A Model–Based Approach to Component Simulation Development

Perakath Benjamin (pbenjamin@kbsi.com)
Dursun Delen (ddelen@kbsi.com)
Richard Mayer (rmayer@kbsi.com)
Knowledge Based Systems, Inc. (www.kbsi.com)

Tim O'Brien
NASA Kennedy Space Center

1.0 Motivations

The increasing complexity of systems has enhanced the use of simulation as a decision–support tool. Often, simulation is the only scientific methodology available to practitioners for the analysis of complex systems. However, only a small fraction of the potential practical benefits of simulation modeling and analysis have reached the potentially large user community because of the relatively high requirement of time, effort, and cost needed to *build* and *successfully use* simulation models.

Current simulation practice (i) is afforded little automated support for the initial analysis, problem solving, and design tasks which are largely qualitative in nature, (ii) involves the unproductive use of time from both the domain expert and the simulation analyst in many routine tasks, and (iii) suffers lack of widespread acceptance by decision makers due to a number of factors including a) the semantic gap between the description of a system internalized by the decision maker and the abstract model constructed by the simulation modeler, b) the relatively long lead times and communication efforts required to produce a simulation model, and c) the extensive training and skill required for the effective design and use of simulation modeling techniques [Erraguntla et. al, 94; Delen et. al 98].

Recent advances in the area of simulation modeling have focused on improving simulation modeling languages. These advances have attempted to reduce the semantic gap between a simulation model design and the corresponding executable simulation program. They represent important advances for improving the productivity of simulation modelers, but do little to aid the non–simulation trained decision maker. This situation is analogous to traditional CAD systems that aid a draftsman in the production of part drawings but provide no support for the actual design decisions behind those specifications.

This paper describes a model–based approach that seeks to address the above problems (MODSIM). The approach includes (i) the application of the IDEF3 and IDEF5 methods for simulation and analysis model specification, (ii) the automatic generation of executable component simulations from IDEF–based models, and (iii) the use of model and component libraries to facilitate effective knowledge capture and re–use over extended periods of time.

Section 2 describes the model–based solution concept. The MODSIM concept of operation is described in Section 3. Section 4 summarizes the MODSIM architecture. The benefits of the research and opportunities for further work are outlined in Section 5.

2.0 Solution Approach

2.1 The Solution Concept

A key solution concept underlying the MODSIM architecture is the individuation of three levels of abstraction to facilitate simulation modeling and analysis (i) Domain Level , (ii) Design Level, and (iii) Execution Level ().

The *Domain Level* refers to the collection of structured knowledge that encapsulates information about the problem area that is targeted by the simulation modeling and analysis effort. We assume that this information is available in a structured and re–usable form, for example, IDEF5 ontology models and IDEF3 process models.

The *Design Level* refers to the collection of models that specify the operation of the different phases of the simulation modeling and analysis effort. In particular these models provide a specification for simulation input analysis, simulation model execution, simulation experiment specification, and simulation–driven search and optimization specification.

The *Analysis Level* refers to the collection of data and information that is generated by the execution of simulations, analysis, and optimizations. This information is generated by simulation engines, experimental analysis tools, output analysis tools, and search and optimization tools.

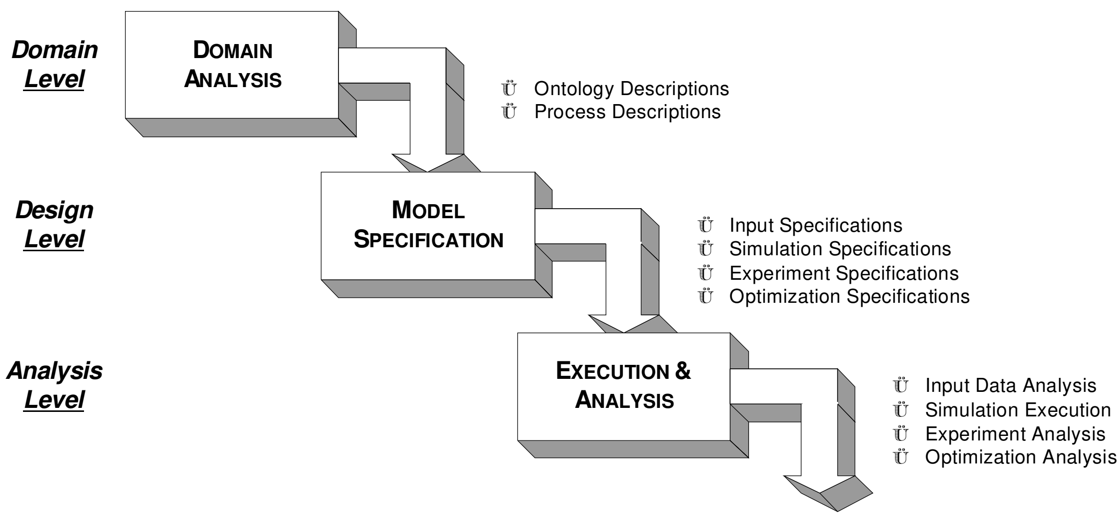

Figure 1. Separation of Levels Extends Re–use Scope

Separation of levels enables different kinds of re–use and provides the conceptual framework for component–based simulation. Maintaining structured domain models facilitates re–use over multiple domains (e.g., manufacturing, logistics, sales, military mission planning, threat assessment, etc.). Maintaining simulation model specifications enables re–use across multiple simulation execution and analysis tools (e.g. different vendor tools and components may be used for different simulation tasks (input data analysis, simulation execution, experiment analysis, simulation output analysis, optimization, etc.). The latter type of re–use allow simulation end users to switch between multiple component simulation tools for different tasks in the simulation life cycle (that is, “plug and play” using multiple simulation tools and utilities).

Concept of Operation

The activities supported by the MOSIM solution architecture and the relationships between these activities are described in the following paragraphs ().

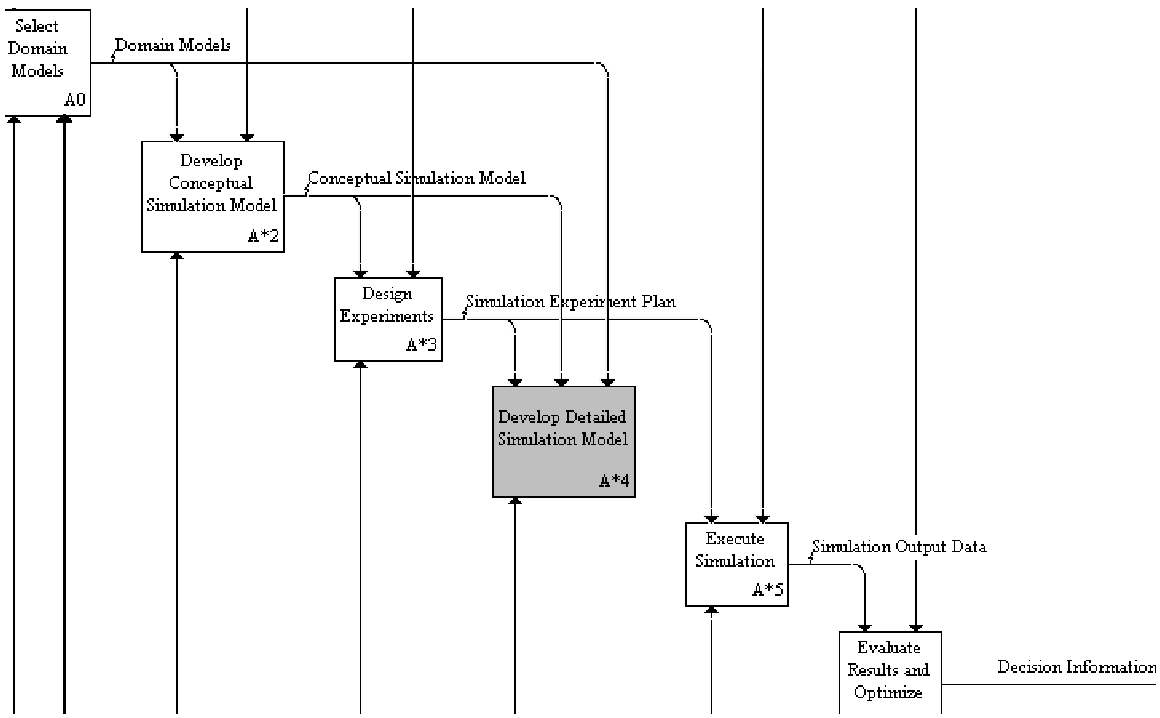

Figure 2. MODSIM Concept of Operation

3.1 Select Domain Models

An important first step is to select appropriate domain models from the domain model library. The domain models provide structured information about the domain of interest that will be used to construct the simulation model. Re–use of organized domain knowledge increases the efficiency of the modeling process through better knowledge management. It reduces dependence on human domain experts. Domain knowledge, once captured and stored in a library, can be repeatedly re–used for different simulation models. Two kinds of domain models are useful —IDEF3 process models and IDEF5 ontology models (and).

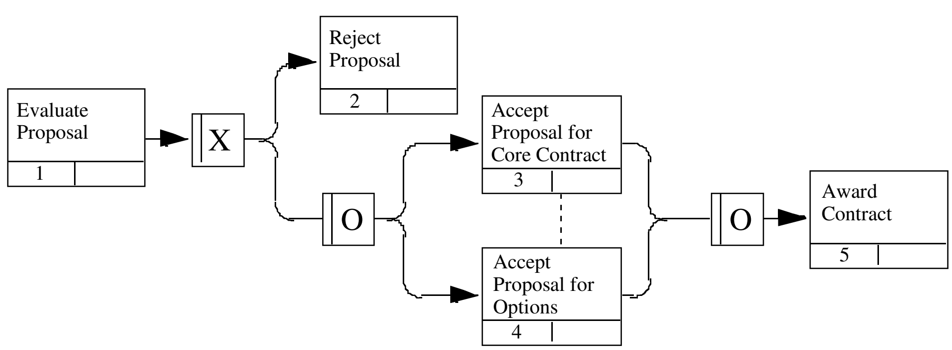

Figure 3. Example IDEF3 Process Flow Diagram

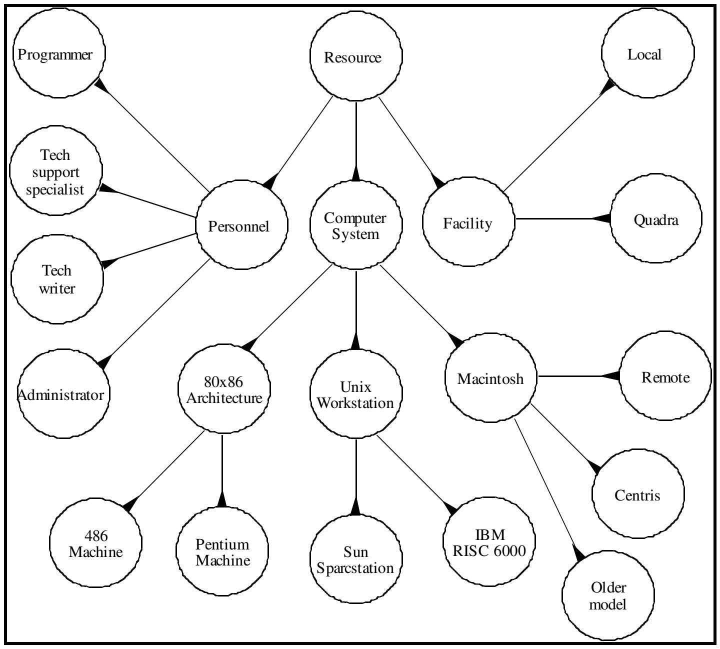

Figure 4. Example IDEF5 Classification Schematic

3.2 Design Conceptual Model

The construction of a conceptual or structural model is typically carried out by an analyst as an undocumented thought process rather than as an explicitly represented design activity. In addition to hindering the modeling effort, the lack of a facility to explicitly represent the conceptual model design also creates problems in re–use of such designs. In practice, the final executable model is often the only model documentation that exists, since none of the thought processes followed in model design, nor the assumptions made, are documented anywhere in a systematic manner. In order to tackle these problems and to better support the entire modeling process, we need to not only understand the cognitive processes involved in the modeling process, but also need a way of explicitly representing and reasoning with both the process and the output of the process, i.e., the conceptual model itself. We developed an adaptation of the IDEF3 process modeling language for conceptual simulation model design, called the IDEF3 Conceptual Modeling Language (I3CML). I3CML provides the development of conceptual simulation models from two perspectives (i) process–centered perspective (using the IDEF3 process flow mechanisms) and (ii) object–centered perspective (using the IDEF3 object state transition mechanisms). I3CML includes a rich library of re–usable generic simulation process types that can be tailored for particular simulation application domains using the IDEF3 and IDEF5 domain models described earlier. Example I3CML simulation process types are shown in .

Table 1. Example I3CML Simulation Process Types

Simulation Process Type	Description
Create/Destroy	A process that creates or destroys objects in the simulation model. Typically the objects created are flow objects (entities).
Transformation	A process that transforms an object in the simulation model. Subtypes of this process include Assembly, Disassembly, Cloning, Batching, and Simple State Change. Transformation process types encapsulate commonly re–occurring behavior types in a variety of application domains.
Transportation	A process that physically moves objects from one location to another.
Logical	A process that facilitates logical operations in the model. Subtypes include attribute value change and decision logic assignment

The I3CML object–centered modeling artifacts are based on the IDEF3 object state transition schematics. These allow for the description of behavior by describing the relevant object states, specifying the allowable transitions between these states, and defining the conditions governing these transitions. An example I3CML diagram that illustrates state transitions for a "resource" object type is shown in .

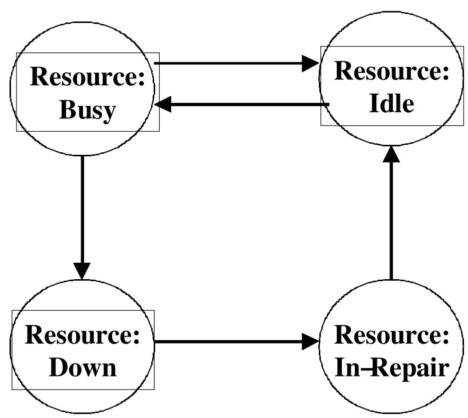

Figure 5. Example I3CML Object State Transition Schematic

The conceptual modeling process comprises several inter–related activities that are described in the following paragraphs.

Determine/Classify Modeling Objective

An important first step in the development of the conceptual model is to determine the specific goals of the simulation study based on the "question/demand for decision data" given by the domain expert. The capture of the question statement as an unstructured description. Consequently, there is a need to refine it further in order to extract the specific goals of the study. The process of refinement, which is performed by the analyst, is based primarily on his interpretation of the query statement and a reasoning mechanism to map this interpretation into a specific goal(s). This reasoning process is often a combination of qualitative and rule–based mechanisms. This reasoning uses the analyst's past experience and knowledge, but also makes extensive use of the constraints of the current description. During the course of such reasoning, the analyst often needs additional information or clarifications from the domain expert in order to clearly identify the user requirements.

The modeling objective plays a key role in determining the structure of the model to be developed, as well as in establishing the boundaries of the system to be analyzed, the level of detail to be included in the model, and the performance measure(s) to be estimated from running the simulation model.as further detailed in the following sections.

Determine Object Roles, Boundary and Level of Detail

• *Establishment of model boundaries*

One of the early activities in developing the conceptual model is the selection of the part of the system to be studied. The choice of boundaries is very closely linked to the specific goals of the analysis. This decision about boundaries is an important step since it gives perspective to the entire simulation study. As it turns out, a description is partial including only those portions of the system which are of special interest to the domain expert. While this might provide clues as to the boundaries chosen for the model, it might occasionally also become necessary to either ask for additional information about the system or to exclude parts of the description from the boundaries. The reasoning process in mapping the analysis goals to the boundaries is based mainly on the analyst's common–sense and domain knowledge.

• *Establishment of level of abstraction*

Once the boundaries of the model have been chosen, the analyst proceeds to select the level of abstraction to be used in modeling the system elements that are included within the boundaries. This activity is significantly impacted by the goals of the analysis. Our observation is that while doing this, the analyst adopted this simple principle: Include only those elements of a system that are relevant to the objective, and do so at the highest level of abstraction. One of the problems observed in carrying out this activity was that it is often difficult to tell which portions of the system will have an influence on the key performance measures of interest. Another principle which was observed in practice is: When in doubt about whether to include a particular subsystem, include it in the model.

• *Identification of model objects and roles*

This step refers to the selection of objects from the description to be included in the simulation model and the specific role that these objects will play in the model. Our research indicates that the reasoning mechanisms involved in carrying out these activities are rather unstructured and hence difficult to make explicit.

3.3 Design Simulation Experiments

Design Strategic Experiment Plan

Designing a strategic experiment plan refers to the process of 1) deciding upon the metrics which evaluate the performance of the simulation model with respect to the goals of the study, 2) designing instrumentation to generate the data needed to evaluate the performance metrics, and 3) specifying the strategic plan of experiments to generate this data at minimum cost.

The performance measures of the simulation model often do not directly give insights or answers to the query posed by the domain expert. However the purpose of building the simulation model in the first place was to provide the information required to answer the domain expert's query. Thus, the query (which is often correlated to the business goals of the domain expert) needs to be mapped onto the performance metrics to be estimated by the simulation model. For example, consider the following query from a manufacturing manager: "How can I streamline my production?" An underlying business goal which may have prompted this query could be that of improving utilization levels of bottleneck machines. Thus this query could be mapped onto performance metrics which will measure resource utilization within the manufacturing system.

Our research indicates that the knowledge needed to support the above process includes domain–specific and simulation modeling knowledge and that the mechanism of generating this mapping often involves qualitative reasoning.

Once the performance metrics have been specified, the simulation model has to be instrumented to facilitate the capture of data needed to calculate these metrics. This involves installing probes into the model which would help collect data over time and then process it into meaningful observations of model behavior. The reasoning involved in the design and placement of appropriate probes is often straightforward and could be expressed in terms of a set of simple rules.

Once the performance metrics have been chosen and appropriate probes have been designed, we need to generate a systematic plan of experiments which would enable the model to be executed at different experimental conditions so that the relationships between the performance metrics and the independent variables of the model can be investigated. These relationships would in turn focus attention on a subset of variables which have a significant effect on the value of the performance measures. These form the basis for the suggestion of possible answers to the domain expert's query.

A key issue in determining the plan of experiments is the cost of experimentation. The chosen experimental plan needs to generate the needed information with the minimum number of experiments. In addition to providing efficiency of experimentation, a scientific plan of experiments ensures that the analysis done with the output is statistically valid. An intimate knowledge of the science and art of the statistical design of experiments, in addition to domain–specific knowledge, is necessary to design the (statistical) plan of experiments

Design Tactical Experiment Plan

The tactical experiment plan refers to those activities which determine the detailed experiment specifications of each individual simulation run. The major decisions taken at the tactical planning stage include determining the length of each simulation run and the number of runs for each experimental condition. Early in this process, a decision whether to treat the simulation as either 'terminating' or 'non–terminating' must be made [Law and Kelton, 91]. Briefly, the distinction is based on whether we are interested in the *steady state* or the *transient* behavior of the model. Often, this decision can be made based on previous knowledge of the domain behavior and some knowledge of statistics. However, in some instances it might be necessary to execute a preliminary model and perform some analysis of the output. If the latter is required, we need to go ahead with the construction of the detailed model. Once we decide whether the simulation is terminating or non–terminating, we can proceed with the determination of the run length and the number of runs. These calculations are based primarily on statistical procedures [Law and Kelton, 91].

Formulate Optimization Design

Finally, a search–based optimization model is formulated. The search–based optimization techniques supported by MODSIM are Simulated Annealing (SA) and Genetic Algorithms (GA). Optimization using SA and GA involves the specification of search and optimization architecture and parameters. Automated support is provided for this activity in order to shield the end user from the complexities of SA and GA design. An example MODSIM Optimization Design user interface screen is shown in .

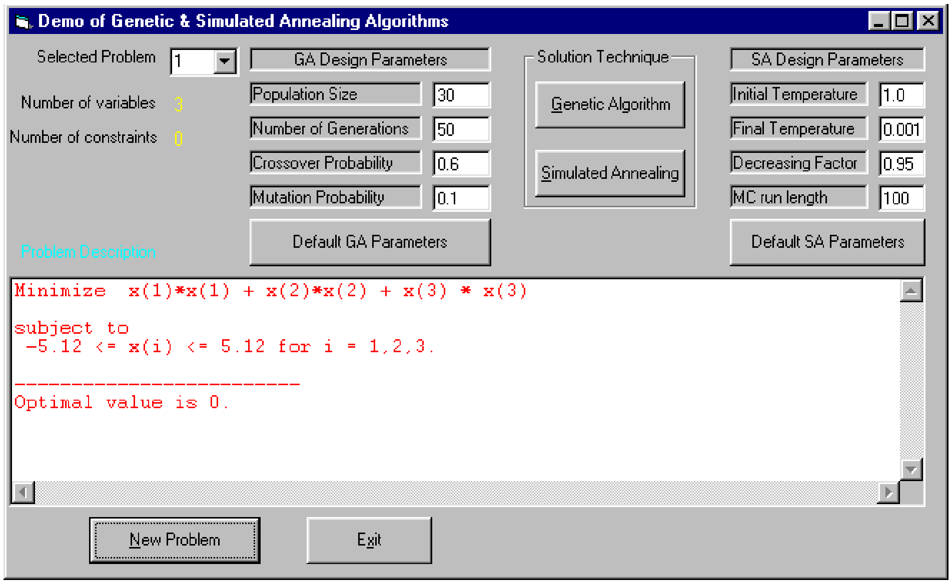

Figure 6. Search–Based Optimization Model Design Specification Interface

MODSIM automatically generates executable code that is interpreted by an optimization engine that performs search–based optimization.

3.4 Develop Detailed Simulation Model

The detailed simulation model design involves formulating the model structure and logic and verifying and validating the model.

Design Model Structure and Logic

Model structure and logic refers to a characterization of the relations between activities in the model. An activity represents the dynamic behavior that comes about when objects interact with each other. The model structure refers to the characterization of this dynamic behavior. For instance, if an activity is a manufacturing process, then its characterization will relate to specifying its processing time, which qualifies the behavior that occurs when a part is processed on a machine. There are two types of model logic – *flow logic* and *decision logic*. Flow logic is the specification of the flow path of all the objects through the system. Decision logic refers to the set of methods used to choose between alternative state transitions which characterize the dynamic behavior of the system. For example, the specific scheduling rule used to load a machine with parts in a manufacturing system will be part of the decision logic for that system.

Typically, an analyst starts by constructing a skeletal representation of the structure and logic. With reference to the elements of the I3CML language, the structure and flow logic is typically associated with process boxes and decision logic maps onto junctions. The modeling constructs associated with a process box are related to the dynamic behavior of the objects which are contained within it. The decision logic that is associated with junctions can be of three kinds: probabilistic, conditional or deterministic [Pegden at. al 90; Pritsker 86]. System information such as the part routings, schedules, distance between stations, and starting conditions, needs to be incorporated into the model structure and logic wherever possible. If such information is not included in the description, it may have to be gathered with the help of the domain expert or may be found in the query statement itself.

The model structure and logic will be successively refined in a stepwise manner until the conceptual model is complete.

Verify and Validate Model

Model verification and validation are important activities that are carried out once the simulation design reaches a satisfactory level of completion. *Model verification* is ascertaining whether the model behaves as intended by the designer. This task is often performed incrementally during simulation model design. Verification is based on common sense rules that evaluate model completeness and consistency. *Model validation* is ascertaining whether the model is a reasonable abstraction of the real world system it is intended to represent [Philips et. al, 76]. MODSIM provides automated support for model verification. The end user will have to validate the model using (i) analysis data generated by the environment, (ii) domain information provided in the domain models, and (iii) input from human experts familiar with the real world system being studied.

3.5 Execute Simulation

The simulation model specification is used to generate executable simulation code that is interpretable by a simulation engine. Our research shows that it is useful to represent the simulation model specification in an intermediate form before actually translating it to executable simulation code. This intermediate and *neutral* model specification is useful for two reasons:

- To provide greater expressiveness to the *intent* of the model/modeler. State–of–the–art simulation languages do not provide an adequate degree of expressiveness, in the sense that the model as it exists in the mind of the modeler is quite different from the model as encoded in a traditional simulation language.
- To provide a *neutral* representation of the model. The main advantage of building a neutral specification is that it gives the analyst the freedom of choosing from a variety of possible target simulation languages. This gives the analyst flexibility since different target languages are inherently advantageous for specific classes of models. For instance, a language that is effective for discrete simulation may be inappropriate for continuous simulation.

The simulation experiments are executed and output data is collected. Animations of the execution provide visual feedback to the modeler and provide a mechanism to communicate dynamic aspects of the represented system to the end user. The MODSIM simulation engine component provides this functionality.

3.6 Analyze Output and Optimize

Analyze Output
Output analysis refers to the detailed analysis of output leading to the generation of data for decision making. Output analysis bridges the model–building and the decision–making processes. Output analysis involves a variety of activities, including (i) formulating appropriate output metrics, (ii) identifying and quantifying output correlation, (iii) statistical estimation (averages and confidence intervals), (iv) initialization bias elimination. Component statistical analysis tools provide the output analysis capability in MODSIM.

Perform Optimization
Sensitivity analysis and optimization provide additional information for decision making. MODSIM facilitates search based optimization that uses simulation as a performance measurement mechanism. The Simulated Annealing (SA) and Genetic Algorithms (GA) specifications developed during the design phase are used to automatically generate executable code (see Section 3.3). The optimization code is interpreted by the MODSIM optimization engine that performs search–based optimization.

4.0 Architecture

The solution architecture is shown in .

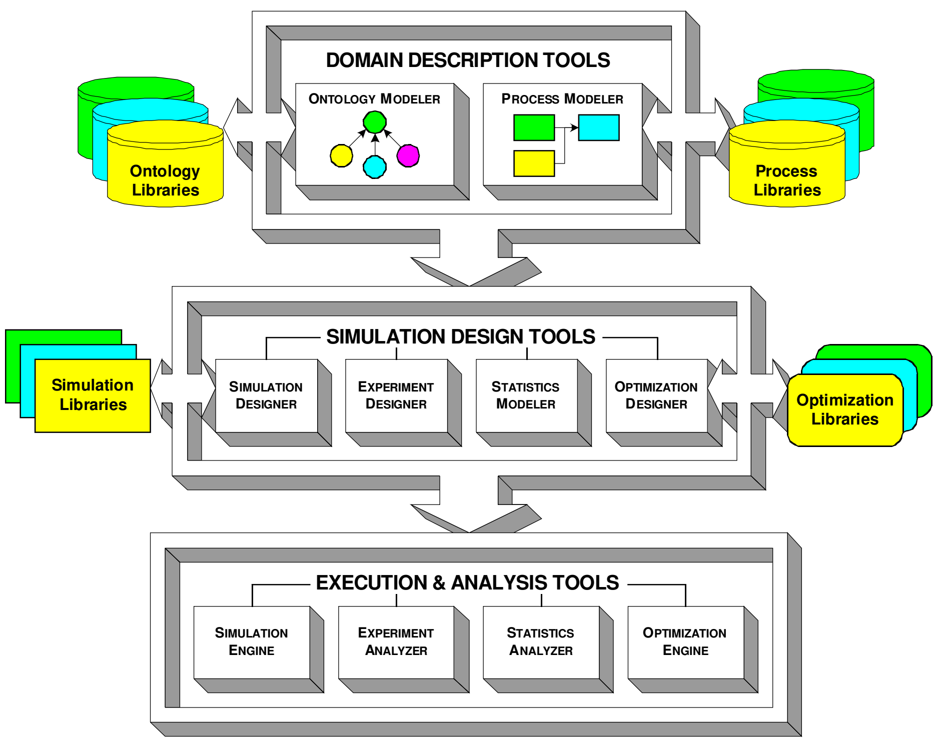

Figure 7. MODSIM Architecture

4.1 Domain Analysis Tools and Domain Libraries

The domain analysis tools and the domain libraries provide a mechanism to capture and re–use domain knowledge for simulation modeling. The use of domain models reduces the dependence on scarce and often expensive domain experts over the life cycle of the modeling effort. The domain modeling and analysis tools include (i) Ontology Modeler:

for the acquisition, and analysis of domain ontologies using the IDEF5 method; and (ii) Process Modeler: for the acquisition and analysis of domain process descriptions using the IDEF3 Method (see [www.idef.com]). The Ontology and Process Libraries are maintained to facilitate effective re–use.

4.2 Simulation Design Tools

Information from the domain analysis tools is transferred automatically to the simulation model tools using a set of translators. The Simulation Designer facilitates (a) the design of the conceptual simulation model using I3CML, (ii) the design of the detailed simulation model using the I3CML, and (iii) automatic generation of executable simulation code in different target simulation languages. The Experiment Designer facilitates (a) Strategic Experiment Design and (b) Tactical Experiment Design. The Statistics Modeler enables (i) simulation input data modeling (including data validation and data repair) and (ii) simulation output data analysis. The Optimization Modeler facilitates simulation–based optimization using Genetic Algorithms (GA) and Simulated Annealing (SA). The specifications of the GA and SA are automatically translated to executable optimization models that are processed by the Optimization Engine. Simulation based optimization is an iterative search process that involves the simulation modeler, the experiment designer, the simulation engine, and the optimization engine ().

Figure 8. Simulation–Based Optimization

4.3 Execution and Analysis Tools

We use the term Execution and Analysis Tools to refer to the collection of component–based tools that facilitate the execution of simulation experiments, collection and analysis of output data, and the generation of optimal solutions using simulation–based search methods. The execution and analysis tools therefore “run” the models, code, and data that are automatically generated by the Simulation Modeling Tools. The tools in this collection include (i) Simulation Engine, (ii) Experiment Analyzer, (iii) Output Analyzer, and (iv) Optimization Engine. Separating these components allows end users to mix and match different vendor components that best addresses the modeling objectives over extended periods of time.

Finally, we note that a subset of the architecture described in this section has been prototyped and is being currently used on several research and development projects.

Research Benefits of and Future Work Opportunities

5.1 Research Benefits

The benefits of the research described in this paper are summarized in the following

5.1.1 Reduced Simulation Lifecycle Costs

MODSIM technology will significantly reduce the time, effort, and cost required to develop, deploy, and maintain simulation models. This benefit will accrue through increased re–use of simulation life cycle information at the domain level and at the design level over extended periods of time. The model–based approach will enable future simulationists to rapidly deploy simulations starting from libraries of domain models and simulation models.

5.1.2 Enhanced Communication Between Domain Expert and Simulation Expert

The automated generation of executable analysis models from domain models will bridge the semantic gap between domain experts and simulation analysts. This enhanced flow of information application domain models and simulation models will increase the effectiveness of the communication required over the simulation development life cycle.

5.1.3 Simulation Agility

The capacity to generate simulation analysis software components from domain models and design models will allow end users to mix and match different simulation tools for any given application problem situation. MODSIM will enable end users to rapidly and cost effectively reconfigure the simulation tool architecture in response to constantly changing problem needs and requirements. We refer to this capability gain as enhanced simulation agility.

5.2 Future Work Opportunities

The following areas provide opportunities for future work in component based simulation (i) development of component based simulation reference architectures, (ii) development of domain libraries and simulation model libraries, and (iii) development of component analysis tools and simulation agents.

Acknowledgement

The research described in this paper was partly supported by NASA Small Business Innovation Research (SBIR) Contract Number NAS10–97002.

References

Delen, D., P. Benjamin, and M. Erraguntla, (1998). "Integrated Modeling and Analysis Generator Environment (IMAGE): A Decision Support Tool", in *Proceedings of the 1998 Winter Simulation Conference*, Washington, DC.

Erraguntla, M., P. C. Benjamin, R. J. Mayer, (1994). "An Architecture of a Knowledge–Based Simulation Engine." Winter Simulation Conference, pp. 673–680.

Law, A. M., W. D. Kelton, (1991). Simulation Modeling & Analysis. McGraw–Hill, Inc., New York, NY.

Pegden, C. D., R. E. Shannon, R. Sadowski, (1990). Introduction to Simulation Using SIMAN. McGraw–Hill, Inc., Hightstown, NJ.

Phillips, D. T., A. Ravindran, J. Solberg. (1976). Operations Research: Principles and Practice, John Wiley & Sons, Inc. New York.
Pritsker, A. A., (1986). Introduction to Simulation and SLAM II. John Wiley & Sons, Inc., New York.