Proceedings of the 1989 Winter Simulation Conference
E.A. MacNair, K.J. Musselman, P. Heidelberger (eds.)

THE SIMKIT™ SYSTEM: KNOWLEDGE-BASED SIMULATION AND MODELING TOOLS IN KEE®

Marilyn Stelzner
Jack Dynis
Fred Cummins
IntelliCorp, Inc.
1975 El Camino Real West
Mountain View, California 94040

ABSTRACT

The SimKit™ system is an integrated set of general-purpose simulation and modeling tools built in and on the Knowledge Engineering Environment® (KEE®) software development system. SimKit is used to build "knowledge-based" simulations. By knowledge-based, we mean an approach that utilizes a variety of techniques that have emerged from artificial intelligence including:

- The LISP programming environment
- High-resolution bitmap graphics
- Object-oriented programming
- Frame-based knowledge representation with inheritance
- Active values or "demons"
- Rule-based reasoning.

Knowledge-based simulations are more understandable, interactive, accessible, and extensible than simulations built using "conventional" techniques. In addition, knowledge-based simulations can easily be integrated with other knowledge system applications.

1. KNOWLEDGE-BASED SIMULATION TOOLS IN KEE

The SimKit system is an integrated set of general-purpose simulation and modeling tools built in and on the KEE software development system. KEE provides tools used in the construction of knowledge-based systems. Many of KEE's tools facilitate the modeling of domain knowledge, a goal that knowledge-based systems share with simulation. Because SimKit is built in and on KEE, it takes full advantage of the expressive representation, powerful reasoning, and user-friendly interface tools that KEE provides.

SimKit is explicitly designed for two classes of users: library developers and model builders. Library developers are programmers who use KEE and SimKit to build libraries, sets of domain-specific object classes and relationships. A factory library, for example, would include detailed descriptions of machines and the potential relations between those machines (e.g. downstream, on-top-of). Model builders, whoa re frequently not programmers, use a library and SimKit's interactive graphic interface to rapidly and easily build, modify and run models. SimKit puts sophisticated simulation and modeling tools in the hands of problem-solvers.

2. MODELING AND SIMULATION

A model is a description of a system that includes both the system's structure and its behavior. The structural model includes representations of the system's components and the relationships between those components, while the behavioral model describes how a system's state changes. The goal of simulation is to gain an understanding of how a system's behavior emerges from the behavior of its interrelated components.

Models and simulations are usually built to study system phenomena that are impractical, undesirable, or expensive to generate and study in the real world. For example, while it is of obvious value to understand the cost-benefit tradeoffs of alternative configurations of a manufacturing facility, it is prohibitively expensive to build, reconfigure and operate such a facility for this purpose.

Ideally, it would be possible to derive mathematical equations for calculating with certainty the values of interesting state variables as a function of time. Unfortunately, only the most simple (and, therefore, the least interesting) systems yield to this approach. Once a system consists of more than a few interacting components, it becomes difficult to predict system behavior with any accuracy or reliability. The goal of modeling and simulation is to create within the computer a representation of the system of interest that mirrors the real-world system in sufficient accuracy and detail to answer specific questions.

3. MODELING AND THE DEVELOPMENT OF KNOWLEDGE-BASED SYSTEMS

In IntelliCorp's® experience supporting users of the KEE system, we have noted that many routinely build structural models as a means to the end of developing applications that reason about those systems. Within the context of knowledge-based system development, models play roles both in knowledge acquisition and in system validation and extension.

In the case of knowledge acquisition, a structural model can be of use first in helping domain experts examine and isolate the system parameters and behavior that are relevant, and then in understanding how experts respond to and reason about such phenomena. The model thus serves as a medium for making expert knowledge explicit, well-organized, and accessible

4. MODELING AND SIMULATION IN A KNOWLEDGE-BASED ENVIRONMENT

When we speak of a knowledge-based approach to modeling and simulation we mean an approach that utilizes a variety of technologies and methodologies that have emerged from programming efforts directed at representing human knowledge ("artificial intelligence"). Those of greatest importance to our discussion include:

- LISP programming environment, a hardware and software environment optimized for symbolic, rather than numeric computation; this includes the extensive editing, tracing, and debugging facilities that have amassed around the LISP language.

- High-resolution bitmap graphics that support windows, mouse-and-menu operation, and the creation of graphic interfaces for developers and end-users.

- Object-oriented programming. Objects, which are autonomous and self-knowing (i.e. they contain the code that defines their behavior), can be replicated, modified, or deleted without modifying the model itself, a feature that results in tremendous flexibility and extensibility.

- Frame-based knowledge representation with inheritance. Frames, which can be thought of as extended data records, can be organized into hierarchies (e.g. a machine "template" can have several sub-classes, including a lathe "template;" that latter template can be further specialized into different types of lathes, etc. Inheritance helps developers describe complex systems quickly and consistently by starting with generic descriptions and specializing them.

- Active Values or demons, a mechanism for localized reasoning and analysis that is activated by a state change.

- Rule-based reasoning, a technique that allows heuristic knowledge, or "rules of thumb," to be represented as IF-THEN rules.

While all of these technologies and methodologies contribute to the effectiveness of the knowledge-based approach, it is the last four that most strongly differentiate it from conventional programming.

Modeling and simulation are concerned with the creation of a computational representation of a system of interest. The approach supported by conventional programming languages -- including nearly all of the specialized simulation languages -- is to create a procedural program that describes the system as a whole. The process of defining structural and behavioral models are very tightly coupled, and the descriptions of both kinds of models are intertwined in a program: definitions of structural elements of the system are intertwined with the procedural code. This non-modular and potentially inconsistent approach leads to programs that are difficult to modify and extend.

In knowledge-based (i.e. object-oriented) modeling and simulation, procedural components of the system are embedded within structural components, effectively reversing the relationship between procedure and structure that is characteristic of conventional approaches. While procedures are still a vital part of the knowledge-based model, the procedures are localized within discrete data structures that represent system objects and their attributes. The system as a whole is represented as a structured, modular, consistent collection of such objects -- a knowledge base -- that can easily be modified and extended.

5. CONVENTIONAL APPROACHES TO SIMULATION

The use of models to explore "what-if" scenarios has long been the goal of conventional simulation languages. We were, in fact, guided in our development of the SimKit product by a study of conventional computing approaches to simulation, and sought to overcome the limitations in these approaches that have hampered their widespread use and acceptance. Conventional simulations are frequently:

- Opaque. The "black box" syndrome -- the obscurity of the programming language makes it difficult for users to validate the accuracy of a model's output.

- Non-interactive. Modified models must be recompiled before effects of the modifications can be observed, resulting in lengthy turn-around time.

- Inaccessible. Only programmers experienced in the language can develop and modify models (other than changing certain paramaters).

- Inextensible. Eae of use is usually inversely proportional to ease of extensibility.

6. BENEFITS OF THE KNOWLEDGE-BASED APPROACH

Knowledge-based simulations are understandable, interactive, accessible, and extensible. In addition, they can easily be integrated with other knowledge system applications.

- Understandable. KEE's expressive, object-oriented knowledge representation makes it easy to describe objects and the relationships between them in a model. This is because the symbolic underpinnings of the knowledge-based approach allow the programmer to describe knowledge declaratively. A shovel, for example, can be described in familiar terms. Once described, it can be inspected using those terms (WEIGHT, COST, BLADE SHAPE, etc.). It may even be asked -- assuming it contains the appropriate function -- how many shovel-fulls it would take to fill a 3 x 3 x 3 hole.

  SimKit's tightly integrated graphics and representation make models visually inspectable at all stages of development. A model builder can see and manipulate icons and symbols that are natural and familiar. The interactions between the elements of the simulation can be seen, and hence understood.

- Interactive. The SimKit environment is wholly interactive. Modifications at the graphic level occur instantaneously at the deeper representation level. The end-user is never separated by a wall of "compilation time" from the most recent state of the system under study.

- Accessible. Conventional languages do not support the sort of inspection possible with KEE and SimKit. Typically, variables used to represent attribute values in conventional languages are declared at the beginning of a program and assigned values as the program executes. While the current value is cached in an array and is programmatically accessible, it is not readily inspectable as is a slot value in KEE.

  The inspectability of frames and slots in KEE and SimKit arises from their predictable structure. We know, for example, that every frame will have a name, lists of its parent and children, and a list of its slots. This predictable structure has been successfully exploited in constructing powerful graphic interfaces for creating, editing, and examining knowledge bases.

  Such general-purpose editors and utilities are not available in conventional simulation languages since the structure of a conventional program relies almost entirely on the discipline of its creator. As a result, as a program grows larger it becomes increasingly difficult to manage its complexity and understand its function. This has given rise to the familiar black box complaint that has hampered the acceptance of simulation.

- Extensible. SimKit libraries are designed as flexible templates. A library can be modified and extended to form one or more sub-libraries for more specific purposes (e.g. a manufacturing library can be specialized for printed circuit board production).

Once a system is described in KEE and SimKit, the free-standing model that results can be re-used for additional applications (e.g. a knowledge-based system for machine diagnostics).

7. A SYNERGISTIC COMBINATION

SimKit and KEE combine two classic approaches to problem solving: reasoning and simulation. The two techniques can be used in combination to attain results neither could attain alone. For example, a generative process planning knowledge-based system could be "tested" by a simulation, or, a knowledge-based system could be used to analyze simulation output. Alternatively, the two techniques could be intermingled, as in a factory simulation that reacts to random machine failures by invoking a knowledge-based system to recommend alternative part routings.

ACKNOWLEDGMENTS

INTELLICORP and KEE are registered trademarks of IntelliCorp, Inc. KNOWLEDGE ENGINEERING ENVIRONMENT and SIMKIT are trademarks of IntelliCorp, Inc.