Proceedings of the 1998 Winter Simulation Conference
D.J. Medeiros, E.F. Watson, J.S. Carson and M.S. Manivannan, eds.

SIMULATION MODELING AT MULTIPLE LEVELS OF ABSTRACTION

Perakath Benjamin
Madhav Erraguntla
Dursun Delen
Richard Mayer

Knowledge Based Systems, Inc.
1408 University Drive
College Station, Texas 77845, U.S.A

ABSTRACT

The purpose of this paper is to characterize the problem of multiple levels of abstraction in simulation modeling and to develop an approach that addresses the problem. In this paper, we describe the notion of abstraction and the technical problems associated with multiple levels of abstraction, how abstractions affect different activities during the simulation modeling process, a preliminary approach for addressing the problems associated with multiple levels of abstraction, the conceptual architecture of a simulation modeling environment that implements the proposed approach, and a summary of the research on questions of abstraction in simulation.

1 INTRODUCTION AND BACKGROUND

The phenomenon of multiple levels of abstraction in simulation modeling has been documented (Law and Kelton, 1991; Pegden et. al., 1990; Curry et. al., 1989; Pritsker, 1986). These authors note the importance of choosing, based on the simulation goals, the appropriate level of detail to include in a simulation model. They do not, however, address the problems associated with model development at multiple levels of abstraction, or the problems associated with the integration of legacy models at different levels of abstraction. The authors also fail to develop methods and techniques that address the problems associated with multiple levels of abstraction. More recent research in the areas of distributed simulation and the High Level Architecture (HLA) initiative have focused attention on some of these problems. However, the focus of these efforts has largely been on selecting appropriate federations based on accuracy/fidelity considerations (Nouragas and Watts, 1997; Foster and Yelmgren, 1997).

(Benjamin et. al., 1993) outlines our preliminary efforts for providing knowledge-based support to simulation modeling, including a heuristic approach to selecting appropriate levels of abstraction. Using those efforts as a foundation, this paper characterizes the broader problem of abstractions in simulation modeling, outlines an initial approach to addressing these problems, and briefly sketches an architecture that may be used to implement the outlined approach. The architecture enhances the Knowledge-Based Simulation Engine (KBSE) reported by the authors in (Erraguntla, et. al, 1994). The original research that led to the KBSE was partially supported by the NSF (KBSI, 1994a); partial support for the work presented here comes from the NASA Small Business Innovation Research (SBIR) program through the Kennedy Space Center (KBSI, 1996).

2 THE PROBLEM OF MULTIPLE LEVELS OF ABSTRACTION

By their very nature, models are developed at some level of abstraction and from some perspective. This section describes the notion of "multiple levels of abstraction" in modeling in general, and, in particular, with regard to simulation modeling. The *level of abstraction* of a model determines the amount of information that is contained in the model. The quantity of information in a model decreases with the levels of abstraction. Thus a "low level abstraction" model contains more information than a "high level abstraction" model. Because the term "quantity of information" is used in a subjective fashion, we will provide an example to illustrate the concept.

Figure 1 shows the concept of modeling abstraction. Model M transforms Input I to Output O. A decomposition of M into M1, M2, and M3 shows a detailing of input – output transformations that is hidden at the more abstract level. Thus, I1, I2, and I3 are transformed by M1, M2, and M3 to O1, O2, and O3, respectively, at the "lower" level of modeling abstraction. This shows how the quantity of information contained at the lower level is more than at the higher levels. Therefore, it is convenient to think of

abstractions as a mechanism to selectively "hide" information.

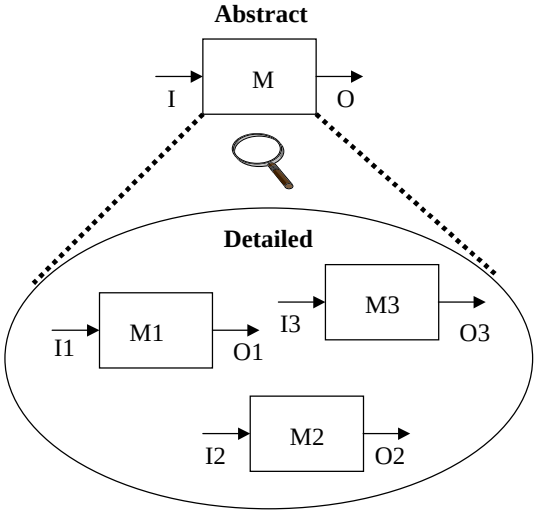

Figure 1. Abstractions in Modeling

The concepts of *abstraction* and *perspective* are different, although there is a strong relationship between them. We use the term perspective to refer to the mechanism that determines what set of model information is *relevant* to (and, therefore, needs to become an explicit part of a model) a given modeling goal. Abstraction level decisions are (as are other modeling decisions) perspective-dependent but these determine the quantity of information that needs to be associated with a model. Thus, abstractions determine *detail* (quantity of information) and perspectives determine *relevance*. As should be evident, the two are strongly coupled notions.

Establishing the level of abstraction is an important conceptual step in simulation modeling and is often done early in the model development life cycle. The choice of the "correct" level of abstraction is not always an easy one and often requires significant trial and error. A good modeling heuristic is to select the highest level of abstraction that will adequately address modeling objectives. Thus, for example, to determine the utilization of fork-lift trucks in a factory, it may not be necessary to model the manufacturing operations at a micro (detailed) level; it may be adequate to simply treat each manufacturing step as a unit task that takes a given amount of time and that requires a given set of resources. If, on the other hand, the modeling goal is to determine the distribution of time that the manufacturing operation resources reside in different states (such as "Waiting for Parts," "Waiting for Set Up," "Busy," "Waiting for Repair," etc.), then we need to develop a relatively fine-grained model of each manufacturing operation. Once the abstraction-level decision has been made, however, it significantly influences many other aspects of the simulation modeling life cycle. Moreover, in situations where previously developed (legacy) models must be integrated and harmonized for purposes quite different from their original use, the phenomenon of multiple levels of abstractions gives rise to several additional technical challenges.

The technical problems associated with simulation modeling and simulation can be grouped into four categories:

1. *Determining the correct level of abstraction.* Determining the correct level of abstraction refers to selecting the quantum of information that must be included in the model to help address the modeling goals. Thus, in an enterprise simulation-modeling context, determining the abstraction level involves answering questions such as "Should the model be constructed at the enterprise level, department level, or at the detailed task level within a sub function of a department?"

2. *Decomposition/Dis-aggregation.* Decomposition or Dis-aggregation refers to the conceptual task of taking a model artifact/concept at some level of abstraction and developing a set of modeling artifacts/concepts that contain more information about the model. As we've discussed earlier, a decomposition always produces a model that contains a greater amount of information. Decomposition may be applied to different kinds of model artifacts and concepts including modeling goals, performance metrics, activities, and objects. In addition, decomposition often entails data collection and knowledge acquisition. For example, to develop a detailing of the activity "Make Coffee" described earlier, one may need to interview the set of people who make coffee on a regular basis, determine how they make coffee, and then structure the results.

3. *Roll-up/Aggregation.* Roll-up or aggregation refers to the conceptual task of processing a set of modeling artifacts/concepts at some level of abstraction and generating a set of "higher level" modeling artifacts/concepts that are useful for decision making. The aggregated model artifacts contain a smaller quantity of information and often manifest themselves as a summary of the information contained at the lower level of abstraction. In addition, roll-up and aggregation often involve data and information transformation. For example, the *Weight* attribute of a ship may be computed by summing the values of the weights of the component subsystems of the ship. Similarly, the average *Duration* attribute of "Make Coffee" may be computed by adding the average durations of the sub activities "Prepare Coffee Pot," "Add Coffee Powder," "Add Water," "Activate Brewing," and "Pour Coffee."

4. *Integration/Harmonization.* Refers to the concept task of performing the needed changes to ensure that two or more different models or model fragments work together

properly. The problem is critical in situations where an attempt is made to re-use and integrate legacy models into a simulation model development effort. For example, suppose that a joint task force (Army, Navy, and Air Force) simulation model needs to be developed. Each entity has a large number of existing simulation models created for previous missions and the task entails that these models be leveraged and modified in developing the federation. Researchers from KBSI have developed an ontology-driven approach to the problem of integrating multiple enterprise simulation developed using standard modeling languages (Tissot and Crump, 1998). However, this approach does not address the problem of multiple levels of abstraction during model integration.

3 SOLUTION APPROACH

This section describes our approach to addressing the multiple-abstraction level problems described in the preceding sections. We begin by summarizing the technical problem and then describe our approach to addressing this problem.

3.1 Determining the Appropriate Level of Abstraction for Simulation Model Integration

One of the issues that needs to be addressed in integrating legacy simulation models is selecting the appropriate level of abstraction in each legacy model. Consider the legacy models shown in Figure 2.

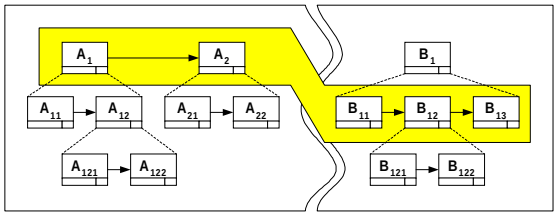

Figure 2. Selection of Appropriate Level of Abstraction for Integration

Assume that the two legacy models, A and B, need to be integrated. However, these two legacy models may have been developed independently and at different levels of abstraction. Thus, it might be more appropriate to integrate the top-level of Model A with the second level of Model B. Even within an individual model, it might be necessary to selectively decompose certain activities, in order to model them in more detail. Hence, selecting the appropriate level of abstraction in each model becomes an important issue in the integration of legacy simulation models. This section details some heuristics that can be used to provide knowledge-based assistance to this problem.

The approach outlined here extends the original approach presented in (Erraguntla et. al., 1994) to integrate multiple legacy models. Our approach also accommodates objects at multiple levels of abstraction, which was not supported before. Our philosophy for determining the *appropriate* level of abstraction can be summarized as follows:

1. If the user has any preferences concerning mixing and matching levels of abstraction, the simulation modeling system must accommodate these preferences. In this case, if any mismatches are noticed between the abstraction levels chosen by the user, the system will notify the user and will allow the user to resolve the mismatches.

2. Alternately, the modeling system will provide automated support for determining the appropriate levels of abstraction to integrate. The system determines the appropriate levels to integrate based on the goals of the simulation. A preliminary simulation model is created by integrating the different models at the highest level of abstraction. This initial simulation model is executed and analyzed. Selected portions of the integrated model are decomposed and the process is repeated until the entire simulation model is more or less at the same level of abstraction. A heuristic for determining the appropriate level of abstraction based on this philosophy is detailed below.

3.1.1 Heuristic for Determining the Appropriate level of Abstraction

Our heuristic to determine the appropriate level of abstraction contains the following steps.

Step I: Integrate legacy models at the top-most level.

Step II: Objects in different legacy models might be at different levels of abstraction. For these objects, apply the heuristics described in Section 3.1.1.1 to ensure object abstraction consistency.

Step III: Execute the preliminary simulation model and observe the simulation results.

Step IV: The processes in the different legacy models may be at different levels of abstraction. Based on the results of model execution, determine whether the different processes are at a consistent level of abstraction (Section 3.1.1.2). If process consistency is achieved, the procedure is terminated. Otherwise, a subset of the processes is decomposed. Process decomposition might again introduce object abstraction inconsistencies. In this case, the Step II procedure is repeated.

3.1.1.1 Achieving Object Abstraction Consistency

Objects residing in different legacy simulation models could also be at different levels of abstraction. In order to integrate legacy simulation models it is necessary to check that all the models refer to the objects at the same level of granularity. Our approach to ensuring object consistency is based on knowledge about two kinds of object relationships: Part-Of, and Sub-Kind-Of (Figure 3). Part-Of is used to represent the relationship between an object and its constituent parts. For example, two Programmers and a System Analyst might be Part-Of the Software Development Team. The Sub-Kind-Of relation is used to represent the generalization/specialization relationships between objects. For example, "General Purpose Milling Machine" and "Special Purpose Milling Machine" might both be Sub-Kind-Of the object "Milling Machine."

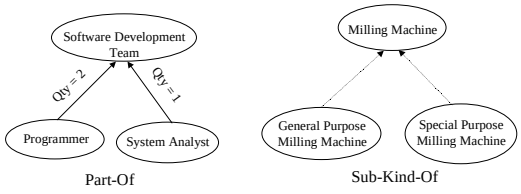

Figure 3. Object Abstractions

Different simulation modules that need to be integrated might have objects represented at different levels of abstraction. For example, a legacy simulation model might refer to the object "Software Development Team." Some other legacy model might refer to "Software Development Team" by its constituent parts: "Programmers" and "System Analyst." Similarly, a simulation model might refer to "Milling Machines," while another simulation model might individuate the machines by their type: "General Purpose Milling Machine" and "Special Purpose Milling Machine."

Detecting and resolving such differences in object abstraction modeling will become an issue if the different simulation models need to be integrated. Our approach uses the IDEF5 ontology-based object hierarchy to capture relationships between objects and to detect/resolve mismatches in legacy simulation models (KBSI,1994b). If the object abstraction is based on a Part-Of relation, consistency is achieved by replacing the object with its constituent parts. In our example, "Software Development Team" is replaced by two "Programmers" and a "System Analyst." On the other hand, if the object abstraction is based on the Sub-Kind-Of relation, then the specialized object type is replaced by its more general object type. That is, "General Purpose Milling Machine" and "Special Purpose Milling Machine" will be replaced by the more generic "Milling Machine." These examples illustrate how object consistency between multi-abstraction simulation models is achieved using a replacement strategy based on the Part-Of and the Sub-Kind-Of relationships.

3.1.1.2 Achieving Process Abstraction Consistency

The processes (behaviors) in different legacy models might also be at different levels of abstraction. Heuristic presented in Section 3.1.1.1 will likely eliminate object abstraction inconsistencies. However, the presence of *process* abstraction inconsistencies might also prevent the realization of simulation modeling goals. A general guiding principle is to represent processes having maximum impact on the current simulation goals in greater detail and to represent other processes at higher levels of abstraction. The heuristic detailed below is based on this simulation modeling principle.

1. Integrate and run the models at the current level of modeling abstraction.
2. Determine the *abstraction parameter* for the current modeling goal. An abstraction parameter is a parameter that provides a basis to decide whether or not to decompose a process. It is our experience that, for most manufacturing goals, time or cost could be used as the abstraction parameter.
3. Before running the simulation, apportion the abstraction parameter to each process. For example, if the goal of the simulation is related to processing time, then the abstraction parameter might be the processing time. Thus, the time the entity spends in the system is apportioned to each of the process steps.
4. Suppose that there are five process steps in the simulation model. Ideally, each of these processes should have roughly 20% of the abstraction parameter apportioned. If any process step has a high proportion of the abstraction parameter apportioned to it (say 40% instead of the expected 20%), it indicates that, with respect to the current goals of the simulation, that process step is modeled at a higher level of abstraction than other process steps in the model. Decomposing this process step, if possible, will facilitate capture, in more detail, those portions of the model that have maximum impact on achieving the goal. Hence, in the next iteration, decompositions of such process steps are included in the model.
5. Suppose that, as a result of applying this decomposition strategy, there are eight process steps in the integrated model in the next iteration. At this stage, we expect each of the process steps to have roughly 12.5% of the abstraction parameter. If not, the decomposition procedure in Step 4 is repeated on selected process steps in the model. Otherwise (that is, when the abstraction parameter values are nearly all equal), we have achieved process abstraction consistency.

3.1.2 Variations to the Heuristic

The heuristic presented in Section 3.1.1 (and detailed in Sections 3.1.1.1 and 3.1.1.2) prescribes how to determine the top-most abstraction levels in different models that can be integrated. For example, it can be used to determine that Level 1 of Model 1 is at the same abstraction level with Level 2 of Model 2 and, consequently, should be integrated. However, depending upon on the goals of the simulation and available time and budget, the modeler might like to develop the integrated model in more detail. For example, the modeler may want to use Level 2 of Model 1 and would like to determine the matching levels in other models. In such cases, the user can selectively decompose one or more models to the required level of abstraction and use the heuristic to determine the matching levels in other models.

3.2 Consistency of Information Between Different Abstraction Levels

In some simulation models, the different abstraction levels were developed from different perspectives and, hence, will have different information content. These differences may not arise only because of the abstraction and aggregation of objects and processes, but also due to the omission of detail and the degree of approximation. Consequently, it is not only allowable, but may also be *desirable*, to have different information content in different abstraction levels. Yet, even after accounting for the differences between different perspectives, information in the different abstraction levels may still remain inconsistent with each other. This is especially true in a collaborative modeling environment, where models at different abstraction levels are developed by different modelers. For example, an activity might have been modeled to require a mean time of 10 days in one abstraction, and its constituent sub-activities might have been modeled to require a total of 15 days in a lower level of abstraction.

Based on the goals of the modeling, some situations may warrant and legitimize these discrepancies, whereas, in other situations the two levels of abstraction might be considered inconsistent. In this section, we discuss consistency between different levels of abstraction and describe heuristic mechanisms to detect such inconsistencies.

3.2.1 Process Duration Consistency

Consider the situation in which activities are represented at multiple levels of abstraction (Figure 4). The duration of an activity and its constituent sub-activities could be different in different levels of abstraction. For example, the actual duration to execute an activity at run time might be affected by the resource requirements and resource availabilities; these requirements and availabilities could differ at different levels of abstraction. To determine the consistency of temporal information between the abstraction levels it is necessary to think of the temporal estimates as either optimistic or pessimistic.

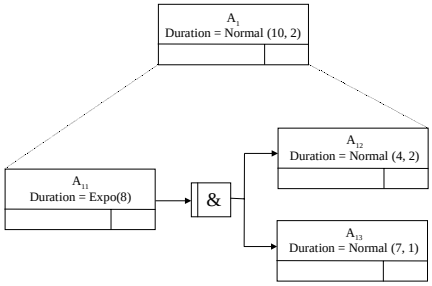

Figure 4. Process Duration Consistency

Based on the modeling goals and perspectives, it might be necessary for the time taken to execute an activity at a higher level of abstraction be greater than the time taken to execute its sub-activities at the lower level of abstraction. In this situation, we say that the higher level is a pessimistic estimate of the temporal information at the lower levels of abstraction. This paradigm is especially useful in top-down planning and design domains, where planners develop plans at a particular level and hand it over to sub-ordinate personnel for detailing. Planning personnel who use plans at higher levels of abstraction might be satisfied as long as the detailed plan meets the constraints imposed by the higher level plan. Similarly, based on modeling goals, it might be necessary that the temporal information at the higher level is an optimistic estimate of the temporal information at lower levels (that is, the duration at higher levels is smaller than the duration in lower levels).

Once the nature of temporal consistency requirements is identified between two abstraction levels, the issue is one of determining whether the two levels meet those consistency requirements. We have developed two kinds of mechanisms for determining consistency: 1) an activation-based method, and 2) an average-based method. In both the methods, the different levels of abstraction are executed with identical entity arrivals. In activation-based consistency, processing times of every entity in the two levels are compared. For example, the consistency requirement could be that Level 1 be an optimistic estimate of Level 2. That is, the duration should be smaller in Level 1 than in Level 2. If, in a particular activation, the processing time of an entity is greater in Level 1 than the processing time of the corresponding entity in Level 2, then the two levels are considered to be inconsistent. On the other hand, in an average-based determination, the

average processing times are compared instead of the individual processing times.

Note that the notion of consistency presented above (as well as other notions of consistency presented below) are not absolute but are instead based on specific model parameters such as entity arrival rates, resource availabilities, etc. Two abstraction levels, consistent in one situation, might turn out to be inconsistent with different entity arrivals or resource availabilities. So, in testing for consistency, these factors should be set to their appropriate levels based on the current modeling goals. Also, note that the measure focuses on whether durations in one level are shorter/longer than the other level. No attempt is made to quantify the degree of closeness.

3.2.2 Object Consistency

With respect to objects, two types of consistency between different abstraction levels can be defined: 1) object definition consistency, and 2) object use specification consistency. Object definition consistency is concerned with whether objects specified in different levels of abstraction, after accounting for the differences in perspective, are in a consistent state. Object use specification consistency evaluates whether the usage (duration and utilization) of objects at different levels are consistent.

Object Definition Consistency:
In general, objects occurring in different levels of abstraction could be different. However, the differences should be based on certain valid modeling guidelines. The following list contains the types of differences between abstraction levels we consider to be acceptable and mandated by the different perspectives.

1. *Omission*: A higher abstraction level might omit an object that occurs in a lower level of detail. However, if there is an object at a higher level of abstraction which is not represented in any form at a lower level, we highlight the difference to the user as a potential source of inconsistency.
2. *Aggregation*: A higher level model might aggregate objects occurring at a lower level based on the Part-Of relation (Section 3.1.1.1). For every aggregate object occurring at a higher level process, one or more constituent parts of the object must be used in the process decomposition. Otherwise, we highlight the difference as a potential source of inconsistency. Also, even though it might be acceptable for lower abstraction level to aggregate objects occurring in higher abstraction levels, we nonetheless highlight such object occurrences as potential sources of inconsistency.
3. *Substitution*: A higher level model can use generic object substitutes occurring at lower levels of abstraction. Such substitution can be based on the Sub-Kind-Of relation (Section 3.1.1.1). If an object occurs at a high level of abstraction, and, if the same object, its constituent parts, or its sub-kind (that is, its specialization) do not occur at a lower level, we highlight the difference as a potential source of inconsistency.

The IDEF5 ontology modeling methodology (KBSI, 1994b) is used to capture these kinds of relationships between objects and to test and validate object definition consistency between different abstraction levels.

Object Use Specification Consistency:

Just as with process duration consistency, object use specification consistency can be determined at the activation level or at the average level. In activation based consistency, the same entity arrivals are simulated using the different abstraction levels. For every activation (entity flow), whether the time interval of usage of an entity at a lower level is a sub-set of the time interval of the usage of either the entity or its aggregate or its substitute at the higher level is closely checked. Comparisons are not made if an object is omitted at higher levels of abstraction. Alternately, the user can specify that the consistency required is the time interval of an object use at the lower level when this time interval is a superset of the time interval of the use of the equivalent object at the higher level.

Consistency checks could be performed for the average of all the activations instead of checking for each activation. In average based checking, utilizations are compared instead of time intervals. Note also that these consistency checks are dependent on the entity arrival rates and the resource schedules. During consistency checking, these factors should be set to levels appropriate for their intended use.

3.3 Data Aggregation

Sometimes it might be necessary to design or run a simulation model at a particular level of abstraction, but present results at a higher level of abstraction. This could happen when simulation model users are at different levels in an organization hierarchy. Instead of running simulation models at multiple levels, it would be more economical to run simulation models at the lowest abstraction user level, roll-up or aggregate the information, and then present the aggregated information at higher levels. We have developed two kinds of mechanisms to perform data aggregation: 1) qualitative and 2) quantitative. For quantitative roll-ups, the simulation model is designed/executed at the lower level of abstraction and heuristics are designed to aggregate and present

information at higher levels of abstraction. These data aggregation heuristics are average based.

The qualitative heuristics are used mainly during the conceptual and detailed simulation model design phases of a simulation development effort. Qualitative aggregation mechanisms are typically used to determine estimates of minimum, maximum, and expected values of parameters. A more detailed description of these qualitative aggregation heuristics is given in (Benjamin et. al., 1994; KBSI, 1997).

4 KNOWLEDGE-BASED SIMULATION ENGINE ARCHITECTURE

This section outlines the elements of an architecture that implements the multi-abstraction mechanisms described in Section 3. This architecture is part of a more comprehensive architecture for knowledge based simulation called the Knowledge based Simulation Engine (KBSE) (Erraguntla et. al., 1994). Previous implementations of KBSE provided knowledge-based support for simulation model generation starting from a description of modeling goals and structured system descriptions (KBSI, 1994a). KBSE provides knowledge-based assistance for system description capture, goal capture, boundary determination, goal decomposition, abstraction level determination, and executable simulation model generation. The authors are currently refining the Abstraction Level Determination module of KBSE based on the concepts presented in this paper. A detailed description of a previous version of KBSE is given in (Erraguntla, et., al., 1994). The KBSE modifications that implement mechanisms to support multi-abstraction level simulation model development are shown in Figure 5.

Simulation models at multiple levels of abstraction can be generated using the Simulation Model Generator. The Simulation Model Generator contains KBSE modules that assist in the automatic generation of executable simulation models from structured system descriptions and a set of modeling goals. These modules communicate with the Consistency Validator and the Data Aggregator for addressing issues arising out of multiple levels of abstraction.

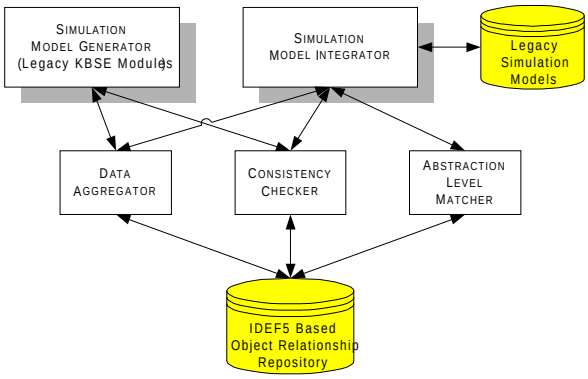

Figure 5. KBSE Components for Multi-Abstraction Modeling

The Consistency Validator checks for consistency between different levels of abstraction in a simulation model, as detailed in Sections 3.1 and 3.2. The Data Aggregator is used to roll-up data from the simulation designs or simulation executions at low levels of abstraction and present this information/data for users at higher levels of abstraction. Integrated simulation models can also be developed by integrating legacy models using the Model Integrator. The Model Integrator makes use of the Abstraction Level Matcher to determine the appropriate levels to integrate in the legacy models (Section 3.1). The Model Integrator also maintains two way communication with the Data Aggregator and with the Consistency Validator. The Data Aggregator, the Consistency Validator, and the Abstraction Level Matcher make use of an IDEF5-based object-relational repository to manage the dependencies between the different objects and processes in the simulation model.

5 SUMMARY

This paper described: 1) a characterization of the problem of multiple levels of abstraction associated with simulation modeling, 2) an approach that addresses a subset of these problems, and 3) an architecture that implements the approach. The work described here addresses an important, albeit poorly addressed, problem associated with simulation model development. The approach described in the paper has the potential to produce significant productivity gains to the simulation modeling process. We hope that the new concepts presented in this paper will trigger additional research initiatives in this important and technically challenging area.

REFERENCES

Benjamin, P. C., Fillion, F., Mayer, R. J., Blinn, T. M. 1993. Intelligent Support for Simulation Modeling: A Description-Driven Approach. *1993 Summer Computer Simulation Conference*, 273-277. Boston, MA.

Benjamin, P. C., Cullinane, T. P., Mayer, R. J., and Menzel, C.P. The Role of IDEF0 and IDEF3 in Business Process Improvement. *Proceedings of the May 1994 IDEF User Group Conference*, 158-170. May 23-26, 1994, Richmond, Virginia.

Curry, G. L., Deuermeyer, B. L., Feldman, R. M. 1989. *Discrete Simulation: Fundamentals and Microcomputer Support*. Oakland, CA: Holden-Day, Inc.

Erraguntla, M., Benjamin, P. C., Mayer, R. J. 1994. *An Architecture of a Knowledge-Based Simulation Engine*, 1994 Winter Simulation Conference, 673-680.

Foster, L., Yelmgren, K. 1997. *Accuracy In DoD High Level Architecture Federations*, 1997 Fall Simulation Interoperability Workshop, Volume I, 405-416.

KBSI. 1994a. *Knowledge Based Assistant for Simulation Model Generation from IDEF3 Descriptions*, NSF SBIR Phase II Final Report, Contract No. III-9123380.

KBSI. 1994b. *IDEF5 Method Report*. Knowledge Based Systems Inc. College Station, TX.

KBSI. 1995. *IDEF3 Method Report*, Knowledge Based Systems, Inc. College Station, TX.

KBSI. 1996. *Optimization Modeling Assistant*, Phase II NASA SBIR Proposal, Contract No. NAS 10-12257.

KBSI. 1997. *Cost Benefit Analysis Support Environment*, Phase II final report to Hanscom AFB, Contract No. F19628-95-C-0045.

Law, A. M., Kelton, W. D. 1991. *Simulation Modeling & Analysis*. New York: McGraw-Hill, Inc.

Nouragas, P., Watts, N. 1997. *Fidelity Assessment in Aviation Related Simulation Applications*, 1997 Fall Simulation Interoperability Workshop, Volume II, 719-725.

Pegden, C. D., Shannon, R. E., Sadowski, R. 1990. *Introduction to Simulation Using SIMAN*. Hightstown, NJ: McGraw-Hill, Inc.

Phillips, D. T., Ravindran, A., Solberg, J. 1976. *Operations Research: Principles and Practice*. New York: John Wiley & Sons.

Pritsker, A. A. 1986. Introduction to Simulation and SLAM II. New York: John Wiley & Sons, Inc.

Tissot, F., Crump, W. 1998. An Integrated Enterprise Modeling Environment. To appear in the *International Handbook of Architectures of Information Systems*, Volume 1, Springer-Verlag.

AUTHOR BIOGRAPHIES

PERAKATH C. BENJAMIN is the Vice President of Research at Knowledge Based Systems, Inc., College Station, Texas. He received his Master's degree in Industrial Engineering from the National Institute for Training in 1983 and his Ph.D. in Industrial Engineering from Texas A & M University in 1991. He has over 12 years of professional experience in systems analysis, design, development, testing, documentation, deployment and training. Dr. Benjamin is the principal investigator or project manager for a number of NSF, DOD and NASA projects.

MADHAV ERRAGUNTLA is a Research Scientist at Knowledge Based Systems, Inc., College Station, Texas. He received his Master's degree in Industrial Engineering from the National Institute for Training in 1989 and his Ph.D. in Industrial Engineering from Texas A & M University in 1996. Dr. Erraguntla conducted extensive research, and published papers in planning, simulation, costing, optimization, neural networks, qualitative reasoning and fuzzy logic. Currently, Dr. Erraguntla is a Research Scientist at KBSI.

DURSUN DELEN is a Research Associate at Knowledge Based Systems, Inc., College Station, Texas. He received his Master's degree in Industrial Engineering from the Yildiz University, Istanbul, Turkey, in 1988; and his Ph.D. in Industrial Engineering and Management from Oklahoma State University, Stillwater, Oklahoma, in 1997. He has more than five years of industrial experience in information systems analysis and design. His research interests include systems modeling, discrete event simulation, object-oriented modeling, knowledge representation and artificial intelligence.

RICHARD J. MAYER received a Master of Science degree in Industrial Engineering from Purdue University in 1977and Ph.D. in Industrial Engineering from Texas A&M University in 1988. He became an assistant professor of Industrial Engineering at Texas A&M in 1989, and was promoted to associate professor with tenure in 1994. From 1984 to 1997, Dr. Mayer was Project Manager and Principal Investigator on 54 funded research efforts at Texas A&M University's Knowledge Based Systems Laboratory. Currently Dr. Mayer is President and Senior Research Scientist at KBSI.