation conceptual model and then addresses the fundamental issue of how to develop such a model, focusing on approaches and formalisms that can enhance simulation conceptual model clarity, completeness, consistency, and correctness. These methods include

- Conceptual model definition processes, which are closely associated with the subject of requirements engineering
- Conceptual model decomposition, which is concerned with the level of detail or aggregation that is appropriate for simulation elements
- "Real world" abstraction to provide representation in the simulation

Conceptual model development is addressed for both unitary and distributed simulations, and includes comments about implementation and control aspects of the model (i.e., the simulation space) as well as its main focus on representational aspects (i.e., the mission space). Simulation conceptual model development is still an art; guiding principles are incomplete and evolving for both unitary and distributed simulation.

## THE SIMULATION CONCEPTUAL MODEL

The current version of the DoD *Glossary of Modeling and Simulation Terms*[13] does not define "simulation conceptual model." It follows the approach of the

Distributed Interactive Simulation (DIS) community by defining "conceptual model" as the agreement between the simulation developer and user about what the simulation will do. In this article the connotation for simulation conceptual model is that found in the current (revised) version of the DoD *Recommended Practices Guide for Verification, Validation, and Accreditation (VV&A)*[3]: a simulation developer's way of translating modeling requirements (what is to be represented by the simulation) into a detailed design framework (how it is to be done), from which the software, hardware, networks (in the case of distributed simulation), and systems/equipment that will make up the simulation can be built. A conceptual model is the collection of information that describes a simulation developer's concept about the simulation and its pieces. That information consists of assumptions, algorithms, characteristics, relationships, and data, which describe how the simulation developer understands what is to be represented by the simulation (entities, actions, tasks, processes, interactions, etc.) and how that representation will satisfy simulation requirements. The more perspicuous and precise the conceptual model, the more likely the simulation development both fully satisfies requirements and demonstrates that the requirements are satisfied (i.e., validation).

A simulation conceptual model should be a primary mechanism for clear and comprehensive communication among simulation developer design and implementation personnel (systems analysts, system engineers, software designers, code developers, testers, etc.), simulation users, subject-matter experts (SMEs) involved in simulation reviews, and evaluation personnel, such as those involved in VV&A. A simulation's conceptual model addresses the simulation's context, elements, and concept (Fig. 1).

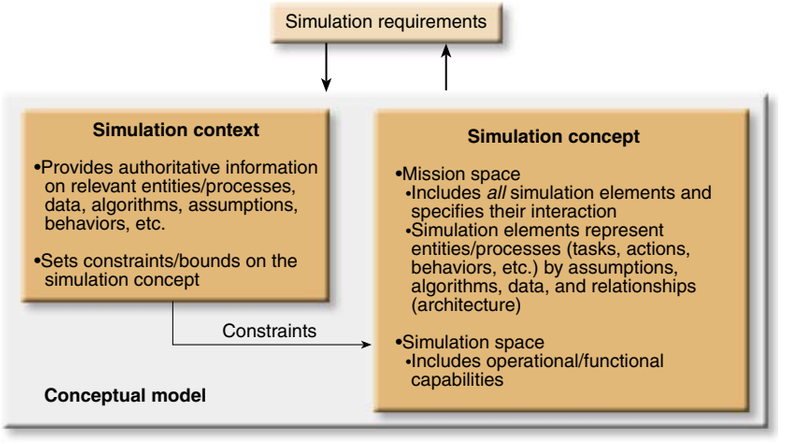



**Figure 1.** The simulation conceptual model.

The simulation context provides "authoritative" information about the domain which the simulation is to address. In simulations that provide realistic representations of physical processes, the laws of physics and principles of engineering are part of the simulation context. For many military-related simulations, the simulation context includes standard organizational structures and general doctrine, strategy, and tactics. Often it is merely a collection of pointers and references to sources that define behaviors and processes for things that will be represented within the simulation. Special care, particularly for distributed simulations, must be used when algorithms are taken from more than one source to ensure that sources do not employ contradictory assumptions or factors (such as different models for the shape of the Earth, differences in characterizing the environment, etc.). The information contained in the simulation context establishes boundaries on how the simulation developer can properly build the simulation.

A simulation element consists of information describing concepts for an entity, a composite or collection of entities, or a process that is represented within a simulation. It includes assumptions, algorithms, characteristics, relationships (especially interactions with other things within the simulation), data, etc., that identify and describe an item's possible states, tasks, events, behavior, performance, parameters, attributes, etc. A simulation element can address a complete system (e.g., a missile or radar), a subsystem (e.g., the antenna of a radar), an element within a subsystem (e.g., a circuit within a radar transmitter), or even a fundamental item (e.g., an atom). It can also address composites of systems (e.g., a ship with its collection of sensors, weapons, etc.). It should be noted that a person, part of a person (e.g., a hand), or a group of people can likewise be addressed by a simulation element. It can also address a process such as environmental effects on sensor performance.

The simulation concept describes the simulation developer's concept for the entire simulation application (all the federates and other pieces in a distributed simulation, i.e., everything that comprises the simulation) and explains how the simulation developer expects to build a simulation that can fully satisfy user-defined requirements. The simulation context establishes constraints and boundary conditions for the simulation concept. If the simulation is concerned with a realistic representation of missile flight, then laws of physics and principles of

aerodynamics are part of the simulation context, making the simulation concept accommodate conservation of momentum, etc. An unrealistic, cartoon-like representation of missile flight would not necessarily be so constrained. The simulation concept includes simulation elements, i.e., the things represented in the simulation. The simulation concept is the total of all simulation elements and specifies how those elements interact. This is essentially the mission space part of the simulation concept.

The simulation space part of the simulation concept includes all additional information needed to explain how the simulation will satisfy its objectives. Such information often addresses control capabilities intended for the simulation (e.g., pause and restart capabilities; data collection and display capabilities; and how data and simulation control factors can be entered into the simulation by keyboard, voice, gesture, touch, or feedback from other parts of the simulation). Simulation space characteristics can range from the identification of specific kinds of computing systems (hardware and operating systems that the simulation must run on), to timing constraints so that real systems can be part of the simulation (e.g., hardware-in-the-loop unitary simulations or involvement of live forces in distributed simulations), to simulation control capabilities described previously. Some simulation space considerations are closely related to implementation issues for the simulation; for example, the selection of a parallel computing architecture has implications for algorithms used to describe simulation elements.

A primary function of the simulation conceptual model is to serve as the mechanism by which simulation requirements are transformed into detailed simulation specifications (and associated simulation design) that fully satisfy the requirements. This transformation is easiest and most reliable if both the requirements and the specifications can be expressed in the same descriptive formalism, because every translation (even if mundane) from one descriptive formalism to another introduces an additional source of potential error. Such errors may result from something as simple as failure to translate one set of units to another, which caused the failure of NASA's Mars Climate Orbiter in September 1999.[14] Misinterpretation of symbols and factors is also possible. This can result from different connotations for items with multiple definitions, as with the statistical term "mean," where the arithmetic mean is quite different from the geometric mean. Similar problems result when there is no convenient construct or concept in the subsequent descriptive format to address all aspects of the information contained in the format from which the information is being translated (this is a well-known problem in natural language translation).

Extensive literature on transformation from one descriptive format to another exists, especially concerning formal software development environments. The literature ranges from earlier works like the 1990 book by Partsch, *Specification and Transformation of Programs*,[15] to more recent endeavors such as Poston's *Automating Specification-Based Software Testing*[16] and Grady's *System Validation and Verification*.[17] Such works provide general insights about issues related to descriptive format transformations. Although no satisfactory general and complete solution to the problem of transformation has yet been devised, these references and related literature provide one starting point for a more general, comprehensive theory of simulation conceptual model development and for a framework within which to discuss how this development is affected by descriptive format choices.

A description of the simulation context, simulation elements, and simulation concept is nearly always ambiguous because some parameters are generic and others are application (run) specific. A simulation element for a sensor may contain parameter values and algorithms that fully characterize sensor states, behavior, performance, etc. For a particular application, it may be desirable to change some parameters from run to run. One could have a parameter set by simulation inputs at the beginning of each simulation run. Or one could have a slightly different simulation element for the sensor, with the appropriate parameter value, for each run.

## CONCEPTUAL MODEL DEVELOPMENT

Once simulation objectives have been established, development of the simulation conceptual model may begin. Simulation requirements and conceptual model development are a classic "chicken–egg" pair. They each stimulate and derive from the other. Conceptual model development may even begin before completion of the simulation requirements. In some cases, the iterative and interactive formulation of simulation requirements with development of the simulation conceptual model can be beneficial.

Conceptual model development may reveal problems with simulation requirements, especially if there has not been a rigorous validation of simulation requirements prior to the initiation of conceptual model development or if the best practices of contemporary requirements engineering have not been employed comprehensively. As the simulation conceptual model is developed to fully satisfy simulation requirements, inconsistencies and a lack of balance among the requirements (some very lax and others very stringent in the same general area) may become apparent. Likewise, development of the conceptual model may reveal serious "holes" (areas where the simulation developer is left to his own initiative about what the simulation

should be able to do) in the requirements. A good simulation development program will insist upon the use of the best contemporary requirements engineering practices, encourage early formal and rigorous validation of simulation requirements, and ensure that requirement deficiencies uncovered during conceptual model development are corrected with appropriate modifications to the simulation requirements.

Improvement in simulation requirements has major benefits. About half of all software errors are due to requirement deficiencies and errors. The cost to correct an error caught early may be only a few percent of the cost to correct that same error later in the development cycle.[18] In addition, formal conceptual model development is likely to produce a description of the simulation that will yield a more reliable estimation of resources required to develop the simulation, e.g., an estimation based on Function Point Analysis instead of the number of source lines of code.[18]

There are four basic steps in the development of a simulation conceptual model (Fig. 2). The first step is to collect authoritative information about the intended application domain that will comprise the simulation context. Development of the simulation concept and collection of authoritative information for the simulation context are likely to occur iteratively as the entities and processes to be represented in the simulation are more clearly defined. Authoritative descriptions of military activities such as those contained in Conceptual Models of the Mission Space (CMMS) in the Defense Modeling and Simulation Office (DMSO) repository[19] can be used in the simulation context when appropriate for a simulation's intended application, as can the laws of physics and similar principles.

It is unlikely that the formal, documented simulation context, should one exist, will cover everything needed to fully describe the domain that a simulation is to address. CMMS endeavors to emphasize a disciplined procedure by which the simulation developer is systematically informed about the real world and the set of information standards used by simulation SMEs to communicate with and obtain feedback from military operations SMEs. Common semantics and syntax, a common format database management system, and common data interchange formats are keys to removing potential ambiguity between the ideas of the warfighting SMEs and the simulation development SMEs. Significant progress has been made in developing a CMMS tool set to provide the keys noted above, but information beyond that likely to be obtained in the first-level abstraction (i.e., the CMMS itself) may be required for a simulation conceptual model. SMEs may be "called upon to fill in details needed by simulation developers" that are "not provided in doctrinal and/or authoritative sources."[20]

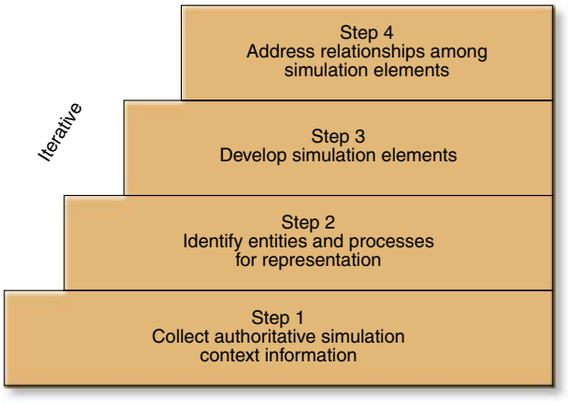



**Figure 2.** Steps in conceptual model development.

Unfortunately, caution is also required. It has long been known that inserting the knowledge engineer (or other agent of information translation) in the role of an intermediary between the expert and the knowledge-based systems (or simulation) developer may create as many problems as it solves.[21] Many approaches have been developed to address this problem. For example, Sharp[22] developed a process to ensure that the expert and the knowledge engineer or program developer have the same understanding of the information being acquired and transferred to a knowledge-based system or to some other form of expression.

The more complete and clearly stated a simulation context is, the easier it will be to understand how one simulation entity (or simulation in a federation) may differ from another in its assumptions about the domain which is addressed. This becomes important when questions of compatibility among simulations (federates) considered for a distributed simulation implementation (federation) are addressed as well as in assessing the coherence and compatibility of simulation parts.

The second step in conceptual model development is to identify entities and processes that must be represented for the simulation to accomplish its objectives. This enumeration process is fundamental. It is here where basic decisions are made about the level of detail and aggregation that is appropriate to support simulation requirements. These decisions determine whether a system (such as a radar) will be represented as a single entity, as a composite of subsystem entities (such as an antenna or receiver), or as a composite of composites of ever-smaller entities (to whatever level of detail is needed for the purpose of the simulation). Decisions are made at this step about the level of representation of human decisions and actions. Take, for example, the movement of a platform (tank, aircraft, ship, etc.). Are decisions and responses of all the people involved (the crew) represented implicitly as a single aspect of the movement control process, or is each person represented explicitly

(as in a tank simulator with a position for every member of the tank crew)?

The third step is the development of simulation elements. A simulation element is needed for each entity or process (or composites of such) identified in step two. Here, decisions are made initially about the level of accuracy, precision, resolution, etc., needed in the representation of the entity or process. Simulation elements determine the functional and behavioral capabilities of the simulation. Simulation fidelity is a function of both the scope of representation in a simulation (the entities and processes identified in step two) and the quality of entity and process representation in terms of accuracy, precision, etc.

The fourth step addresses relationships among simulation elements to ensure that constraints and boundary conditions imposed by the simulation context are accommodated. In addition, it ensures that simulation space issues (e.g., control capabilities that allow the simulation to be stopped, backed up in time, restarted, etc., or that identify data to be collected during the simulation) are addressed appropriately. These four steps will be iterated often in most simulation developments.

Conceptual model development should always be done within the context of simulation theory, such as Application Domain Modeling[23] or Discrete Event System Specification (DEVS)[24] methodology, to ensure that conceptual model development has coherence and can be related directly to simulation development. Sarjoughian and Zeigler[25] developed a collaborative method for employing DEVS constructs in High Level Architecture (HLA) federation development. YEROOS,[26] an acronym derived whimsically from "yet another project on evaluation and research on object-oriented strategies," provides another approach. This is a collaborative venture of researchers from Belgium, Switzerland, Argentina, and Senegal housed at the QANT Research Unit of the University of Louvain, Louvain-La-Neuve, Belgium, to investigate and publish theoretical foundations for conceptual modeling.

## SIMULATION REQUIREMENTS AND CONCEPTUAL MODEL DEFINITION

Simulation consumers, i.e., those who use simulation results, usually develop simulation requirements. They seldom are expert in formal methods that allow requirements to be stated with mathematical precision. Simulation specifications and the requirements from which they are derived through the simulation conceptual model are typically stated in natural language (even when a computer-aided software engineering or CASE tool is used to facilitate tracing requirements to implementation). This informal approach to requirements limits the likelihood that the simulation will prove to be correct because of the "ambiguity, context sensitivity, and vagueness inherent in natural language in specification."[27] And worse, the best practices of contemporary requirements engineering often are not applied in developing requirements for the simulation, causing benefits that could result from the application of these best practices to be missed.

It may be impossible to fully express the requirements for a simulation, the conceptual model embodying those requirements, and the specifications derived therefrom in precise, mathematically formal terms because the simulation subject is too complex, those expressing this information have limited fluency in the mathematical languages, or their mastery of the formal descriptive format is incomplete. Even when mathematically precise formal methods cannot be used fully, their partial application can provide significant benefit in allowing parts of the simulation to possess provable correctness. This perspective should be taken for each simulation element and its associated requirements and specifications as well as for the overall simulation conceptual model and its requirements and specifications.

A long-term goal of the simulation community should be increased familiarity and competence with set theory, propositional logic, predicate logic, etc. These are the foundation for formal development environments such as those employed with languages like the Vienna Development Method (VDM) from IBM Laboratories in Vienna or Z developed at Oxford University. Increased familiarity with these topics would encourage the use of more formal and precise methods for simulation requirements and specifications, as well as for the conceptual model which connects them and the simulation elements within the conceptual model. The more that such formal approaches can be used, the greater the utility of various "automatic" verification tools, so that the scarce resource of knowledgeable, competent personnel can be focused more on validation issues to ensure that the simulation not only works properly but that it is really working as intended. This has been a major motivation in NASA's emphasis on the value and utility of formal methods.[28]

The movement toward greater use of more formal constructs in requirements and specifications is not unique to simulation development, but is a general trend as software engineering matures.[29] This progression is essential to allow more use of automation in the verification and validation of software and simulation, especially during design and early stages of development.[30]

## CONCEPTUAL MODEL DECOMPOSITION

Booch et al.[31] identify four basic principles of modeling:

1. The choice of models has a profound influence on how a problem is attacked and how a solution is shaped.
2. Every model may be expressed at different levels of precision.
3. The best models are connected to reality.
4. No single model is sufficient.

These principles suggest that modeling is essentially an art that has not yet matured into a scientific method. This is especially true for simulation conceptual model development. However, it does not prevent the application of rational processes to conceptual model development. We address this rational process in terms of conceptual model decomposition and conceptual model abstraction.

Conceptual model decomposition determines the simulation elements contained in the conceptual model. This determines the scope of representation in the simulation and the discernible levels of the simulation. The following rationale can guide the selection of simulation elements for inclusion. These six items can serve as a checklist (Fig. 3) in conceptual model decomposition while the model is being developed. Using this rationale will help to ensure that a conceptual model is complete and coherent.

1. There should be a specific simulation element for every item (parameter, attribute, entity, process, etc.) specified for representation by the simulation requirements. This rationale reemphasizes the importance of appropriate requirements for the simulation. Since a primary function of the conceptual model is to facilitate the transformation of requirements into specifications, there must be a total tracking of items in the requirements to the conceptual model. This item sets a minimum level of detail in simulation decomposition.

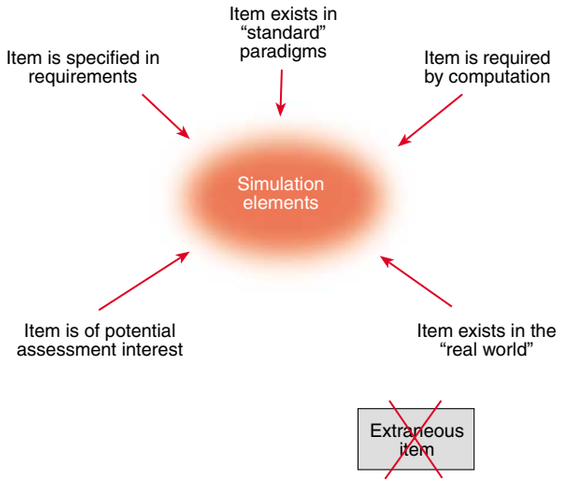



**Figure 3.** Checklist: rationale for simulation conceptual model decomposition.

2. There should be a specific simulation element for every item (parameter, attribute, entity, task, state, etc.) of potential assessment interest related to the purpose of the simulation. This stresses the importance of understanding the potential use of the simulation so that all measures of performance, measures of effectiveness, and measures of merit that might be associated with the use of the simulation are fully understood in order that the conceptual model may fully accommodate them. This rationale normally has implications for the simulation space aspects of the simulation (especially in regard to data collection and display capabilities) as well as for the representational aspects of the mission space.
3. As far as possible, there should be "real world" counterparts (objects, parameters for which data exist or could exist, etc.) for every simulation element. The potential impact of data and metadata structures on simulation elements and on the overall simulation conceptual model should not be underestimated. Credibility for simulations with realistic representations is always enhanced when there is easy correspondence between its elements and the real world so that direct comparisons can be made more readily.
4. Wherever possible, the simulation elements should correspond to standard and widely accepted decomposition paradigms to facilitate acceptance of the conceptual model and effective interaction (including reuse of algorithms and other simulation components) with other simulation endeavors. In most disciplines, standard paradigms exist for how an entity or process is described, measured, and evaluated. The clarity and credibility of a simulation conceptual model are enhanced when such standard paradigms are employed.
5. Simulation elements required for computational considerations (e.g., an approximation used as a surrogate for a more desirable parameter which is not computationally viable) that fail to meet any of the previously stated items should be used only when absolutely essential. Many computationally intensive problems require the use of approximations or less than the most rigorous mathematical expressions. Such should be used only when necessary, and a clear statement of the reason for use should be included as part of the conceptual model.
6. There should be no extraneous simulation elements. Elements not directly related to specific items in the simulation requirements, not implied directly by potential assessment issues, and without a specific counterpart in the real world or in standard decomposition paradigms should not be included in the simulation conceptual model. Every extraneous element is a source of potential simulation problems. Eliminating extraneous items removes unnecessary potential sources of errors and follows the principle of parsimony encapsulated in Ockham's Razor.

General guidance about the construction of an HLA federation is available[32]; however, the rationale for how to decompose an HLA federation (or other distributed simulation) into federates (component simulations) is embryonic. Pollack and Baker[33] developed HLA-specific software metrics for use in determining the appropriate level of decomposition in a specific HLA application. The metrics provided a quantitative measure for achieving a balanced federation in the effort to optimize the sometimes competing goals of utility and efficiency. Applying these metrics to a specific legacy simulation and a baseline federation object model identified the need for further decomposition to effectively support the intended application. As others perform similar assessments, it is likely that a set of metrics having general utility will emerge and become accepted by the community. Until these metrics are developed, those designing a distributed simulation will have to consider the availability of compatible simulation components as well as computational and bandwidth factors in determining how to best decompose the distributed simulation (HLA federation) for the intended application.

## REPRESENTATIONAL ABSTRACTION

The identification of simulation elements in conceptual model decomposition determines the scope of subject representation and the discernible levels possible in the mission space. How the characteristics of the simulation elements are abstracted determines the accuracy and precision of the representation. Because no single model is sufficient, the Unified Modeling Language (UML) that Booch et al.[31] developed has nine kinds of diagrams (class, object, use case, sequence, collaboration, statechart, activity, component, and deployment) to fully express the five most useful views (use case, design, process, implementation, and deployment) comprising the architecture of a software-intensive system. Similarly, representational abstraction for simulation elements is likely to need a multifaceted descriptive approach.

Simulation fidelity is a complex function of the scope and discernible levels of the simulation as well as accuracy, precision, and other parameter quality characteristics. During recent years, substantial attention has been paid to describing simulation fidelity, with progress being made toward standardizing connotations for fidelity-related terms and an awareness of issues associated with simulation fidelity.[34] Widely accepted principles for determining required levels of fidelity and abstraction and approaches that will produce them have not yet evolved.

The failure to fully account for uncertainties and errors that may exist in the data used as the basis for models, algorithms, entity characteristics and behaviors, processes, and other aspects of a simulation is a common problem in simulation verification and validation.[35] This issue is closely associated with the subject of simulation fidelity and an important consideration in representational abstraction.

Knowledge engineering provides abstraction principles that can be helpful in developing a simulation conceptual model. Theoretical approaches to knowledge engineering typically break it into three phases: knowledge acquisition, knowledge elicitation, and knowledge representation. Such theoretical approaches usually identify three knowledge structures: declarative knowledge (why things work the way they do), procedural knowledge (how to perform a task), and strategic knowledge (the basis for problem solving). Typically different acquisition, elicitation, and representation techniques are used for each kind of knowledge. Unfortunately, these theoretical approaches do not yet allow abstraction to be performed as a scientific method; thus, abstraction remains an art.[36] It takes only a casual review of recent articles in such publications as the *Journal of Data and Knowledge Engineering*[37] to reveal that contemporary researchers in this arena often develop a "new" descriptive language (or dialect of a language) or formalism for the problem at hand because current techniques do not yet have broad, general application capabilities.

In order to develop a conceptual model which is clear, complete, consistent, and correct, the rationale presented earlier for simulation conceptual model decomposition should be used, and quality criteria from Teeuw and van den Berg[11] should be employed to judge abstractions. Again, these criteria are completeness, propriety (pertinence), clarity, consistency, orthogonality (modularity, the independence of aspects of the subject represented), and generality (as implementation independent as feasible).

As one develops a simulation conceptual model and evaluates it by these criteria, it is important to document how one assesses the model, and then to note why it changes in response to the evaluation and how criteria for a quality conceptual model are met more fully. Otherwise, the rationale for some changes (and their benefits) may be lost as time passes, and lessons learned from the conceptual model development will not be so readily available for use in subsequent developments.

It has been recommended that simulation conceptual model documentation employ the scientific paper approach, even if also employing the design accommodation approach by using an implementation-oriented descriptive format such as UML.[2] Nine items (listed below) are suggested for the description of a portion of the conceptual model (such as a simulation element) or the entire conceptual model in the scientific paper approach to documenting a simulation conceptual model:

1. Conceptual model portion identification
2. Principal simulation developer point(s) of contact for the conceptual model (or part of it)
3. Requirements and purpose
4. Overview
5. General assumptions
6. Identification of possible states, tasks, actions, behaviors, relationships and interactions, events, and parameters and factors for entities and processes being described
7. Identification of algorithms
8. Simulation development plans
9. Summary and synopsis

This list is functionally equivalent to the 10 items in the generic content guidelines from *IEEE/EIA Industry Implementation of International Standard ISO/IEC: ISO/IEC12207 Standard for Information Technology Software Life Cycle Processes* for describing a planned or actual function, design, performance, or process, i.e., date of issue and status, scope, issuing organization, references, context, notation for description, body, summary, glossary, and change history.[4]

## CONCLUSIONS

This article discussed the tenebrous topic of how to represent knowledge so that simulation requirements and specifications, as well as the simulation conceptual model connecting them, can be described completely, correctly, consistently, and clearly. When this occurs, the likelihood that the simulation will function as desired is enhanced. There are many horror stories from simulation developments that have no explicit conceptual model, a poorly (or only partially) developed conceptual model, or incomplete documentation of the simulation conceptual model. Simulation developers and users can avoid problems caused by inadequate conceptual models by following the suggestions presented here.

As indicated, much remains to be done before simulation conceptual models can be routinely developed with the rigor, precision, and formalism needed to significantly increase their correctness. The obstinate complexity of the logical, philosophical, and computational foundations of knowledge representation is not as widely appreciated as it should be.[38] Consequently, one must be pragmatic in developing a simulation conceptual model. One must accommodate the capabilities (and limitations) of those responsible for simulation requirements and those who will implement the simulation. Comprehensive, clear, correct, and consistent descriptions of requirements, specifications, and the conceptual model that can be understood by all are far more important than adherence to a doctrinaire emphasis on a particular methodology or formalism. Yet at the same time, it would be prudent for most involved in simulation development and use to move toward an increased application of formal methods to expand mathematically precise descriptions in simulation requirements, specifications, and conceptual models as far as circumstances will permit. This is a thrust which NASA has used effectively.[28,39]

Suggestions presented in this article will help the simulation community move toward more effective methods in developing simulation conceptual models, even though these suggestions are more general and limited than many, including myself, would prefer. Sources identified may help the reader to a broadened appreciation of the various aspects of conceptual model development. I have observed that few in the simulation community have very broad awareness of the available literature, especially in the arenas of simulation conceptual model development and related topics such as simulation fidelity and VV&A. This lack of awareness can prevent the use of best contemporary practices in simulation development, evaluation, and application.

The ideas presented in this article are applicable to acquisition, analysis, and training simulation application domains for both unitary and distributed simulations in the defense community as well as to scientific, technical, economic, social, educational, gaming, etc., simulations, whether unitary or distributed, in non-defense communities. There is much room for improvement in the development and descriptive methods for requirements, specifications, and conceptual models of simulations in all of these arenas.

## REFERENCES


[1] Cook, D., "Evolution of Programming Languages and Why a Language Is Not Enough To Solve Our Problems," *CrossTalk: J. Def. Software Eng.* **12**(12), 7–12 (Dec 1999).

[2] Pace, D. K., "Development and Documentation of a Simulation Conceptual Model," in *Proc. 99 Fall Simulation Interoperability Workshop* (Sep 1999), available at http://www.sisostds.org/siw/ (accessed 28 Feb 2000).

[3] *Recommended Practices Guide for Verification, Validation, and Accreditation* (*VV&A*), DoD Defense Modeling and Simulation Office (DMSO), available at http://www.dmso.mil (accessed 28 Feb 2000).

[4] Moore, J. W., "An Integrated Collection of Software Engineering Standards," *IEEE Software* **16**, 51–57 (Nov–Dec 1999).

[5] Sorensen, R., "Software Standards: Their Evolution and Current State," *CrossTalk: J. Def. Software Eng.* **12**(12), 21–25 (Dec 1999).

[6] Mayhew, D. J., *Principles and Guidelines in Software User Interface Design*, Prentice Hall, Englewood Cliffs, NJ (1992).

[7] *DoD Joint Technical Architecture*, Version 3.0 (29 Nov1999), available at http://www-jta.itsi.disa.mil/ (accessed 28 Feb 2000).

[8] *Conceptual Modeling Language—Syntax and Semantics for IDEF1X97* (*IDEFobject*), IEEE Standard 1320.2-1998, IEEE, Piscataway, NJ (1998).

[9] *An Intelligent Brokering Service for Knowledge-Component Reuse on the World Wide Web*, available at http://www.swi.psy.uva.nl/projects/IBROW3/home-ibrow.html (accessed 28 Feb 2000).

[10] *J. Conceptual Modeling*, available at http://www.inconcept.com/JCM (accessed 28 Feb 2000).

[11] Teeuw, W. B., and van den Berg, H., "On the Quality of Conceptual Models, in *Proc. 16th Int. Conf. on Conceptual Modeling* (3–16 Nov 1997), available at http://osm7.cs.byu.edu/ER97/workshop4/tvdb.html (accessed 28 Feb 2000).

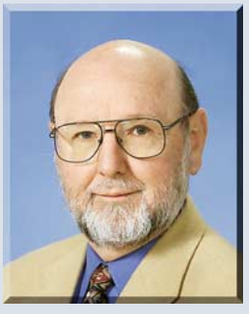
[12] Lindland, O. I., Sindre, G., and Solvberg, A., "Understanding Quality in Conceptual Modeling," *IEEE Software* **11**(2), 42–49 (Mar 1994).

[13] *Glossary of Modeling and Simulation Terms*, DoD Defense Modeling and Simulation Office (DMSO), available at http://www.dmso.mil (accessed 28 Feb 2000).

[14]*Mars Climate Orbiter*, available at http://lunar.ksc.nasa.gov/mars/msp98/orbiter/ (accessed 28 Feb 2000).
[15]Partsch, H. A., *Specification and Transformation of Programs: A Formal Approach to Software Development*, Springer-Verlag, New York (1990).
[16]Poston, R. M., "Automating Specification-Based Software Testing," in *Proc. IEEE Computer Society*, Los Alamitos, CA (1996).
[17]Grady, J. O., *System Validation and Verification*, CRC Press, Boca Raton, FL (1997).
[18]Nelson, M., Clark, J., and Spurlock, M. A., "Curing the Software Requirements and Cost Estimating Blues: The Fix Is Easier Than You Might Think," in *Program Manager*, Vol. XXVIII, No. 6, Defense Systems Management College (DSMC), DSMC Press, Ft. Belvoir, VA, pp. 54–60 (Nov–Dec 1999).
[19]Sheehan, J., Prosser, T., Conley, H., Stone, G., Yentz, K., and Morrow, J., "Conceptual Models of the Mission Space (CMMS): Basic Concepts, Advanced Techniques, and Pragmatic Examples," in *Proc. 98 Spring Simulation Interoperability Workshop*, Vol. 2, pp. 744–751 (Mar 1998).
[20]Johnson, T. H., "Mission Space Model Development, Reuse and the Conceptual Models of the Mission Space Toolset," in *Proc. 98 Spring Simulation Interoperability Workshop*, Vol. 2, pp. 893–900 (Mar 1998).
[21]Gaines, B. R., "An Overview of Knowledge Acquisition and Transfer," *Int. J. Man-Machine Stud.* **26**(4), 453–472 (Apr 1987).
[22]Sharp, J. K., "Validating an Object-Oriented Model," *J. Conceptual Modeling*, Issue No. 6 (Dec 1998), available at http://www.inconcept.com/JCM (accessed 28 Feb 2000).
[23]Hone G. N., and Moulding, M. R., "Developments in Application Domain Modelling for the Verification and Validation of Synthetic Environments: Training Process and System Definition." in *Proc. 99 Spring Simulation Interoperability Workshop*, Orlando, FL (15–19 Mar 1999), available at http://www.sisostds.org/siw/ (accessed 28 Feb 2000).
[24]Zeigler, B. P., Kim, T. G., and Praehofer, H., *Theory of Modeling and Simulation*, 2nd ed., Academic Press, New York (1999).
[25]Sarjoughian, H. S., and Zeigler, B. P., "The Role of Collaborative DEVS Modeler in Federation Development," in *Proc. 99 Fall Simulation Interoperability Workshop* (Sep 1999), available at http://www.sisostds.org/siw/ (accessed 28 Feb 2000).
[26]YEROOS Research Group, available at http://yeroos.qant.ucl.ac.be/yeroos.html (accessed 28 Feb 2000).
[27]Nissanke, N., *Formal Specification: Techniques and Applications*, Springer-Verlag, New York (1999).
[28]*NASA Formal Methods Guidebook: Formal Methods Specification and Verification Guidebook for Software and Computer Systems, Vol. I: Planning and Technology Insertion*, NASA/TP-98-208193 (1998), available at http://eis.jpl.nasa.gov/quality/Formal_Methods/index.html (accessed 28 Feb 2000).
[29]*SEI Interactive: Software Requirements Engineering* **2**(1) (Mar 1999) available at http://www.sei.cmu.edu/products/sei.interactive (accessed 28 Feb 2000).
[30]*World Congress on Formal Methods*, available at http://archive.comlab.ox.ac.uk/formal-methods.html (accessed 28 Feb 2000).
[31]Booch, G., Rumbaugh, J., and Jacobson, I., *The Unified Modeling Language User Guide*, Addison-Wesley, Reading, MA (1999).
[32]HLA Federation Development and Execution Process (FEDEP) Model, HLA Interface Specification, and Run-Time Infrastructure-Next Generation (RTI-NG), available from http://www.dmso.mil (accessed 28 Feb 2000).
[33]Pollack, A. F., and Baker, J. P., "Federation Decomposition: HLA Metrics," in *Proc. 99 Fall Simulation Interoperability Workshop* (Sep 1999), available at http://www.sisostds.org/siw/ (accessed 28 Feb 2000).
[34]Gross, D. C. (ed.), "Report from the Fidelity Implementation Study Group," 99S-SIW-167, in *Proc. 99 Fall Simulation Interoperability Workshop*, available at http://www.sisostds.org/siw/ (accessed 28 Feb 2000).
[35]Roache, P. J., *Verification and Validation in Computational Science and Engineering*, Hermosa Publishers, Albuquerque, NM (1998.)
[36]*Knowledge Engineering*, available at http://church.cse.ogi.edu/~walton/know_eng.html (accessed 28 Feb 2000).
[37]*J. Data and Knowledge Engineering*, available at http://www.elsevier.com/inca/publications/store/5/0/5/6/0/8/ (accessed 28 Feb 2000).
[38]Sowa, J. F., *Knowledge Representation: Logical, Philosophical, and Computational Foundations*, Brooks/Cole, Florence, KY (2000).
[39]*NASA Formal Methods Guidebook: Formal Methods Specification and Verification Guidebook for Software and Computer Systems, Volume II: A Practitioner's Companion*, NASA-GB-001-97 (1997).

ACKNOWLEDGMENTS: This article drew heavily upon work sponsored by the Defense Modeling and Simulation Office (DMSO), with oversight by the DMSO Verification, Validation, and Accreditation (VV&A) Technical Director, Simone M. Youngblood (as reported in a paper at the March 2000 Simulation Interoperability Workshop). The article also drew from sources identified in the reference list, the revised version of the DoD *Recommended Practices Guide for Verification, Validation, and Accreditation (VV&A)*, unpublished doctoral work in Turkey by Cüneyd Firat about conceptual modeling and analysis in HLA, and the work of the 1999 Conceptual Model Tiger Team established by Youngblood. Tiger Team members included Candace Conwell (SPWAR), Robert-Jan Elias (TNO-FEL, The Netherlands), Reuben Jones (Boeing), Averill Law (Averill Law & Associates), Bob Lewis (TecMasters), Michael Moulding (Cranfield University, UK), Dale Pace (JHU/APL), Terry Prosser (Logicon), Bob Senko (DMSO), Jack Shehan (DMSO), Susan Solick (TRAC), Dave Thomen (SAIC), and Bernard Zeigler (University of Arizona). Gary Coe (IDA) also helped.

## THE AUTHOR



DALE K. PACE is a member of APL's Principal Professional Staff assigned to the Information Analysis Group of the Joint Warfare Analysis Department. He studied mathematics and physics as an undergraduate at the University of Chicago (1957–1960), and received a B.D. from Capital Bible Seminary in 1969 and a Th.D. from Luther Rice Seminary in 1974 (his dissertation addressed contemporary praxis of the correctional chaplaincy). He came to APL in 1963, left in 1971 to become a jail chaplain, and returned in 1979. Dr. Pace is a specialist in operations research, systems analysis, wargaming and seminar gaming, scenario development, and defense analysis. From 1987–1989, he was APL's liaison with the Naval War College, where, as an adjunct professor, he developed and taught an elective course on technology and naval warfare. Dr. Pace is a member of the DMSO Verification, Validation, and Accreditation Technical Working Group and its Technical Support Team, with responsibilities in the area of conceptual model development. His e-mail address is dale.pace@jhuapl.edu.