# *Introduction to Discrete Event Simulation*

***By M. C. (Mike) Albrecht, P.E. (AZ)***



Table of Contents

Table of Figures

Table of Tables

# Introduction

In my studies in Industrial Engineering/Operation Research, I became very interested in Discrete Event simulation (actually I became interested in simulation back in my undergraduate work, but that is another story). I was preparing a research proposal on this area, when for several reasons I decide not to continue at this time (hence the Masters of Engineering). So that my work would not go to waste, I have created this site to make the work available to others.

What is Discrete Event Simulation (DES)? In classical thinking there are three types of simulation; discrete event, continuous, and MonteCarlo. In current thinking and work, these lines are becoming less distinct. DES (based on Nance (1993)) is a model (mathematical and logical) of a physical system that has changes at precise points in simulated time. Customers waiting for service, the management of parts inventory or military combat are typical types of DES. This site has it's emphasis on discrete event simulation (DES), but does include information on decision support systems (DSS) (particularly how DES can be used in DSS). This site will try to present an overview of DES as a tool. Particular emphasis will be on the use of DES in engineering decision support.

The following topics are covered in on this site:

- Selecting a Simulation and Modeling Language
- Evaluating a Simulation and Modeling Package
- Using Discrete Event Simulation in Decision Support
- What is Hybrid Simulation

I have also include a general Bibliography and a more focused Annotated bibliography that was used in preparing this site.

So again, "What is Discrete Event Simulation (DES)"? In classical thinking there are three types of simulation; discrete event, continuous, and MonteCarlo. They were articulated by Nance (1993) as:

*Discrete event simulation* utilizes a mathematical/logical model of a physical system that portrays state changes at precise points in simulated time. Both the nature of the state change and the time at which the change occurs mandate precise description. Customers waiting for service, the management of parts inventory or military combat are typical domains of discrete event simulation.

*Continuous simulation* uses equational models, often of physical systems, which do not portray precise time and state relationships that result in discontinuities. The objective of studies using such models do not require the explicit representation of state and time relationships. Examples of such systems are found in ecological modeling, ballistic reentry, or large scale economic models.

*Monte Carlo simulation*, the name given by John van Neumann and Stanislaw M. Ulam to reflect its gambling similarity, utilizes models of uncertainty where representation of time is unnecessary. The term originally attributed to "a situation in which a difficult non-probabilistic problem is solved through the invention of a stochastic process that satisfies the relations of the deterministic problem". A more recent characterization is that Monte Carlo is "the method of repetitive trials. Typical of Monte Carlo simulation is the approximation of a definite integral by circumscribing the region with a known geometric shape, then generating random points to estimate the area of the region through the proportion of points falling within the region boundaries.

In current thinking and work these lines are becoming less distinct, but for this site the main emphasis is Nance's discrete event simulation (DES). This site will try to present an overview of DES as a tool. Particular emphasis will be on the use of DES in engineering decision support.

## Selecting Simulation and Modeling Languages

In conducting a simulation modeling study one or more software tool is needed. There are many (87+) packages available for DES alone. Selecting the appropriate language or package for your study can be a study of its own. Tools for this selection are generally scarce.

## Overview

In simulation and modeling selecting the appropriate language or package for your needs can be difficult. Nance (1993) and (1995) identified the six characteristics of a discrete event simulation language. He proposed that discrete event simulation programming languages (SPL's) must meet a minimum of six requirements:

- generation of random numbers to represent uncertainty,
- process transformers, to permit other than uniform random varieties to be used,
- list processing capability, so that objects can be created, manipulated, and deleted,
- statistical analysis routines, to provide the descriptive summary of model behavior,
- report generation, to provide the presentation of potentially large reams of data in an effective way for decision making, and
- a timing executive or time flow mechanism.

Many software tools can and do meet these requirements. Finding them and evaluating them can be tedious.

## Simulation Software

A survey (Appendix A: Current (2007) Simulation Software) using several sources was performed to identify current simulation software. This survey used several sources:

- "Simulation Reloaded: Simulation Software Survey" by Swain (2005)
- Modeling & Simulation Resources website by Arsham

- A Collection of Modeling and Simulation Resources on the Internet website by Rizzoli
- The McLeod Institute of Simulation Sciences at California State University, Chico;
- Simulation Software for the Classroom, Simulation Education, Society for Computer Simulation International
- and a general Internet search

The survey identified 87 discrete event simulation and modeling tools (SMT's) which included SPL's and simulation application packages (SAP's) currently (or recently) available. Of these five (5) are not currently available (AWESIM, KAMELEON, MODSIM III, Silk, and Visual SLAM). Thirty-three (33) are either academia supported or open source supported. The remaining 49 are commercial offerings. Three (3) of the commercial offerings (Arena, Extend, and SIGMA) have academic versions used for teaching simulation courses.

The identified DES SMT's are listed in Appendix A: Current (2007) Simulation Software.

## Software Selection Frameworks

Selecting the appropriate simulation tool from these selections can be difficult. Several sources have proposed methods and or frameworks for evaluating these tools.

- Page, E. H., 1994, "Simulation Modeling Methodolgy Principles and Etiology of Decision Support," *PhD Dissertation Virginia Polytechnic Institute and State University*
- Hlupic, V., Z. Irani, and R. J. Paul, 1999, "Evaluation framework for simulation software," International Journal of Advanced Manufacturing Technology, 15(5), 366-382.
- Banks, J. and R. Gibson, 1997, "Selecting simulation software," IIE Solutions, May 1997, pp. 30-32

- Ahmed, R., T. Hall, and P. Wernick, 2003, "A Proposed Framework for Evaluating Software Process Simulation Models," *Proceedings Prosim'03* May 3-4 2003, Portland state University

## New Generation Modeling Framework (Page 1994)

Page's (1994) dissertation main purpose was to define the requirements for a next-generation modeling framework (NGMF). At the time of his research, computing technology was making great strides, and effecting simulation particularly in distributed interactive simulation and parallel discrete event simulation. This is also impacting on decision support systems. This lead to the two questions are addressed by this research:

- can the existing theories of modeling methodology contribute to these new types of simulation, and
- how, if at all, should directions of modeling methodological research be redefined to support the needs of advancing technology.

Page (1994) directed his work towards refining and strengthing the Conical Methodology/Condition Specification to better meet the NGMF that he developed. But in doing so he provided insight into what discrete event simulation is and the methodology of performing it. Page states that discrete event simulation at its most fundamental level is a tool for decision support. He also stated that discrete event simulation models are able to adopt many forms:

- A single, large, relatively static model that serves over a protracted period of use, e.g. a weather simulation.
- A single model which evolves rapidly during experimentation for system design or optimization, e.g. a cache model.
- A model which consists of a synthesis of results from several existing models in an effort to answer questions on a metasystem level.
- Models used for analysis.
- Models used to animate and visualize systems.
- Models used to provide an interactive training environment.

- Models used to stimulate hardware prior to operational deployment.
- Models used for real-time decision support.
- Models which provide various combinations of the above.

Page defines simulation as:

> "The use of a mathematical/logical model as an experimental vehicle to answer questions about a referent system." Page (1994)

and

> "arriving at the correct decision is the overriding objective of simulation." Page (1994)

Following the work Nance, Page has presented that for a DES the two concepts of *time* and *state* are of major importance. He uses the primitives defined by Nance for defining the relationship between time and state:

An **instant** is a value of system time at which the value of at least one attribute of an object can be altered.

An **interval** is the duration between two successive instants.

A **span** is the contiguous succession of one or more intervals.

The **state of an object** is the enumeration of all attribute values of that object at a particular instant.

Page further follows the work by Nance to present some widely used (and misused) simulation concepts:

An **activity** is the state of an object over an interval.

An **event** is a change in an object state, occurring at an instant, and initiates an activity precluded prior to that instant.A n event is said to be **determined** if the only condition on

event occurrence can be expressed strictly as a function of time. Otherwise, the event is **contingent.**

An **object activity** is the state of an object between two events describing successive state changes for that object.

A **process** is the succession of states of an object over a span (or the contiguous succession of one or more activities).

Page shows these concepts pictorially as:

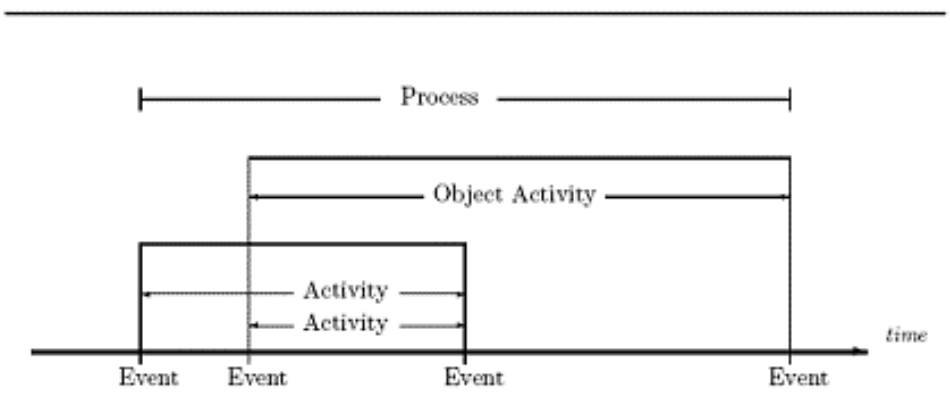



**Figure 1: Illustration of Event, Activity and Process. (Page (1994))**

Note that activities are bounded by two successive events. He then presents that event, activity and process form the basis of three primary **conceptual frameworks** (**world views**) within discrete event simulation.

In an **event scheduling** world view, the modeler identifies when actions are to occur in a model.

In an **activity scanning** world view, the modeler identifies why actions are to occur in a model.

In a **process interaction** world view, the modeler identifies the components of a model and describes the sequence of actions of each one.

From this it can be inferred that alternative worldviews are different perceptions of the same reality.

Page identified several factors that are positively correlated with the probability of making a correct decision:

An adequate understanding of the problem to be solved. If the problem to be solved is not well-defined and manageable, then little hope exists that a solution to the problem is readily forthcoming. (This is fundamental to every known problem-solving technique and certainly not unique to simulation.)

An error-free model. The correctness of the model is paramount to a cost-effective solution in light of the overall objective. Errors induced in the model, if never detected, could lead to the acceptance of results based on an invalid model – a potentially disastrous action. If an error is detected, but its detection comes late in the development stream, the cost of correction involves the cost of correcting the model and repeating the development steps. To be cost-effective, the methods for model development should foster the initial development of correct (error-free) models.

An error-free program. Recognizing that the program is but one representation of a model – usually the last in a line of development, a correct program can only be generated from a correct model.Th e arguments for program correctness mirror those for model correctness.

Experiment design. Construction of the model and program must reflect the objectives in carrying out the simulation; the right questions must be asked of the program in order that the appropriate answers can be derived. The problem understanding must be

sufficient and the model and program designed to facilitate the experiment design process.

Interpretation of results. A key recognition here is that no simulation program ever built produced the answer to anything.T ypically, simulation output measures are observations of random variables, and a proficiency in statistical methods, including variance reduction and multivariate analysis, is required to successfully – and correctly interpret the results provided by a simulation.

From this he develops his next-generation modeling framework (NGMF). Page bases his NGMF on a 1977 report to the National Bureau of Standards by Nance (1977) in which Nance identifies six (6) characteristics of a simulation model specification and documentation language (SMSDL) and a panel session held at the 1992 Winter Simulation Conference, entitled “Discrete Event Simulation Modeling: Directions for the ’90s,” where Sargent (1992) outlines fourteen (14) requirements for a modeling paradigm.

Nance's Characteristics are:

N1. The semantics of a SMSDL must facilitate model specification and model documentation.

N2. A SMSDL must permit the model description to range from a very high to a very low level.

N3. The degree of detail - the level of description - should be controllable within the SMSDL.

N4. A SMSDL must exhibit broad applicability to diverse problem areas.

N5. A SMSDL should be independent of, but not incompatible with, extant simulation programming languages.

N6. A SMSDL should facilitate the validation and verification of simulation models.

Sargent's requirements are:

S1. General purpose - to allow a wide variety of problem types and domains.

S2. Theoretical foundation - to move the modeling process towards science.

S3. Hierarchical capability - to facilitate the modeling of complex systems.

S4. Computer architecture independence - sequential, parallel and distributed implementations from same model.

S5. Structured - to guide user in model development.

S6. Model reuse - support a model database for component reuse.

S7. Separation of model and experimental frame - model should be separate from model input and model output.

S8. Visual modeling capabilities - to permit graphical model construction.

S9. Ease of modeling - world view(s) should ease the modeling task.

S10. Ease of communication - the conceptual model(s) should be easy to communicate to other parties.

S11. Ease of model validation - should support both conceptual and operational validity.

S12. Animation - model animation provided without burden to the modeler.

S13. Model development environment - to support modeling process.

S14. Efficient translation to executable form - model automatically converted to computer code, or if not automated, facilitate programmed model verification.

Page combined these into his NGMF.

**Table 1: Requirements for a Next-Generation Modeling Framework**

| No. | Requirement | Satisfies |
|---|---|---|
| 1 | Encourages and facilitates the production of model and study documentation, particularly with regard to definitions, assumptions and objectives. | N1,S7,S10 |
| 2 | Permits model description to range from very high to very low level. | N2,S10 |
| 3 | Permits model fidelity to range from very high to very low level. | N3,S3 |
| 4 | Conceptual framework is unobtrusive, and/or support is provided for multiple conceptual frameworks. | S8,S9 |
| 5 | Structures model development. Facilitates management of model description and fidelity levels and choice of conceptual framework. | N3,S2,S5,S8 |
| 6 | Exhibits broad applicability. | N4,S1 |
| 7 | Model representation is independent of implementing language and architecture. | N5,S4,S14 |
| 8 | Encourages automation and defines environment support. | S2,S13,S14 |
| 9 | Support provided for broad array of model verification and validation techniques. | N6,S11,S12 |
| 10 | Facilitates component management and experiment design. | S2,S3,S6,S7 |

Page then evaluates eight (8) DES methods for there level of support of his NGMF. In addition he evaluated the use of Conical Methodology in conjunction with the Condition Specification for use in defining DES models. He model development approach given by the Conical Methodology in conjunction with the Condition Specification as its model specification form.The methods are rated on a scale of 1 to 4 for their level of support where:

1 - Not Recognized;

2 - Recognized, but Not Demonstrated;

3 - Demonstrated;

4 - Conclusively Demonstrated.

The methods reviewed are:

CC - Change Calculus;

GST - General Systems Theory; 999

ACD - Activity Cycle Diagrams;

EvGr - Event-Oriented Graphs;

PN - Petri Nets;

LB - Logic-Based;

CFG - Control Flow Graphs;

GSMP - Generalized Semi-Markov Process.

CM/CS - Conical Methodology/Condition Specification

**Table 2: DES Methodology Evaluation**

| No. | CC | GST | ACD | EvGr | PN | LB | CFG | GSMP | CM/CS |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 2 | 3 | 2 | 1 | 1 | 2 | 1 | 3 |
| 2 | 2 | 2 | 3 | 3 | 1 | 1 | 2 | 1 | 2 |
| 3 | 1 | 2 | 2 | 3 | 3 | 2 | 2 | 1 | 3 |
| 4 | 2 | 1 | 1 | 2 | 1 | 1 | 2 | 1 | 2 |
| 5 | 2 | 3 | 2 | 2 | 2 | 1 | 2 | 1 | 3 |
| 6 | 2 | 4 | 3 | 2 | 3 | 2 | 2 | 3 | 3 |
| 7 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 8 | 2 | 3 | 3 | 3 | 3 | 2 | 2 | 1 | 3 |
| 9 | 2 | 3 | 2 | 3 | 3 | 2 | 2 | 1 | 4 |
| 10 | 1 | 4 | 2 | 3 | 1 | 1 | 2 | 1 | 3 |
| Average | 1.9 | 2.7 | 2.4 | 2.6 | 2.1 | 1.6 | 2.1 | 1.4 | 2.9 |

Pages evaluation had the CM/CS approach comparing favorably with the other approaches, but the CM/CS was not much better than GST or EvGR. It should also be noted that Page (working with Nance) was instrumental in developing the conical methodology with condition specification framework for DES. Also while he separates out Petri nets, event oriented graphs, activity cycle diagrams, and control flow graphs; they are very similar in style, look, and feel.

Pages comparisons are quantifiable, but are on a theoretical basis, they could be converted to a system for evaluating alternative simulation packages.

The following is a modification of Page's NGMF oriented towards SMT's:

The creation of a well documented and easy to understand model should be straightforward and uncomplicated. The modal components should be easy to determine and define.

- Model description should be possible at different levels of detail from the simple to very detailed. The resultant models should be easy for others to comprehend.
- Models of simple as well as very complex systems should be possible. The level of detail should be flexible and adjustable. The ability to build hierarchical models is desirable.
- The ability to model using different frameworks or worldviews is desired. The system should not require a particular framework/worldview.
- The ability to model by use of an easy to understand interface (graphical ("drag and drop") construction) is desired.
- The system should lead the developer into the model construction or development. System should allow management of model development and description. Model development should lead to proper framework.
- Must be applicable to a wide selection of problems. System should not be limited to a few types of models.
- Models should be independent of the programming language and underlying system architecture. Ability to convert to stand alone programs in a selection of languages (C/C++, Java, etc.).
- The model system should allow for fast model generation using predefined components with the ability to modify, add, and layer to achieve the desired level of detail. The resultant model should be easy to convert to a standalone program.
- System should facilitate model verification and validation. Internal checking and debugging is important.

- Addition of animation without extra work.
- System should support the creation of reusable components.
- The analysis of the resultant model and its results should be separate from the basic input and output of the model.

## Simulation checklist (Hlupic, Irani, and Paul 1999)

Hlupic, Irani, and Paul (1999) present a checklist to compare simulation software. Criteria are grouped by;

- General features
- Visual aspects
- Coding aspects
- Efficiency
- Modeling assistance
- Testability
- Software compatibility
- Input/output
- Experimental features
- Statistical facilities
- User support
- Financial and technical features
- Pedigree

The checklist is comprehensive, but it is somewhat repetitive, and definitely shows its age. An example of this deals with the interface. Interface related items pertaining to mouse support exist in Efficiency, Modeling assistance, and Input/output. Which were of interest in the mid 1990's, would now be covered by discussing if the system had a graphical interface or not. It also looks to have been generated from other software evaluation checklists.

While the topics and items covered by Hlupic, Irani, and Paul (1999) are similar to the concepts described by Page (1994) the evaluation is rather subjective, and it is difficult to quantify the relative merits of two separate SPL's.

The evaluation framework by Hlupic, Irani, and Paul (1999) is shown in Appendix B, below.

## Selection of simulation software (Banks and Gibson 1997)

Banks and Gibson (1997) present an overview of how to select simulation software. With the large number of DES packages available, the user needs to make a careful selection for most suitable. They provide guidelines to make general evaluation in five (5) areas: input processing, output, environment, vendor, and cost. They highlight three (3) items (warnings) that need to be considered:

- know which features are appropriate for your situation
- don't judge on the basis of yes and no (in overview check boxes)
- you may not need certain features

The items Banks and Gibson (1997) reference are:
Input Considerations

- Point and click capability - GUI interface
- CAD translations - import existing graphics
- File import - ability to import data
- File export - ability to export data for other software usages
- Syntax - simple syntax
- Interactive run controller - debugging features
- Interface to another language - convert to another programming language
- Input Data Analysis - ability to do statistical and distribution analysis
- Processing considerations
- Powerful constructs - complex models or components
- Speed - able to run complex models with no or little performance degradation
- Runtime flexibility - batch runs, scenario generations

- Random variate generator - generate standard distributions
- Reset - ability to reset statistics for steady state
- Independent replications - change the random number sequence (seed)
- Attributes and global values - need a large number of each available
- Programming - ability to change the level of detail or include special algorithms
- Portability - cross platform functionality
- Output Considerations
- Standardized reports - able to create standard outputs
- Customized reports - able to specify special outputs
- Business graphics - create graphical output
- Database maintenance - output to a file for latter analysis
- Collect mathematical expressions - add special algorithms
- Custom performance measures - define or create special measures
- Write to a file - output to a file for latter analysis
- Environment considerations
- Ease of use - power is probably more important
- Ease of learning - important to the casual learner
- Quality of documentation - on-line and easy to understand
- Animation capability - quality, smoothness, and portability
- Run only - ability to create a stand alone version
- Vendor considerations
- Stability - how long has vendor been in business
- History - annual updates, committed to improvement
- Track record - updates, on time and error free, backward compatibility
- Support - quality and level of support
- Cost considerations
- Productivity over price
- General
- Accuracy and detail
- Powerful capabilities
- Highest sped

- Substance over glitz
- Go beyond checklists
- Implementation and capability are important

Banks and Gibson provide a more user oriented framework then Ahmed, Hall and Wernick, but it again is mostly subjective.

## Simulation Evaluation Criteria (Ahmed, Hall and Wernick 2003)

Ahmed, Hall and Wernick (2003) describe the development and rational for a formal evaluation criteria for a simulation. They present the background for developing a formal set of criteria and summarizes previous work by others. They review work by Boehm (B. W. Boehm, "Software Engineering Economics", Prentice Hall Inc, 1981); Lindland, Sindre, and Solvberg(O. V. Lindland, G. Sindre and A. Solvberg, "Understanding Quality in Conceptual Modeling", IEEE Software, March 1994, pp.42-49); and Kitchenham, et al. (B. Kitchenham, L. Picakrd, S. Linkman and P. Jones, "A Framework for Evaluating Software Bidding Model", Proceedings of Conference on Empirical Assessment in Software Engineering, April 2002) in developing their proposed framework. Their work primarily builds on the work by Kitchenham, and proposes five main criteria:

- syntactic quality,
- semantic quality,
- pragmatic quality,
- test quality
- maintainability.

Ahmed, Hall and Wernick's work is a refinement over previous work by including enhancement of semantic quality and the addition of maintainability. They also consider the value of the simulation in their framework as a separate evaluation. While fairly detailed, the framework is not analytical and fairly subjective.

The evaluation framework from Ahmed, Hall and Wernick (2003) is shown in appendix C below.

## Conclusion

In conducting a simulation modeling study one or more software tools are needed. There are many (84+) packages available for discrete event simulation alone. Selecting the appropriate language or package can be a study of its own. Tools for this selection are generally scarce. Several sources have proposed methods and or frameworks for evaluating these tools.

Page (1994) in his dissertation on developing the next generation modeling framework presented a series of criteria for evaluating that framework. His criteria were quantifiable and could be adapted to simulation package evaluation.

Hlupic, Irani, and Paul (1999) presented an evaluation framework for simulation software that was fairly cumbersome and the analysis was not quantifiable.

Banks and Gibson (1997) also presented some qualitative features to consider. Ahmed, Hall, and Wernick (2003) provided more guidelines.

It was felt that a more user friendly approach could be developed. By combining the work of Ahmed, Hall, and Wernick (2003); Banks and Gibson (1997); Hlupic, Irani, and Paul (1999); and Page (1994) a Modeling Package Evaluation procedure was developed. This is developed further in the next section.

## Modeling Package Evaluation

With 87+ potential SMT's selecting the appropriate one can be difficult. Several sources have proposed methods and or frameworks for evaluating these tools. Some early work was by Banks and Gibson (1997) and by by Hlupic, Irani, and Paul (1999) in evaluating modeling packages. The more recent work by Ahmed, Hall, and Wernick's (2003), while fairly detailed, is not analytical and is fairly subjective. Banks and Gibson (1997) provide a more user oriented framework then Ahmed, Hall and Wernick, but it again is mostly subjective. The checklist by Hlupic, Irani, and Paul (1999) is comprehensive, but it is somewhat repetitive, and definitely shows its age. The checklist is close to 10 pages long and while the topics and items covered by Hlupic, Irani, and Paul (1999) are similar to the concepts described by Page (1994) the evaluation is rather subjective, and it is difficult to quantify the relative merits of two separate SPL's.

It was felt that a more user friendly approach could be developed. By combining the work of Ahmed, Hall, and Wernick (2003); Banks and Gibson (1997); Hlupic, Irani, and Paul (1999); and Page (1994) a Modeling Package Evaluation procedure was developed.

The following draws heavily on the work by Banks and Gibson (1997) and of Page's NGMF but is oriented towards SMT's. The procedure covers seven (7) major areas.

- Modeling Environment
- Model Documentation and Structure
- Verification & Validation
- Experimentation facilities
- Statistical facilities
- User support
- Financial and technical features

## Modeling Environment

The earliest SMT's were sets of predefined functions and models that could be manually pieced together to form a model. Two early efforts at a modeling environment were efforts at Virginia Tech and the London School of Economics (and later Brunel University).

Some early work was done at the London School of economics and presented by Balmer and Ray (1986) and later at Brunel by Paul (1992).. They describe Computer Aided Simulation modeling (CASM) for automating simulation. Project was an early work to form what is now called an Integrated Development Environment (IDE) for developing and preparing a simulation model. Key part was improving graphical components. They presented that a major problem in simulation use in operations research (as of 1986) was the lack of knowledge by users and mangers. They present that for simulation to beocme a wide spread tool it needs to be easy and inexspenive to use.

Balmer and Ray identify several areas of simulation that need further development.

- Model structure
- Interactive simulation program generators
- Simulation environment
- Programming languages and computers

For the model structure Balmer and Ray follow the work of others that the strucuter is often defined by the approach or worldview. They also present Nance's view that it may be due to time and state relationship. I believe it can also come from how the modeler looks at the problem. They hold that an interactive simulation program generator can increase speed of development over direct coding. And they also state that programming languages and computers need to be user friendly. This then lead to to the development of the Computer Aided Simulation Modeling (CASM).

The work at Virginia Tech was described by Balci and Nance (1992). they present an idealized framework for the development of a simulation IDE (they use the term simulation model development environment (SMDE)). They state that the purpose of the IDE is to:

- offer cost-effective, integrated and automated support of model development throughout the entire model life cycle;
- improve the model quality by effectively assisting in the quality assurance of the model;
- significantly increase the efficiency and productivity of the project team; and
- substantial decrease the model development time.

They define that their idealized IDE is to have four layers (starting at 0):

(0) Hardware and Operating System,
(1) IDE Kernel,
(2) Minimal IDE, and
(3) IDES.

The hardware and operating system layer, is the basic computer system needed to operate the IDE. Depending on the actual IDE this may be a very specific system, or totally platform independent.

The IDE kernel is the underlying software that does the modeling. This is the layer that the tool set of layer 2 (minimal IDE) communicates with and where the basic code that they use is found.

The minimal IDE is the tool set for creating the models and performing the simulation. These tools are in two (2) groups system supplied tools and software supplied tools. The first group, or system supplied, is part of the operating system and consists of the source code manager, and other general tools (such as an e-mail system and a text editor). The second group are those tools unique to the SMDE and would consist of a

Project Manager, Premodels Manager, Assistance Manager, Command Language Interpreter, Model Generator, Model Analyzer, Model Translator, and Model Verifier.

The Project and Premodels Managers handle the supervisory portion of the IDE and general performs the housekeeping functions such as:

(1) administer the storage and retrieval of items from the Project Database;
(2) keep a recorded history of the progress of the simulation modeling project
(3) trigger messages and reminders (especially about due dates); and
(4) respond to queries concerning project status.
(5) locate and reuse components of completed simulation studies (component database).

The Assistance Manager is the learning or tutorial tool to assist the user in becoming familiar with modeling using this IDE. They describe the assistance manager as having four components providing:

(1) information on how to use an IDE tool;
(2) a glossary of technical terms;
(3) introductory information about the IDE, and
(4) assistance for tool developers for integrating help information

The Command Language Interpreter is the language through which a user invokes an IDE tool. It is directly tied to the GUI and handles the exchange of information between the GUI and the other tools.

The Model Generator, or the simulation model specification and documentation generator, is the portion where the actual model is created. It assists in:

(1) creating a model specification;
(2) creating model documentation, and
(3) accommodating model qualification.

The model generator takes the information from the command language interpreter and prepares the model for use.

The Model Analyzer provides the initial verification step, and checks for errors in language or simple logic. The model analyzer may do simple corrections or provide alternative ways to solve the problems found.

The Model Translator coverts the created model into executable code either for runing under the IDE or as a separate stand alone program.

The Model Verifier is for formal verification. It may do systematic testing and checking of the model.

The IDES can include other tools or components beyond the minimal. It can include tools that support specific applications, or maybe needed for a specific project or a modeler finds useful. If there are no additional tools the minimal IDE is the entire package. This layer would also be the source of a GUI.

The framework presented by Balci and Nance (1992) was used for developing the “Computer-Aided Simulation Software Engineering Environment” and the “Simulation Support Environment" at Virginia Tech.

The current trend in modeling environments is to provide an integrated development environment (IDE). This can significantly automate the creation of the model, and reduces the need for extensive programming experience. The IDE’s can provide tools to support visual (drag and drop) modeling process, but still allow alternative model creation methods. This last is often by use of an integrated text editor. The IDE should be structured to guide user in model development.

A good modeling environment will allow varying degrees of detail or level of description, controllable within the model. It should also allow full accessibility to the underlying

model code, and provide good readability of that code. It should also allow model animation provided without burden to the modeler. If it can offer some virtual reality features that is even a bonus.

# Model Structure

One of the first steps in developing a simulation model is creating the specification for that model. Several terms are often used such as model formalism, ontology, and taxonomy. While taxonomies (in the form of worldviews ) have been around for many years, some recent work has been directed towards ontologies. In the past the use of a taxonomical basis has been the standard method of defining the simulation approach and for comparing alternative simulation approaches.

## Taxonomy, Ontology, and Formalism

The purpose of taxonomies, ontologies, and formalism is to allow the simulation developer to better describe the system and thus model it.

According to Merriam-Webster, taxonomy is the study of the general principles of scientific classification, and is especially the orderly classification of items according to their presumed natural relationships. And ontology is concerned with the nature and relations of being. It is a particular theory about the nature of being or the kinds of things that have existence.

In our case we would have, at the ontology level: DSS – a system to assist in deciding. And at the taxonomy we would have applications for DSS:

- Artificial Intelligence
- Complex (Adaptive) Systems
- Expert Systems
- Intelligent Enterprises
- Simulation

From a taxonomical view this would be:

Decision support Systems

Simulation
Continuos simulation

Monte Carlo simulation

Discrete event simulation

Worldviews

Resource/job driven views

Formalism is creating a model specification to define how to create the actual model. Bernard P. Zeigler in the 1970's developed a formal approach for building models, using a hierarchical and modular approach (lately integrated with object-oriented programming techniques) called the Discrete Event System Specification (DEVS) formalism. This method would allow the developer to build a Model Base, permitting easy reuse of models that has been validated.

## Taxonomy

The work by Balci (1988), Nance (1993), Overstreet (1982), Overstreet and Nance (2004), Schruben and Roeder (2003), and by Roeder (2004) have focused on the taxonomy of DES. Conventional simulation taxonomies have been considered either from a worldview perspective or resource or job-driven perspective (Roeder proposes also considering how information is handled). Some of this can be taken from how the simulation is approached and the purpose of the simulation. The majority of systems modeled could be from any one of the perspectives depending on what is being sought.

In comparing alternative simulation packages for DES there are two ways of describing the packages approach. The traditional method has been by referring to the worldview of software. A more recent method is based on how the software handles system entities.

## Worldviews

Balci (1988) presented four conceptual frameworks (later called world views by others) of DES

- Event scheduling,
- Activity scanning,
- Three-phase approach,
- Process interaction.

Subsequently Nance (1993) and others took the view articulated by Overstreet (1982) by using the concept of locality and defined three world views:

- Event scheduling: Event scheduling provides locality of time: each event routine describes related actions that may all occur in a single instant.
- Activity scanning: Activity scanning provides locality of state: each activity routine describes all actions that must occur because a particular model state is reached.
- Process interaction: Process interaction provides locality of object: each process routine describes the entire action sequence of a particular model object.

In examining the various worldviews, the method of implementation is different, which would lead to the belief that execution rates should be different.

## Comparison of Worldviews

Overstreet, and Nance (2004) used the concept of Action Cluster Interaction Graphs (ACIG) to show what is involved in transforming from one worldview to another. They used the classic repairman concept.

> "Informal Model Description: A single repairman services a group of n identical semiautomatic machines. Each machine requires periodic service based on a negative exponential random variable with parameter "meanUptime." The repairman starts in an idle location and, when one or more machines require service, the repairman travels to the closest failed machine. Service time for a machine follows a negative exponential distribution with parameter "mean-Repairtime." After servicing a machine, the repairman travels to the closest

machine needing service or to the idle location to await the next service request. The closest machine is determined by minimal travel time. Travel time between any two machines or between the idle location and a machine is determined by a function evaluation." Overstreet, and Nance (2004)

The ACIG for this would then be represented:

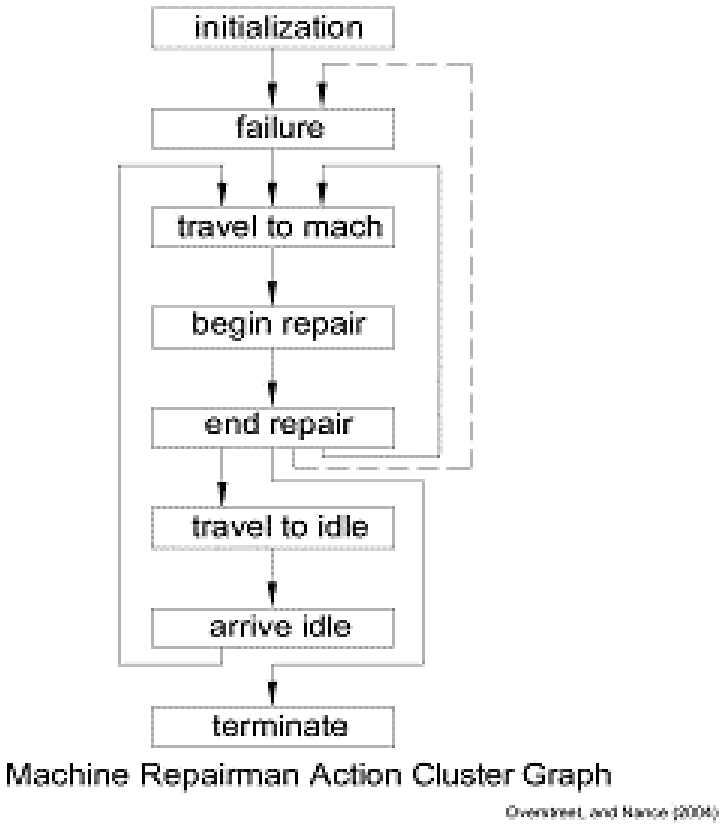



**Figure 2: Machine Repairman ACIG**

The ACIG for each of worldviews would then be represented:

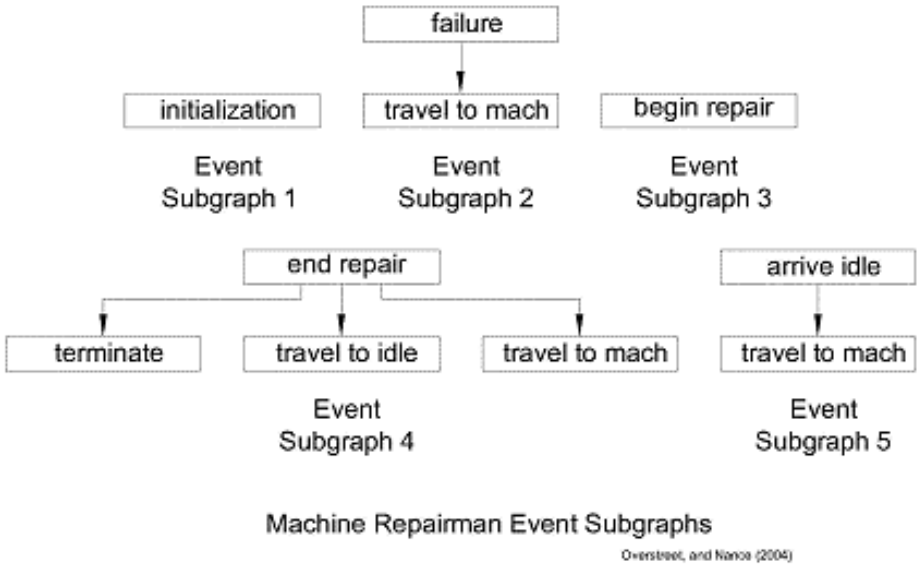



**Figure 3: Event Scheduling ACIG**

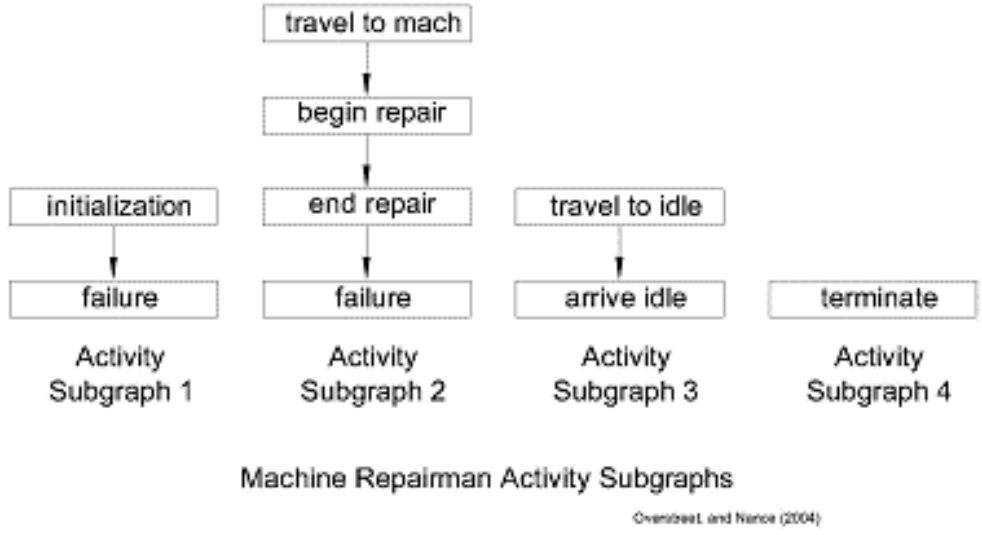



**Figure 4: Activity Scanning ACIG**

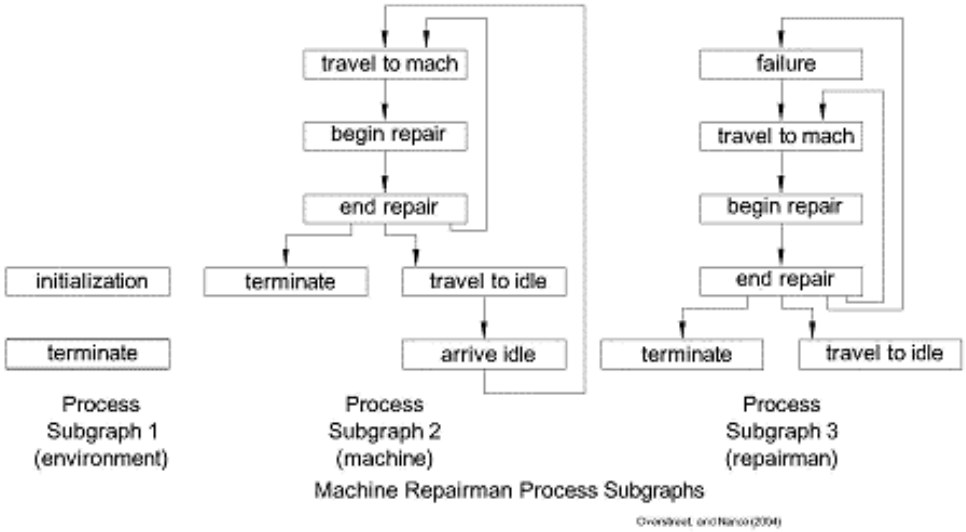



**Figure 5: Process Interaction ACIG**

> "Since each world view is based on a particular SPL that provided its own approach to time-advance, model implementations are usually closely tied to both a world view and the time-advance technique of the SPL used. These time-advance techniques can vary significantly in the run-time characteristics of executing models (depending on characteristics of the model). If a "world view independent" model specification could be created, then the choice of time-flow technique could be based on issues such as run-time efficiencies."
>
> Overstreet and Nance (2004)

Based on the ACIG's for the different worldviews, it maybe inferred that different run times might be expected.

## Resource-Driven versus Job-Driven

Recent work by Schruben and Roeder (2003) and by Roeder (2004) have proposed an alternative method based on how the software handles system entities (Resource-Driven versus Job-Driven Simulations).

**Table 3: Comparison of the job-driven and resource-driven paradigms**

| | Job-Driven | Resource-Driven |
|---|---|---|
| Disadvantages | • Large memory footprint for congested systems | • Insufficient information to a priori model certain system behaviors |
| | • Simulation execution slows as system size increases | • Available output limited |
| Advantages | • Detailed modeling possible | • Memory requirements insensitive to system congestion |
| | • Detailed output statistics available | • Simulation execution speed insensitive to system congestion |
| | • Easy animation | |

Roeder (2004)

While Resource-Driven and Job-Driven taxonomy's differ from the World views, they do have a relationship to each other.

**Table 4: Comparison of the implementation approaches**

| | Process Interaction | Activity Scanning | Event Scheduling |
|---|---|---|---|
| Job-Driven | Failure modeling inaccurate | N/A without many customizations | Schedule many events in congested systems (slows simulation) |
| | Deadlock possible | | |
| Resource-Driven | Must "trick" software into performing desired behavior | Model size increases dramatically with system size | |
| | | Many extensions required to enable modeling | |

Roeder (2004)

Both taxonomies can be used to evaluate simulation software. Based on the data supplied by Roeder, a comparison of different SPL's and how they relate to both taxonomies was prepared for this review. Sixteen simulation programming languages (SPL's) have been identified as being in common use. Of these fourteen SPL's have been identified as currently (early 2007) available. In addition two others were available

until recently (Silk and MODSIM III). Using the worldview and resource/job driven taxonomies, they breakout as follows.

**Table 5: SPL's by Taxonomy**

| | Software | Resource Driven | Job Driven | Activity Scan | Process Interaction | Event Scheduling | Other |
|---|---|---|---|---|---|---|---|
| 1 | Chi | **X** | | | **X** | | **X** |
| 2 | GPSS/H | **X** | | | **X** | | |
| 3 | GPSS/PC | **X** | | | **X** | | |
| 4 | JiST | | **X** | **X?** | | | |
| 5 | MODSIM III | | | | | | **X** |
| 6 | PASION Simulation System | **X** | | | **X** | **X** | |
| 7 | QNAP2 | **X** | | **X?** | **X?** | | |
| 8 | SIGMA | **X** | | | | **X** | |
| 9 | Silk | **X** | | | **X** | | |
| 10 | SIMAN | **X** | | | | **X** | |
| 11 | SIMPLE++ | | **X** | **X** | **X** | | |
| 12 | Simple 1 | | **X** | **X** | **X** | | |
| 13 | SimPy | **X** | | | **X** | | |
| 14 | SIMSCRIPT II.5 | **X** | | | | **X** | |
| 15 | Stoboscope | | **X** | **X** | | | |
| 16 | Visual SLAM | **X** | | | | **X** | |

There were also 39 simualtaion application packages (stand alone systems) (SAP's) identified as currently available. Note: there is some cross over, where the SPL can also be considered a SAP (such as GOSS/PC, Sigma, and Visual SLAM). Using the worldview and resource/job driven taxonomies, they breakout as follows.

**Table 6: SPL's by Taxonomy**

| | Software | Resource Driven | Job Driven | Activity Scan | Process Interaction | Event Scheduling | Other |
|---|---|---|---|---|---|---|---|
| 1 | ALPHA/Sim | | X | | | X | |
| 2 | AnyLogic 5.0 | | | | | | X |
| 3 | Arena | X | | | | X | |
| 4 | AutoMod | | X | X | | | X |
| 5 | AutoSched | | X | X | | | X |
| 6 | AWESIM | | | | | X | |
| 7 | CPN Tools | | X | X | X | | |
| 8 | CSIM19 | X | | | X | | |
| 9 | Design/CPN | | X | X | X | | |
| 10 | eM-Plant | | X | X | | | X |
| 11 | Extend Suite | X | | | | X | |
| 12 | Flexsim | X | | | X | | |
| 13 | GPSS/PC | X | | | X | | |
| 14 | GPSS World for Windows | X | | | X | | |
| 15 | HighMAST | X | | | X | | |
| 16 | MAST | X | | | X | | |
| 17 | Micro Saint | X | | | | X | |
| 18 | Modline | X | | X? | X? | | |
| 19 | POSES++ | | X | X | X | | |
| 20 | ProModel | X | | | X | | |
| 21 | Proplanner | | X | X | | | |
| 22 | QUEST | | | | | | |
| 23 | SansGUI | | X | X? | X? | | |
| 24 | ShowFlow | X | | | | X | |
| 25 | SIGMA | X | | | | X | |
| 26 | SIMAS II | X | | | X | | |
| 27 | SimCAD Pro | X | | | | X | |
| 28 | simjava | X | | | X | | |
| 29 | Simple 1 | | X | X | | | |
| 30 | SIMPLORER | | | | | X | |
| 31 | SIMPROCESS | X | | | | X | |
| 32 | SimPy | X | | | X | | |
| 33 | SIMUL8 | X | | | X | | |
| 34 | SLX | X | | | X | | |
| 35 | UMCYclone | | X | X | | | |
| 36 | Ventsim | X | | | X | | |
| 37 | VSE | | | | | | X |
| 38 | Visual SLAM | X | | | | X | |
| 39 | WebGPSS | X | | | X | | |

This shows a strong correlation between the two taxonomies and that similar SPL's and SAP's would be selected using either approach.

This leads to the question as to wether the taxonomy can have an impact on the model performance.

**Performance Optimization**

Developing a simulation model can be expensive, as well as running the model can take a large amount of computing resources. Obtaining the most information for this effort has been and will continue to be important. Work has been directed towards getting the most results from each run, to the tools used to examine the output, to work towards reducing the execution time.

Work by Schruben and others in the Berkeley Simulation Group (Schruben and Roeder (2003) and Roeder (2004)) have proposed that event graphs are inherently faster in execution. This was based on comparison of run times for two similar models for a wafer fabrication system. One model was prepared as a job driven view in Autosched AP (ASAP); the other model was an event graph system in Sigma. The Sigma model executed an order of magnitude faster than the ASAP model. (see below).

Unanswered questions are whether this was truly due to the worldview/taxonomy or due to the software implementation, and is this true for other models and other SPL's?

With the interest in running larger and more complex models that may require using a parallel discrete event simulation (PDES) approach. If there is an execution rate advantage to one world view, this could lead to improved performance.

Based on the ACIG's for the different worldviews (as decribed above in Model Structure), it maybe inferred that different run times might be expected.

The recent work by Schruben and Roeder (2003) and by Roeder (2004) have proposed that a significant difference in run times was found for two alternative models.

**Table 7: Simulation run lengths for the fab model**

| | Job-Driven | Resource-Driven |
|---|---|---|
| With PMs/Failures | ? 3 hours | < 10 minutes |
| | (? 352 seconds) | |
| Without PMs/Failures | ? 1.5 hours | < 10 minutes |
| | (? 300 seconds) | |

Schruben and Roeder (2003) and Roeder (2004)

How much of this was due to the different approach to the model, versus how much was due to SAP and system requirements was not determined.

**Improvement in Execution Rate**

Ways to improve on the execution rate have been discussed by Schruben and Roeder (2003), Roeder (2004), Soper (1984), and Page (1994).

Other than the work by Schruben and Roeder (2003) and Roeder (2004), and some anecdotal references (Sturgal personal correspondence (2005)), no formal evaluation of execution rate has been performed. Even the work by Schruben and Roeder was incidental to their actual work.

Soper (1984) identified that 40 to 80% of the execution time was related to event file management, but did not quantify this conclusion. His (Soper) work on an improved algorithm for event file maintenance, did not use actual SPL's but was confined to a stand alone Fortran program.

Page (1994) in his seminal work on the next generation formalism, alluded to execution rate as being important, but did not quantify it.

Several sources have described methods to improve the synchronization of PDES runs or to define when the run is over, the goal being to reduce the effective execution time (Alrefaei, and Alawneh (2004); Balakrishnan, et. al. (2001); Balsamo and. Manconi (1998); Bizarro, Silva, and Silva (1998); Colajanni, Dell'Arte and Ciciani (1998); Cowie (1998); Darcan and Kaylan (2000); Feldman, Muni and Swindle (1997); Hirata and Kramer (1997); Kartam and Flood (2000); Ki et. al. (1997); Peng and Chen (1996); Van Halderen and Overeinder (1998); and Zarei and Pidd (2001))

Balsamo and. Manconi (1998) specifically identified CPU overhead as concern in improving execution rate.

Hirata and Kramer (1997) addressed the use of one synchronization method (Time Warp) in PDES using alternative worldviews.

Much has been inferred about the execution rate of the various approaches, and work on specific aspects (such as file handling) have been reviewed, but the basic question of “is there a difference in execution rate between implementation methodology”, has not been answered.

Roeder (2004) in her dissertation deals with DES taxonomy and information usage in the DES model. In her dissertation, Roeder compares alternative DES simulation programming languages (SPL) by the way and method they handle information. It is her assertion that execution rate for DES SPL’s is effected by how much information the SPL’s handle. Including extra information beyond what is needed for the particular study being performed can slow down execution.

In addition she contrasts DES taxonomy’s based on the information handling of the languages (Resource-Driven and Job-Driven). In Resource-driven simulation the focus is on the resident entities in the model. For Job-driven simulation the focus is on the transient entities in the model. She also compares the information taxonomy with the classical World Views. The worldview approach describes SPL implementation but not

how information is handled. Her Taxonomy deals with information flow or how the SPL works and not how it represents the models. The purpose is to reduce excess information and thus memory usage. Roeders's dissertation did not provide specific examples showing this concept. If there is a significant difference, similar models that use different amounts of storage should have different run times.

To examine if there is a significant difference the following study using Sigma was performed for this review. Two models were used; CarwashA1 and Carwash B1.

The first model (CARWASHA1) is a standard queue model where customers enter a queue on a random basis. If an unoccupied server is available the next customer (FIFO) is served. CARWASHA1 is a system with up to five (5) servers (carwash lines) (three (3) used for this study) each having the same service time. For this simulation, the following state variables are defined: QUEUE: NUMBER OF CARS IN LINE (integer valued) and SERVERS: NUMBER OF AVAILABLE MACHINES (integer valued). Cars enter the system and are serviced by the next available server. Upon completion the cars leave the system.

The second model (CARWASHB1) is similar to the first accept each server can have a different serving rate. Again, if an unoccupied server is available the next customer (FIFO) is served. CARWASHB1 is also a system of up to five (5) servers (carwash lines) (three (3) used for this study) each having a variable service time (set similar to CARWASHA1 for this study. For this simulation, the following state variables are defined: QUEUE: THE NUMBER OF JOBS WAITING IN LINE (integer valued), SERVER[5]: NUMBER OF IDLE SERVERS OF EACH OF N TYPES (integer valued), N: NUMBER OF TYPES OF SERVERS (integer valued) and A: INDEX OF SERVER TYPE (A=AGE) (integer valued)

While similar in approach the second model has more variables, and stores more information. Graphical representations of the models are shown below.

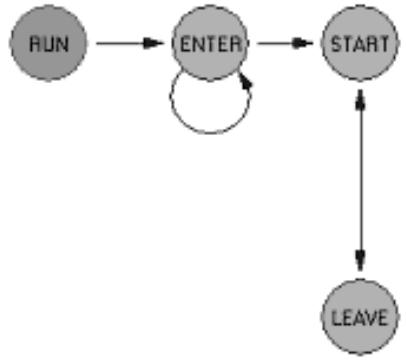



**Figure 6: Model 1(CARWASHA1):**

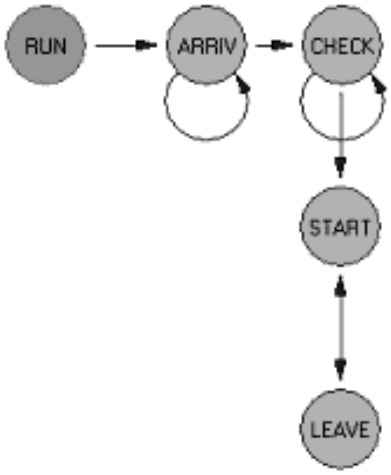



**Figure 7: Model 2 (CARWASHB1):**

In the second model the CHECK stage determines which server is used and then passes that servers time to the START – LEAVE step. For the first part of the study an early version of Sigma was used as later versions had the license lapse which deactivated the library. A version of Sigma issued in 1994 and included in Graphical Simulation Modeling and Analysis Using Sigma for Windows, by Schruben, L.W., 1995, Boyd and Fraser Publishing was used.

Since the results from the models were not of interest, the runs wee made as similar as possible. All servers had the same serving time. The translate function was used to convert each model into a C program which was then complied using Microsoft Visual C++™. The models were called CarwahsA1 and CarwashB1. The compiled programs

were both the same size 184kb. This was done using an early Sigma run time library to achieve equality between the programs.

A series of runs were performed using input files and executable bat files to call the programs. Input for the runs is shown below.

**Table 8: Model 1 (Carwash A1)**

| Output File | Over Write | Random No. Seed | Model Run Time | Initial Queue | Number of Servers |
|---|---|---|---|---|---|
| CW1-1.xls | y | 57128 | 100,000 | 1 | 1 |
| CW1-2.xls | y | 17517 | 100,000 | 2 | 2 |
| CW1-3.xls | y | 7347 | 100,000 | 3 | 3 |
| CW1-4.xls | y | 63795 | 100,000 | 4 | 3 |
| CW1-5.xls | y | 13593 | 100,000 | 5 | 3 |
| CW1-6.xls | y | 56482 | 100,000 | 6 | 3 |
| CW1-7.xls | y | 18338 | 100,000 | 7 | 3 |
| CW1-8.xls | y | 57528 | 100,000 | 8 | 3 |
| CW1-9.xls | y | 26518 | 100,000 | 9 | 3 |
| CW1-10.xls | y | 941 | 100,000 | 10 | 3 |

**Table 9: Model 2 (CarwashB1)**

| Output File | Over Write | Random No. Seed | Model Run Time | Initial Queue | Number of Servers | Time for Server[0] | Time for Server[1] | Time for Server[2] |
|---|---|---|---|---|---|---|---|---|
| CW2-1.xls | y | 57128 | 100,000 | 1 | 1 | 5 | 1 | 1 |
| CW2-2.xls | y | 17517 | 100,000 | 2 | 2 | 5 | 5 | 1 |
| CW2-3.xls | y | 7347 | 100,000 | 3 | 3 | 5 | 5 | 5 |
| CW2-4.xls | y | 63795 | 100,000 | 4 | 3 | 5 | 5 | 5 |
| CW2-5.xls | y | 13593 | 100,000 | 5 | 3 | 5 | 5 | 5 |
| CW2-6.xls | y | 56482 | 100,000 | 6 | 3 | 5 | 5 | 5 |
| CW2-7.xls | y | 18338 | 100,000 | 7 | 3 | 5 | 5 | 5 |
| CW2-8.xls | y | 57528 | 100,000 | 8 | 3 | 5 | 5 | 5 |
| CW2-9.xls | y | 26518 | 100,000 | 9 | 3 | 5 | 5 | 5 |
| CW2-10.xls | y | 941 | 100,000 | 10 | 3 | 5 | 5 | 5 |

The Model run time is the number of time units modeled for each run. In CarwashB1 Time for Server is the basic (minimum) time required for each service. The output was

directed to an Excel file for each run and the size of file and length of time for the run recorded.

**Table 10: Model 1 (Carwash A1): Results**

| Output File | Start Time | End Time | Run Time (Seconds) | Output file Size |
|---|---|---|---|---|
| CW1-1.xls | 3:02:11 | 3:02:14 | 3.47 | 1,409 |
| CW1-2.xls | 3:02:14 | 3:02:17 | 3.47 | 1,991 |
| CW1-3.xls | 3:02:17 | 3:02:21 | 4.63 | 2,342 |
| CW1-4.xls | 3:02:21 | 3:02:25 | 4.63 | 2,342 |
| CW1-5.xls | 3:02:25 | 3:02:28 | 3.47 | 2,354 |
| CW1-6.xls | 3:02:28 | 3:02:32 | 4.63 | 2,346 |
| CW1-7.xls | 3:02:32 | 3:02:35 | 3.47 | 2,346 |
| CW1-8.xls | 3:02:35 | 3:02:39 | 4.63 | 2,349 |
| CW1-9.xls | 3:02:39 | 3:02:43 | 4.63 | 2,341 |
| CW1-10.xls | 3:02:43 | 3:02:46 | 3.47 | 2,345 |
| Average | | | 4.05 | 2,217 |

**Table 11: Model 2 (CarwashB1) Results**

| Output File | Start Time | End Time | Run Time (Seconds) | Output file Size |
|---|---|---|---|---|
| CW2-1.xls | 2:38:05 | 2:38:40 | 40.51 | 24,200 |
| CW2-2.xls | 2:38:40 | 2:39:16 | 41.67 | 26,200 |
| CW2-3.xls | 2:39:16 | 2:39:50 | 39.35 | 26,134 |
| CW2-4.xls | 2:39:50 | 2:40:25 | 40.51 | 26,271 |
| CW2-5.xls | 2:40:25 | 2:41:00 | 40.51 | 26,068 |
| CW2-6.xls | 2:41:00 | 2:41:34 | 39.35 | 16,177 |
| CW2-7.xls | 2:41:34 | 2:42:08 | 39.35 | 26,057 |
| CW2-8.xls | 2:42:08 | 2:42:42 | 39.35 | 26,075 |
| CW2-9.xls | 2:42:42 | 2:43:17 | 40.51 | 26,150 |
| CW2-10.xls | 2:43:17 | 2:43:51 | 39.35 | 26,082 |
| Average | | | 40.05 | 24,941 |

Average run time for Model 1 (CarwashA1) was 4.05 seconds and average output file size of 2,217 kb. Average run time for Model 2 (CarwashB1) was 40.05 seconds and average output file size of 24,941 kb. A major part of the increase in file size was due to tracing each server in CarwashB1 and only the number of unused servers in CarwashA1. From this the increase in memory requirements did cause a very

significant increase in run time. This study did not evaluate what could be done in CarwashB1 to reduce this tim

Once you move beyond simple simulations, a large simulation can have aspects of several worldviews and resource and job driven attributes. It can also have continuous and discrete components as well as many stochastic elements. Describing this interrelationship would be the function of ontology as an ontology describes classes while taxonomy describes instances. Several people have been working to develop how an ontology based approach to simulation would differ from the current taxonomical based approach.

## Ontology

How would an ontology based approach to simulation differ from the current taxonomical based approach? There has been some work on ontology (Benjamin et al. (2006); Cuske, Dickopp and Seedorf (2005); Fishwick and Miller (2004); Gruninger, Schlenoff, Knutilla, and Ray (1997); Miller et al. (2004); and Silver, Lacy and Miller (2006)) for DES.

Benjamin, Patki, and Mayer (2006) have presented the use of ontologies for simulation and modeling. They identify four stages in the modeling process; Establish Purpose & Scope, Formulate Conceptual Model, Acquire and Analyze Data, and Design Detailed Model. In each of these four areas they identify ontologies can be used.

**Table 12: Use of Ontologies by Stage**

| Simulation Modeling Activity | Activity Description | Role of Ontologies |
|---|---|---|
| Establish Purpose & Scope | Capture needs, questions, objectives. Integrate across multiple perspectives. Map organization / mission goals to simulation goals. | Terminology harmonization to enable shared and clear understanding. |

| | | |
|---|---|---|
| Formulate Conceptual Model | Validate system descriptions. Identify model boundaries. Identify level of modeling abstraction. Identify model objects and roles. Determine model structure and logic. | Ontology knowledge is used to determine the unambiguously differentiated abstraction levels. Ontological analysis helps to map system objects to model objects and to identify appropriate object roles. Ontological analysis helps reason with system constraints to facilitate determination of model logic. |
| Acquire and Analyze Data | Identify data sources and data dictionaries. Perform data and text mining. Perform statistical analyses, data reduction (distribution fitting, etc.). | Ontologies play an important role in detailed data analysis, especially in disambiguating terminology and interpreting text data descriptions. |
| Design Detailed Model | Refine, detail, and validate model objects. Refine, detail, and validate model structure and model logic. | Ontologies will facilitate detailed analysis of objects and constraints including mapping the simulation model constraints to evidence / specifications of real world constraints in the domain descriptions. |

For any specific system the ontology describes classes while taxonomy describes instances. While an ontology can describe a higher level relationship with the formalism being the method used to describe that ontology the terms are often used interchangeably to differentiate from the taxonomy.

The work by Cuske, Dickopp and Seedorf (2005) on JOntoRisk (an ontology based system for risk management) is less about simulation and more about risk management. They use DES as part of their ontology instead of using an ontology for developing a DES model. They propose the use of an ontology based knowledge management system to create simulation models for risk management (banking). In their system the ontology is a layered approach to cover meta risk, domain risk, and others.

Working for the National Institute of Standards and Technology (NIST) on a Process Specification Language (PSL), Gruninger, Schlenoff, Knutilla, and Ray (1997) discuss the characteristics of a process language to be used for developing an ontology. They establish that a process has four (4) components; a core, an outer core, extensions, and application specific items.

- The Core items are those common to all processes, even the most simple, such as time resource and activity.
- The Outer Core items are those that common to many processes, such as time constraints, resource grouping, and alternative tasks.
- The Extension items are those that common to some but not all processes that increase the functionality of the process. They have identified six extensions:
    - administrative/business,
    - planning/analysis,
    - realtime/dynamic,
    - process intent,
    - aggregate processes/resources, and
    - stochastic/statistics.
- The Application Specific items are highly changeable, and only concern specific processes, such as dynamic rescheduling for a production scheduling application.

Gruninger, et al. then present an alternative to these by describing processes by groupings:

- grouping of requirements depending on their relationship to resources, tasks, time (the three basic attributes of a process), or some combination of them
- grouping of requirements as primitive concepts, their characteristics, and relationships
- grouping of requirements with respect to their level of granularity as a function of the manufacturing life cycle. Some requirements may only be necessary later in the manufacturing life cycle (when detailed information is required) while others may only be relevant earlier in the life cycle.

From this Gruninger, et al.then describe how an ontology could be constructed by setting an ontology as a set of axioms. They begin with foundational theories being how the ontology should describe the problem. Then the use the foundational theories to create specific theories for the problem which are the onotological building blocks. From here the propose creating a generic ontology.

Gruninger, et al. Conclude by stating that a process language could meet the requirements for the ontology. They do not provide a definitive answer, but refer on to other internal (NSIT) reports.

Major work on the development of ontologies for simulation and modeling has been done at the University of Florida and the University of Georgia. The work at Florida *by Fishwick) has been oriented around Rube ™ (named after Rube Goldberg), while the work at Georgia (by Miller) has been around Ontology Web Language (OWL) and in particular Discrete-event Modeling Ontology (DeMO). An overview of this work is presented in Fishwick and Miller (2004). This work is heavily oriented to Internet based systems. Fishwick and Miller (2004) build their case for DeMO and how it can help relate divergent work.

Miller, et al. (2004) present the development of Discrete-event Modeling Ontology (DeMO) which is based on the work towards an Ontology Web Language (OWL). They state “an ontology may be more than just a taxonomy with a simple hierarchic structure, but can capture more complex knowledge. It may indeed implicitly capture multiple taxonomies.”

Following description of the basic worldview taxonomies Miller, et al. then describe the structure and nature of DeMO. They describe DeMO in a programming language format. Their top order class (in programming) is called DeModel and is represented graphically as:

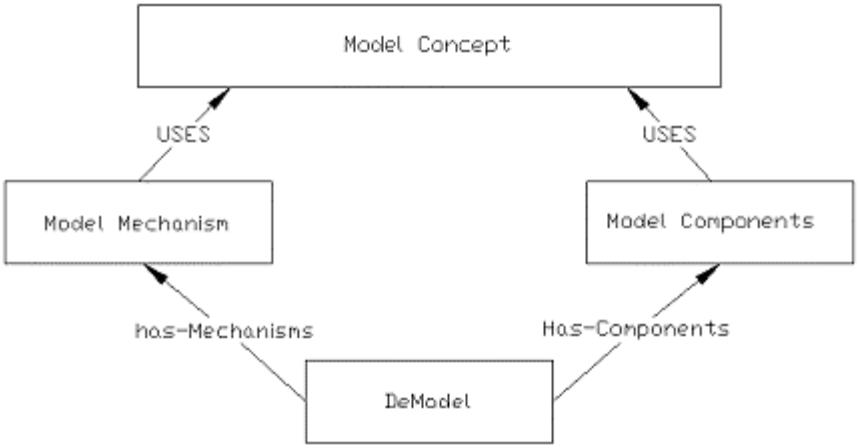



**Figure 8: Graphical representation of the rationale behind DeMO design**

The base of the DeMO system is set of Model Concept classes, which are the subsets of various worldview taxonomies. These work through the Model Component classes for common functions such as event sets, clock functions, and transition functions. For worldview specific items, the Model Mechanism classes contains the tools used by particular worldviews (e.g. process interaction) or programming systems (e.g. Petri Nets) such as event scheduling mechanisms, clock setting mechanisms, and similar.

Purpose of the work presented by Miller, et al. (2004) was to describe a potential ontology framework (DeMO) for web based simulation and modeling.

Silver, Lacy, and Miller (2006) describe further work on DeMO, but take a major turn from the previous work. While others (including Miller) have stated or implied that the Worldviews are at the taxonomy level, this paper (Silver, Lacy, and Miller (2006)) implies that worldviews are separate from the taxonomy/ontology concept (note: I disagree with this and prefer to think of worldviews as one of alternative taxonomy structures). The paper then presents an ”ontology” for the Process Interaction worldview, based on the use of DeMO.

## Model Specification (Formalism)

Formalism is creating a model specification to define how to create the actual model. Bernard P. Zeigler in the 1970's developed a formal approach for building models, using a hierarchical and modular approach (lately integrated with object-oriented programming techniques) called the DEVS formalism. This method would allow the developer to build a Model Base, permitting easy reuse of models that has been validated.

Radiva and Sargent (1987) present the generally accepted requirements for a DES. As part they describe the elements of a modeling formalism:

- Model Specification Language
- Semantics
- Model its Structure and Behavior
- Specification of Model Validity
- Proof System
- Modeling Methodology

Frantz (1995) describes going from a real world system, through a conceptual model to the simulation model as going to increasing levels of abstraction as you are going to simpler levels. A key object is to maintin a validity as you go to increase abstraction. His method is drawn from both simulation and modeling and from artifical intelligence. His work describes how the abstractions are developed based on three primary ways of modification in going from real to model, Under each he sub catagorieses by how you can achieve abstraction.;

- Model Boundary Modification
  - Explicit Assumptions
    - Hierarchies of Models
    - Delimit Input Space
  - Derived Abstractions
    - Approximation
    - Selection by Influences
- Modification of Behaviors
  - State Aggregation
    - Behavior Aggregation

        - Causal Decomposition
        - Aggregation of Cycles
        - Numeric Representation
    - Temporal Aggregation
    - Entity Aggregation
        - By Function
        - By Structure
    - Function
- Modification of Model Form
    - Look-Up Table
    - Probability Distribution
    - Linear Function Interpolation
    - Metamodeling

### Defining a Model Structure

The actual SPL that underlies the SMT environment will define how the model is structured. The SPL needs to provide varying levels of model description from a very high to a very low level. The SPL also should be operating system and computer architecture (sequential, parallel and distributed) independent. It should support the reuse of previous models and modeling components. Having a hierarchical capability to facilitate the modeling of complex systems is a benefit. This can include model chaining (i.e. linking outputs from different models). Ideally allowing varying worldviews is nice.

It is important that the model(s) are easy to communicate to others. Also that they can be efficiently translated to stand alone, executable form, and be compatibility with, extant simulation programming languages. There is significant overlap between model documentation and the topics related to verification and validation.

## Verification & Validation

Verification & Validation (V & V) establishs that the simulation model is good and correct. There is significant overlap between V & V and the topics related to model documentation.

Having a model and being able to have it generate results is only the start. Is the model truly reflective of the real world (or at least the portion being modeled) and are the

results true. Verifying the model developed and validating the results is critical in getting acceptance of the simulation. Journal articles by Naylor and Finger (1967); Gass (1983); and Cosset, Harrison, Winthrop, and Gass (1991) dealt with verification and validation of simulations. Having a model and being able to have it generate results is only the start. Is the model truly reflective of the real world (or at least the portion being modeled) and are the results accurate. Verifying the model developed and validating the results is critical in receiving acceptance of the simulation. The United States (US) Department of Defense (DoD) and the General Accounting Office (GAO) have driven much of the work in this area. Much of this was generated by the need to show that the US government was achieving a good return for the money invested. In this context verification deals with constructing the model correctly. Validation deals with results of the model and assurance that it functions properly.

Osman Balci (1998) defined verification and validation succinctly as:
"…. Model verification deals with building the model right. … Model validation deals with building the *right* model. …"

Naylor, T.H. and J. M. Finger, (1967) provides a definition of verification & validation as proving the model to be true, which requires (1) a set of criteria defining what is true from not true, and (2) that the user has the ability to apply the criteria to the model.

Gass's (1983) paper presents how operational research (OR) methodology, and in particular the use of computer modeling can be applied to policy analysis issues (or "squishy" problems). He presents the steps in applying OR to these types of problems and the concomitant use of detailed validation.

Cosset, C.A., D. Harrison, H. Winthrop, and S. I. Gass, (1991) presents the GAO's guidelines for assessing simulation models from a general perspective.

Naylor and Finger's (1967) work is one of the earliest discussions on formal computer simulation verification and validation. While they consistently refer only to verification,

what they describe is now generally referred to as verification and validation. The work has two major components, the first being defining the “why” of verification and contrasting alternative philosophies. The time that this was written (1966) and published (1967) was in the early formative stages of discrete event simulation (Nance (1995)). Simulation programming languages were just being developed, and computer usage was gaining ground. The need for formalized verification and validation was just being recognized. McKenney (1967) in his critique of Naylor and Fingers paper infers that computer simulation is done only when numerical modeling is too complex. Computers and computer simulations were new tools, and how to evaluate them was just becoming important to regular users.

Naylor and Finger define that for the purpose of industrial management systems, verification (“& validation”) is proving the model to be true, which requires (1) a set of criteria defining what is true from not true, and (2) that the user has the ability to apply the criteria to the model. They describe three existing methods: rationalism (not provable, but true by statement); empiricism (if it can not be proven true it is not true); and positive economics (ability to predict true results).

As an alternative, Naylor and Finger propose a combination of these or multi-stage verification. Where the model is checked for flaws and assumption errors, and the results are examined for accuracy of the results and how they match real world conditions. While they refer to it as verification, this would now be classed as verification and validation. They use sampling and analysis based on probability theory as the basis for their verification. Their three stage process is: (1) formulate hypothesizes, (2) verify (as much as possible) the postulates, and (3) test the ability to predict accurate results.

Also describes probability methods to do the third stage (test for goodness of fit). Their work is aimed at a statistical analysis of the model and it’s results. They recommend using several standard statistical tools to evaluate the results to include:

- Analysis of variance

- Chi-square Test
- Factor Analysis
- Kolmogorov-Smirnov Test
- Non-parametric Tests
- Regression analysis
- Spectral analysis
- Theil's Inequality Coefficient

Naylor and Finger (1967) do not give examples of the use of the statistical tools, but imply, as does Gass (1983), that standard operational research methods can be used. The procedures by Law and Carson (1979) and Schmeiser (1982) in the application of the Central Limit theory to data gives better confidence that the results can be analyzed.

Gass's (1983) paper presents a review of how operational research (OR) methodology, and in particular the use of computer modeling can be applied to policy analysis issues (or "squishy" problems). The application of OR methodology, and in particular the use of computer modeling to business and industry was well established by the early 1960's. At that point work began on how to apply the same OR techniques to policy analysis issues (or "squishy" problems). These problems are seldom clear-cut or well defined, as such the use of modeling has not always been successful. Proving the credibility of the model to the users and reviewers (who may have little training in OR) is difficult, as they often have a mental picture of how things work and if the model disagrees with that picture then they assume the model is wrong. To overcome this Gass presents that the use of verification and validation is important.

Gass stresses that for these types of problems (policy analysis) a well developed verification and validation procedure is very important. While mentioning the importance of verifying the model, he goes in to detail on the validation process separating out technical validation, operational validation, and dynamic validation.

Technical validation is after model verification and deals with analyzing the results in a formal manner. Technical validation deals with proving the model represents the real world, the data used is correct and accurate, and the logical and mathematical constructs realistically handle the data.

Operational validation deals with assessing the impact on the use of the model of errors found during technical validation (N.B. Gass assumes that errors will be found). Operational validation also deals with the perceived accuracy of the results and how they can vary. Gass proposes performing sensitivity analysis on the models to determine how they will vary as the model parameters change.

In dynamic validation he refers to the maintenance of the model in particular updating and reviewing. For this the model developer needs to provide for how to keep the model current as the conditions it models change during the model life.
Gass presents a four stage validation procedure originally presented by Emshoff and Sisson:

Developer checks that the model does what he wants it to, using real data if possible.
Check that the results produced are reasonable, both at the sub-model level and overall, and if possible the end user should also check it.
The end user should then become familiar with the model and results it produces.
As the model is used for decision analysis, records kept and the model updated as needed.

Following validation, comes assessment, which is an on-going process and which Gass recommends being performed by an outside party. He recommends that it be based on input from the developer (who should know the model best) and cover the following 13 items.

- Computer program documentation
- Model documentation
- Computer program consistency and accuracy

- Overall computer program verification
- Mathematical and logical description
- Technical validity
- Operational validity
- Dynamic validity
- Training
- Dissemination
- Usability
- Program efficiency
- Overall model validation.

The purpose behind this approach is providing the end user with confidence in the model, and credibility of the results. Through out this paper Gass reiterates importance of documentation. He states:

> "We do not know of any model assessment or modeling project review that indicated satisfaction with the available documentation. As documentation is the ***sine qua*** non of model validation and assessment, we cannot let the opportunity pass without noting that serious problems exist regarding the production and availability of model documentation."

Cosset, C.A., D. Harrison, H. Winthrop, and S. I. Gass, (1991) presents the GAO's guidelines for assessing simulation models from a general perspective. This paper is part of the set of papers presenting the GAO's work on VV&A. The authors present information showing that the US Government is a major user of computer models. They hold that the exact number of models is unknown, but estimated in the tens of thousands. The GAO estimated that in 1980 the DoD spent over $250 million on quantitative studies. Their position is that a portion of this expense is wasted on little used models. This is caused by not following a verification and validation procedure and by not providing complete documentation.

While the main thrust of this paper is DoD usage, and the examples presented are geared to weapon systems modeling, the basic premise and procedure is applicable to other models. As such their framework (shown below) has been adapted for general use

**Table 13: A Framework For Assessing The Credibility Of A Simulation**

| | Area of Concern | | Factor |
|---|---|---|---|
| a. | Theory, model design, and input data | 1 | Match between the theoretical approach of the simulation model and the questions posed. |
| | | 2 | Consideration of the modeled system's important operational measures of effectiveness |
| | | 3 | Portrayal of the immediate environment in which the modeled system will be used |
| | | 4 | Representation of the modeled systems operational performance |
| | | 5 | Depiction of the critical aspects of the broad-scale environment in which the modeled system will perform in the real world. |
| | | 6 | Appropriateness of the mathematical and logical representations of real world the modeled system will be in. |
| | | 7 | Selection of input data |
| b. | The correspondence between the model and the real world | 8 | Evidence of a verification effort |
| | | 9 | Evidence that the results are statistically representative |
| | | 10 | Evidence of sensitivity testing |
| | | 11 | Evidence of validation of results |
| c. | The support structure, documentation, and reporting | 12 | Establishment of support structures to manage the simulation's design, data, and operating requirements |
| | | 13 | Development of documentation to support the information needs of persons using the simulation results |
| | | 14 | Disclosure of the simulation's strengths and weaknesses when the results are reported. |

* references to weapon's system changed to modeled system

After Cosset, C.A., D. Harrison, H. Winthrop, and S. I. Gass, 1991, "An assessment procedure for simulation models: a case study,"

As models become more extensive and are being applied to more areas, assurance of their soundness and applicability become more important. Also the cost of the models becomes greater. Cosset, Harrison, Winthrop, and Gass (1991) strive to provide a more formal structure for evaluating models and for assuring their continued use and or reuse. Their framework (Table I above) describes how a V&V program should be structured, and they break the program into three main areas.

a. Theory, model design, and input data
b. The correspondence between the model and the real world
c. The support structure, documentation, and reporting

The earlier work by Naylor and Finger (1967) and the work by Gass (1983) deal with the second area of concern (b. The correspondence between the model and the real world. This is in providing a statistical analysis of the model and it's results.

From my personal experience, Cosset, Harrison, Winthrop, and Gass (1991) last area (c. the support structure, documentation, and reporting) is the single weakest area for any V&V effort. Getting any developer (whether it is a simulation model or any engineering package) to provide documentation and reporting is a major problem. It seems that most technical people take the attitude that if it works who needs to document it.

The SMT should have the tools that facilitate the verification and verification of simulation models. A well designed SMT will support both conceptual and operational validity. Use of an IDE can provides interactive model checking and better quality of error messages. If a trace file can be provided (showing events and entity status), this is another bonus.

## Experimentation Facilities

Once the model is created and verified, it needs to be used, often many times for varying conditions. Automatic batch runs, allowance for a warm-up period, and independent replications of experiments with re-initialization between runs without totally restarting the model are good features to have.

Other desirable features include, re-start from non-empty state, breakpoints, and on-line speed adjustment. If the SMT can provide experimental design capability with accuracy checking and automatic determination of run length they also are bonuses.

## Statistical Facilities

A well designed SMT needs to provide alternative statistical distributions, and even user-defined distributions to model varying conditions. Included with this is the ability to handle large volumes of random numbers, and if possible several random number streams. Each of the random number streams should allow for user-specified seeds or at least separate seeds.

To improve variance reduction the ability to handle antithetic sampling is desired, as well as distribution fitting. Standard experiment analysis functions such as goodness-of-fit tests, and output data analysis are advantageous. As well as quality of data analysis facilities and the capability to analyze confidence intervals.

### Statistical Analysis

Two of the articles, Law and Carson (1979) and Schmeiser (1982) describe methods to improve the confidence interval of the mean for simulation results. Both recommend the method of separating the run into groups or batches and then calculating the statistics for each batch. L'Ecuyer (1996) presents a method to prevent the recycling of random numbers and thus improve on achieving IID status.

Law and Carson (1979) describe the problem faced in stochastic simulation in constructing a confidence interval (CI) that will adequately cover the true mean. They present a method using a sequential procedure and batch means for a CI with the desired level of coverage. They report that when using a batch size of 20 or more (examples use a batch size of 40) the distribution of the batch averages approaches a normal distribution. They present a procedure of dividing the results from a simulation into batches of equal size. They then take the average of each batch and show that if the results are split into enough batches and the individual batches are large enough, the individual batch averages will be normally distributed, and effectively be IID. A part of their work is based on selecting a covariance to achieve the desired batch size. The smaller the covariance the greater the batch size.

Schmeiser (1982) continues the work of Law and Carson (1979) with emphasis on choosing the actual batch size. He uses batching to compute confidence intervals on the mean by transforming correlated observations into effectively uncorrelated and near normal distribution. Using the following assumptions

- Initial transient effects have been removed.
- For a run length of $n$, the existence of a number of batches $k^*$ and associated batch size $m^* = n/k^*$ such that dependency and the normality of the batch means are negligible, and
- The problem of $n/k^*$ not being an integer is insignificant.

Schmeiser (1982) further strengthens ii to be:

For a run length of $n$, there exists a number of batches $k^* >= 2$ and associated batch size $m^* = n/k^*$ such that for all $k <= k^*$, or equivalently for all $m >= m^*$, the dependency and the normality of the batch means are negligible.

Schmeiser (1982) reports that when using batching to reduce the number of data points, and to convert correlated data to semi-uncorrelated data with near normal distributions,

selecting the proper batch size is important. He reviews the theory of batch size selection to determine the appropriate size and the appropriate number of batches. Schmeiser examines the impact on the moments of the half width to the batch size (inversely proportional) and examines the impact of the coverage of the mean by the batch size (decreases as the number of batches increases, decreases as the value of alpha increases, is small when sigma is < 1).

Implications:

- the run must be long enough to provide the desired accuracy,
- the run must be long enough to calculate a valid confidence interval,
- smaller number of batches will often more easily satisfy normality and independence In general: 10<=k<=30 will satisfy most experiments.

In his title and write-up Schmeiser consistently refers to batch size *m where m= n/k*, *k* = number of batches, and *n* = total number of points. He actual uses *k* and infers *m*. His results are reported based on *k*. While they never call it that, the procedures used by both Law and Carson (1979) and Schmeiser (1982) are very similar to the Central Limit Theory and the Law of Large Numbers.

The Central Limit theorem (CLT) states that the distribution of the data under study does not have to be Normal because its average will be. The distribution of an average will tend to be Normal as the sample size increases, regardless of the distribution from which the average is taken except when the moments of the parent distribution do not exist.

Starting from a large enough population (what is large enough is intentionally left vague), a sample of size m is taken. The average of this sample is $\overline{Y}_n$. In a statistical context, the law of large numbers imply that the average of a random sample from a large population is likely to be close to the mean of the whole population.

If this were repeated many times to generate a series of samples $\overline{Y}_1, \overline{Y}_2, \overline{Y}_3, ..., \overline{Y}_k$, with *k* being large enough the distribution of these sample averages would approach a normal distribution. Thus in application the CLT is that if $\bar{y}$ is the mean of sample size n taken from a population with a mean $\mu$ and a standard deviation $\sigma$, then $\bar{y}$ itself is a random variable and:

The mean of $\bar{y}$ is $\mu$.

The standard deviation of $\bar{y}$ is $\sigma / \sqrt{n}$

If the population is normal, the distribution of $\bar{y}$ will be normal.

If the population is non-normal, the distribution of $\bar{y}$ will become close to normal as *n* gets large.

The closer the population is to normal the smaller the *n* required for the distribution of $\bar{y}$ being approximately normal.

This will not work for all results, only for results that have a statistically calculable mean. Some results are from distributions where this is not possible. The Cauchy is an example of a distribution with nonexistent moments. Thus the mean (the first statistical moment) doesn't exist. If the mean doesn't exist, then we might expect some difficulties with an *estimate* of the mean like $\overline{Y}_n$ and the CLT would not apply.

Law and Carson (1979) provides a very lengthy analysis and proof that the CLT and LLN holds for simulation results. While they infer that greater number of individual batches improves the results, they use an arbitrary batch size of 40 (note Schmeiser (1982) proposes that this method uses a batch size that is too large). If the original sample is large (Theory of large Numbers) then batch sizes of 30 or greater will approach a normal distribution. Both papers derive and prove the Central limit theory and the Law of Large Numbers.

An observation from reviewing these papers is that, as with any analysis, just going out and running your tests can lead to massive amounts of data that are difficult to analyze.

You need to consider, what is the optimal batch size, and what is the optimal number of batches. Part of this depends on the method of selecting the individual elements. If sampling without replacement is done then at least 30 batches of 30 items each are probably required, or at least 900 discrete points (more if initialization bias is present. But, if sampling with replacement is used, significantly less discrete points are needed (perhaps as few as 30). This leads to the higher level topic of "Design of Experiments." This is also the main premise in the Roeder dissertation, that controlling the amount of data can positively effect your simulation performance.

As a check of this theory a simulation was run using Sigma95 of a Car wash. The time in the queue for service was analyzed. The simulation was run for 5000 time units and the results generated 902 observations. From this set 30 batches of 30 samples each were taken. The 30 samples in each batch were taken without replacement; each batch was drawn with replacement. The results are shown in the table below:.

**Table 14: CLT check using Carwash Model**

| | Sample | Batches |
|---|---|---|
| Number | 905 | 30 |
| Mean | 4.002 | 3.902 |
| StDev | 5.136 | 0.931 |
| Variance | 26.375 | 0.866 |
| 95% CI | 0.383 | 0.381 |
| Min | 3.619 | 3.521 |
| Max | 4.384 | 4.283 |

The statistics for the total sample were calculated, as well as for each batch. The total sample is shown in first figure below. The batch means were combined and the statistics calculated for them, and shown in the second figure. The distribution for the original sample was skewed to the left. The distribution for the batched results appeared to be near-normally distributed.

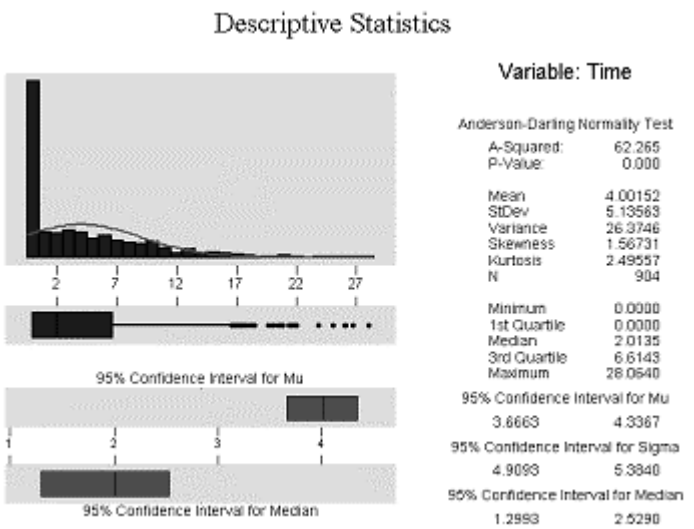



**Figure 9: Time in Queue**

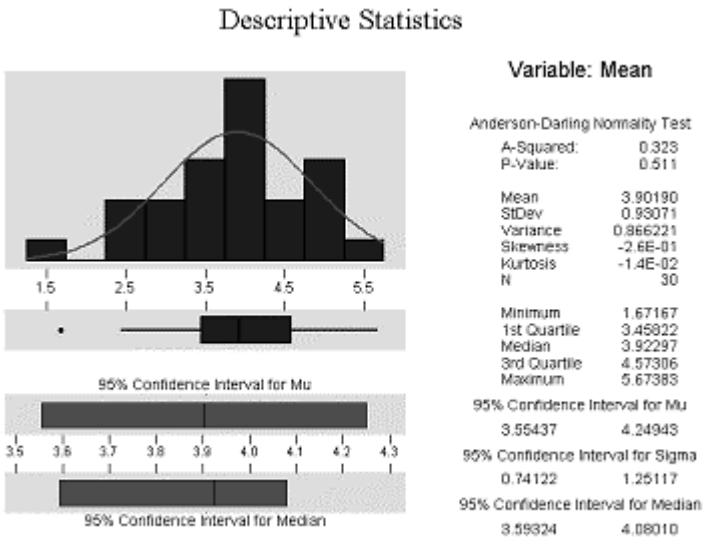



**Figure 10: Batched Sample Results Time in Queue**

In dealing with large amounts of data the method of batching can reduce the volume data handled without significant loss of information. The resultant confidence interval can be tighter.

## Random Number Generation

L'Ecuyer (1996) presents a method for generating random numbers that can overcome an inherent problem with standard generators, the recycling of numbers. Random number generators are actually psuedo-random numbers since, if they are used long enough the sequences will repeat. With larger and faster computers running larger simulations standard random number generators can exhaust their period length in a short period of time (as quick as 30 minutes). This can cause statistical problems when striving for IID status.

he standard generators used in most simulation packages and computers are linear congruential generators (LCG). A linear congruential generator (LCG) produces a series of numbers between 0 and m-1 based on

$$X_i = (aX_{i-1} + b) \bmod m$$
$$X_0 = seed$$
$$a = const$$
$$b = increment$$
$$m = \bmod$$

To achieve a sequence of 0 to 1: $R_i = X_i / m$

The following is a trivial example to generate random numbers using linear congruential method for $X_0$=27, a=17, c=43, and m=100

**Table 15: LCG periods**

| | X | a | B | m | R |
|---|---|---|---|---|---|
| 0 | 27 | 17 | 43 | 100 | |
| 1 | 2 | 17 | 43 | 100 | 0.02 |
| 2 | 77 | 17 | 43 | 100 | 0.77 |
| 3 | 52 | 17 | 43 | 100 | 0.52 |
| 4 | 27 | 17 | 43 | 100 | 0.27 |
| 5 | 2 | 17 | 43 | 100 | 0.02 |
| 6 | 77 | 17 | 43 | 100 | 0.77 |

Interestingly the period for this LCG is 4 unique numbers. If instead m had been a function of 2 (i.e. $2^7 = 128$) you would generate m-1 psuedo-random numbers (m= 128 repeats at the 128th number).

In practice LCG's take m to be at least $2^{31}$-1=2,147,483,647 (about 2.1 billion) and the other parameters are carefully chosen to achieve full or nearly full cycle length. Still in normal operations a large simulation can exhaust all these random numbers in a matter of minutes. By combining these (using two or more LCG's with different moduli and then adding them together) extended period lengths are possible. Period lengths grow to about $2^{60}$. This can still be exhausted when simulation runs of several hours are required.

L'Ecuyer (1996) proposes using combined multiple recursive generators (MRG), which are based on linear generators with high period lengths ($2^{200+}$). He proposes that MRG's of this length can overcome the problem with LCG's, but at a price. Combined MRG's can take twice as long as combined LCG's to generate the number.

The MRG is based on a recursive process.

$$X_i = (a_1 X_{i-1} + a_2 X_{i-2} ... + a_k X_{i-k}) \bmod m$$
$$X_0 = seed$$
$$a_n = const's$$
$$m = \bmod$$

A multiply recursive generator could be:

$$X_i = (a_1 X_{i-1} + a_2 X_{i-2} ... + a_k X_{i-k}) \bmod m_1$$
$$Y_i = (b_1 Y_{i-1} + b_2 Y_{i-2} ... + b_k Y_{i-k}) \bmod m_2$$
$$X_0, Y_0 = seeds$$
$$a_n, b_n = const's$$
$$m_1, m_2 = \bmod's$$

To achieve a sequence of 0 to 1:

$$P_i = (X_i - Y_i) \bmod m_1$$

$$R_i = \begin{cases} P_i/(m_1+1) : if P_i > 0 \\ m_1/(m_1+1) : if P_i \le 0 \end{cases}$$

For this MRG the cycle length is about: $3.1 \times 10^{57}$ which would take years to exhaust.

Larger and more complex models require more random numbers. Having a generator that starts recycling can lead to statistical problems when striving for IID status. While L'Ecuyer (1996) infers that the MRG will increase simulation run length, with faster machine clock speeds this should be less of a problem then recycling the numbers.

## User support

Unless the user develops their own SMT, user support tools are important. The level of documentation provided as well as the quality of documentation is important. Good technical and promotional information (e-mail, internet discussion groups) can assist in learning the SMT, as can training course (basic, advanced), tutorials, and demonstration models. If the SMT is meant for educational use, the provision of a lecturer's guide is important. General support functions are also critical such as providing package maintenance.

## Financial and technical features

The last part of a SMT evaluation should consider the cost of the package, how easy it is to install, what it will cost to maintain the package, as well as what hardware/system requirements it has.

## Conclusion

Following the above criteria a Modeling Package Evaluation procedure was developed. The procedure covers seven (7) major areas.

- Modeling Environment
- Model Documentation and Structure

- Verification & Validation
- Experimentation facilities
- Statistical facilities
- User support
- Financial and technical features

The procedure is described below.

## Modeling Package Evaluation Checklist

Taking the above factors, a quantitative method of comparing alternative SMT's is shown in Appendix D (Model Evaluation Package). Alternative SMT packages can be compared and ranked. If desired, additional weighting can be given to particular items. To show use of this method, four SMT packages are evaluated.

- Arena
- Extend
- Sigma
- Ptolemy II

## Arena

Arena is by Rockwell Software (http://www.arenasimulation.com/) that is based on the older SPL SIMAN. It is an integrated package with an academic version often used in simulation course when it is bundled with Simulation with Arena (Kelton, Sadowski, and Sturrock, 2004).

The Arena modeling environment is an IDE that is specific to it self. The IDE allows the use of visual (drag and drop) modeling using predefined and user defined blocks. These elements can be combined into higher level components to provide a single level hierarchy. There are minimal alternatives in model creation beyond the supplied IDE. Some ability to increase or decrease the level is provided. The IDE has a text editor as part of the report module.

The basic code is not easily accessible, or readable. Arena is structured to aid in model development and has some animation ability. The Arena IDE allows a varying level of model description. Arena is a Windows (tm) system designed for a single compute. Model components can be reused from a library, with a two level hierarchy. Minimal chaining of other models is possible. Arena is based on Siman, an early SPL based on the event scheduling worldview. Arena is fairly easy to learn, and run time models can be created for sharing..

Arena has a debugging and verification mode, and tools to allow validation of results. Models can be run in step wise and fast modes. Error messages are helpful.
One area that Arena is good in is the tools for experimentation. In setting up your model choices allow for warm up periods, independent replications, reiniitialzation, breakpoints, speed adjustment, checking accuracy, and determining run length. The ability to do automatic batch runs could be better. Arena provides the basic statistical tools, such as standard distributions, seeding random numbers, sampling, and distribution fitting. The ability to perform results analysis is also provided suing built in tools. Alternative distribution functions and alternate random number streams are difficult to implement.

Arena is often distributed as part of course textbook (Kelton, Sadowski, and Sturrock, 2004) providing excellent basic documentation. Other information is available from literature reviews and Internet searches. The academic use of Arena is cultivated by Rockwell (supplier). Advanced training, portability and conversion to other systems are weak. Having Arena distributed with a textbook reduces the total cost for the basic package, but commercial versions are significantly more expensive. Being Windows (tm) based samples the hardware requirements, but updates are not often, and can be expensive.

## Extend

Extend is a DES SMT by Imagine That, Inc. (http://www.imaginethatinc.com/) that is an extensible program with add-ons from several sources. It can be used to model continuous, discrete event or discrete rate processes.

As with Arena, the Extend modeling environment is an IDE that is specific to it self. The IDE allows the use of visual (drag and drop) modeling using predefined and user defined blocks. These elements can be combined into higher level components to provide a single level hierarchy. There are minimal alternatives in model creation beyond the supplied IDE. Some ability to increase or decrease the level is provided. The IDE has a limited text editor. The basic code is not easily accessible, or readable. Extend is structured to aid in model development and has some animation ability. Extend provides for a varying level of model description. Extend is a Windows (tm) system designed for a single computer, but with some parallel processing and distributed processing capability. Model components can be reused from a library, with a two level hierarchy. Minimal chaining of other models is possible. Extend (with external packages) can provide alternative worldview approaches. Extend is fairly easy to learn, and run time models can be created for sharing.

Extend has a debugging and verification mode, and tools to allow validation of results. Models can be run in step wise and fast modes. Error messages are helpful.

Extend has good experimentation tools. In setting up your model choices allow for warm up periods, independent replications, reiniitialzation, breakpoints, speed adjustment, checking accuracy, and determining run length. The ability to do automatic batch runs is weak. Extend provides the basic statistical tools, such as standard distributions, seeding random numbers, sampling, and distribution fitting. The ability to perform results analysis is also provided suing built in tools. Alternative distribution functions and alternate random number streams are difficult to implement.

Extend is primarily a commercial tool with some academic support. General support is adequate, particularly if you work with one of the Extend add-on sources. Advanced training, portability and conversion to other systems are weak. Extend's academic version is reasonably priced, but if more power is needed the price goes up quickly. Being Windows (tm) based simplifies the hardware requirements, but updates are not often, and can be expensive.

## Sigma

SIGMA is a DES SMT by Custom Simulations (http://www.customsimulations.com/) based on event graphs. It is also used in many simulation courses as it is relatively easy to learn.

Sigma also has an IDE that is specific to it self. The IDE allows the use of visual (drag and drop) modeling using predefined blocks. The user cannot create blocks or combine them into higher level components. Sigma models can be created and edited using an internal text editor, or any standard text editor. The basic code is easily accessible and the IDE has the ability to translate the model into more readable version. Sigma is structured to aid in model development and has some animation ability. Very little variation in the level of model description is provided. Sigma is a Windows (tm) system designed for a single compute. Model components can be reused from a library. Sigma is based on event graphs. Sigma is fairly easy to learn, and C based executable program for distribution can be created.

Sigma has a debugging and verification mode, and tools to allow validation of results. Models can be run in step wise and fast modes. Error messages are helpful.

Sigma provides some tools for experimentation. The ability to do automatic batch runs by creating separately executable programs that read input files is very good. Sigma provides the basic statistical tools, such as standard distributions, seeding random numbers, sampling, and distribution fitting. The ability to perform results analysis is also

provided suing built in tools. Alternative distribution functions and alternate random number streams are not available.

Sigma is often used as an introductory simulation teaching tool and is available with a course textbook (Graphical Simulation Modeling and Analysis Using Sigma for Windows (Schruben, 1995)) providing excellent basic documentation. Very little other information is available. Advanced training, portability and conversion to other systems are weak. Commercial versions are significantly more expensive. Being Windows (tm) based samples the hardware requirements, but updates are not often, and can be expensive.

## Ptolemy II

Ptolemy II is a DES/Continuous SMT developed by the Center for Hybrid and Embedded Software Systems (CHESS), at the Department of Electrical Engineering and Computer Sciences, University of California at Berkeley (http://ptolemy.eecs.berkeley.edu/). Ptolemy II is a software framework developed as part of the Ptolemy Project. It is a Java-based component assembly framework with a graphical user interface called Vergil. Vergil itself is a component assembly defined in Ptolemy II.

The Ptolemy II modeling environment is a general purpose IDE called Virgil that draws heavily upon the Eclipse project. The IDE allows the use of visual (drag and drop) modeling using predefined and user defined blocks. These elements can be combined into higher level components to provide a multi-level hierarchy. Models can be created from assembling previously developed components, are by direct programming (in Java). There is almost no limit on the hierarchical levels used. The IDE has a text editor as part of the report module. The basic code is in Java and is easily accessible and fairly readable. Ptolemy II is structured to air modeling and has some animation ability. Model description and documentation ability were part of the basic concept behind Ptolemy II. Being a Java based system; Ptolemy II is highly portable and can be structured in a parallel or distributed network. A key component of Ptolemy II is being able to create user libraries for subsequent reuse. Ptolemy II is worldview

independent. Unfortunately its complexity makes it more difficult to learn then the other SMT's reviewed. Models can be created to operate as Java applets to assist in sharing.

Ptolemy II has a debugging and verification mode, and tools to allow validation of results. Models can be run in step wise and fast modes. Error messages are moderately helpful, but a trace file is created.

One area that Ptolemy II is weak in is the tools for experimentation. This is primarily due to the fact, that most work to date has been in signal processing and electronics simulation. To allow for warm up periods, independent replications, reinitialzation, breakpoints, speed adjustment, checking accuracy, and determining run length routines would need to be developed. But a feature of Ptolemy II is the creation of higher level controls (called directors) to handle such items. Ptolemy II provides the basic statistical tools, such as standard distributions, seeding random numbers, sampling, and distribution fitting. Alternative distribution functions and alternate random number streams are easy to implement.

Ptolemy II is a development environment and is not a commercial package. There is a large amount of documentation and several demonstration models. But the tutorial is rather difficult to follow and is not geared for general users, but rather for developers.

## Analysis

For the four (4) SMT packages (Arena, Extend, Sigma, and Ptolemy II), Ptolemy II is the most flexible and offers the most features. But, since it is primarily a test bed, it has few standard features, although it has a true open source system to allow development of features. Arena and Extend, being commercial packages, offer the most comprehensive tools ready to use. They also hide more of the underpinnings. Sigma is the least developed of the four, but offers a compromise between true open source and closed.

Using the format developed above (and shown in Appendix C (Modeling Package Evaluation Form)). The four SMT's were evaluated as:

**Table 16: Modeling Package Evaluation Results**

| | | | Modeling Package | | | |
|---|---|---|---|---|---|---|
| CATEGORY | | Maximum | Arena | Extend | Sigma | Ptolemy II |
| A. | Modeling Environment | 50 | 29 | 28 | 34 | 39 |
| B. | Model Documentation and Structure | 50 | 27 | 27 | 19 | 45 |
| C. | Verification & Validation | 25 | 14 | 15 | 13 | 16 |
| D. | Experimentation facilities | 50 | 48 | 42 | 36 | 33 |
| E. | Statistical facilities | 50 | 39 | 38 | 27 | 32 |
| F. | User support | 50 | 39 | 32 | 33 | 35 |
| G. | Financial and technical features | 25 | 16 | 16 | 13 | 23 |
| TOTAL | | 300 | 212 | 198 | 175 | 223 |

No specific factors were assigned to give extra weighting for any of the criteria. If a particular need or requirement was more important than others fro a particular project, weights could be adjusted to reflect this.

## Simulation in Decision Support

Page (1994) in his dissertation stated that discrete event simulation at its most fundamental level is a tool for decision support. From this is can be inferred that DES is a subset of DSS and should have many of the same characteristics.

Nance's (1993)(1995) six characteristics or requirements of a discrete event simulation language can be used as a starting point.

- generation of random numbers to represent uncertainty,
- process transformers, to permit other than uniform random varieties to be used,
- list processing capability, so that objects can be created, manipulated, and deleted,
- statistical analysis routines, to provide the descriptive summary of model behavior,

- report generation, to provide the presentation of potentially large reams of data in an effective way for decision making, and
- a timing executive or time flow mechanism.

Labadie (2004) presented t that the components of a decision support system consists of a data base management system (DBMS), a user interface, and an analysis and modeling component.

The DBMS is for:

- Coordination, integration, integrity, storage and extraction of information
- Separation of data and decision models

The user interface is for:

- user/model interaction; alerting
- Manipulate model--check logic
- Input data during model execution
- Define display preferences; colors, windowing

The analysis and modeling component is for:

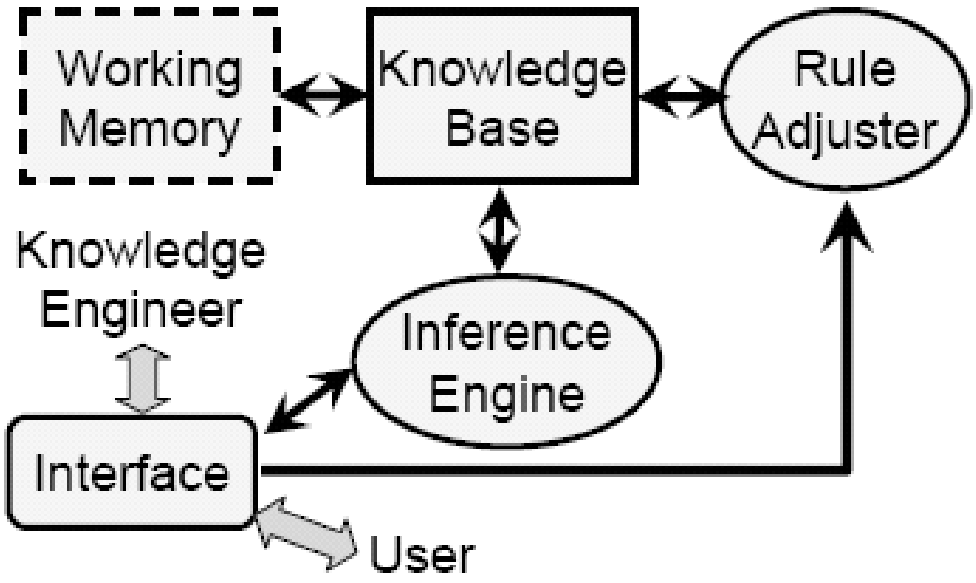
- Statistical data analysis
- Forecasting algorithms
- Simulation, optimization and other OR methods
- Knowledge encoding

All DSS's must contain all 3 of these components to some extent. In general the requirements of DSS softeware are:

- End user friendly and interactive; self contained; on-line; End user designed
- Easy access to pertinent information
- Data access and meaning
- Variables; use of scripting languages (e.g., PERL)
- Decision analysis techniques

- Statistical, simulation and optimization tools
- Syntax and semantics necessary for usage
- Knowledge about rules
- How information displayed
- High interaction between user and system
- Ability to evolve to user needs
- Portability
- Reliability
- Acceptable performance

In the development of DSS a goal is often to create an expert system to guide the user into making the proper decisions. From Labadie (2004), an expert system, also known as a knowledge network, is usually a computer program that contains some of the subject-specific knowledge, and contains the knowledge and analytical skills of one or more human experts.




Labadie (2004)

**Figure 11: Generic Expert System**

The expert system is made up of a set of rules that analyze information supplied by the user of the system or through some data acquisition system (Fonseca and Knapp

(2000)). The system will have the ability to provide mathematical analysis of the problem(s), and, depending upon their design, recommend a course of user action in order to implement corrections. It is a system that utilizes what appear to be reasoning capabilities to reach conclusions.

Beyond the basic expert system is the class of artificial intelligence. Artificial intelligence (or AI) where a program or routine (usually called an “agent”) has been developed that can perceive its environment and takes actions which maximizes its chances of success. John McCarthy, who coined the term in 1956, defines it as "the science and engineering of making intelligent machines."

## DES in Decision Support

Applications of DES in DSS has while not uncommon is still not an everyday use. Others have referenced similar problems and solutions. The use of this approach was articulated by Johansson and Kaiser (2002):

“World-class utilization of manufacturing resources is of vital importance to any manufacturing enterprise in the global competition of today. This requirement calls for superior performance of all processes related to the manufacturing of products. One of these processes is the resetting of machinery and equipment between two product runs, which will be the focus area of this text. This paper examines to what extent Discrete Event Simulation (DES) can be applied to the evaluation of resetting performance in manufacturing systems.”

A more common approach was shown by Balasubramanian and Grossmann (2004) where they solve a scheduling problem for a chemical batch plant using mixed integers and two-stage approach. Castillo and Roberts, (2001) presents using DES and search algorithms to optimize batch chemical plant operation. They provide an example using timed Petri Nets. Chen, Lee and Selikson (2000) describe the development of a semi-hybrid model of a chemical processing facility. The system modeled is the storage and shipping side (from plant to storage to shipment), emPlant was used, with a unit size of

2 tonnes to provide a semi-continuous system. Experiments were conducted using different numbers and sizes of silos. Joines, Sutton, Thoney, King, and Hodgson (2003) describe the application of Virtual Factory for a scheduling simulation. López-Mellado, Villanueva-Paredes and Almeyda-Canepa (2005) describe the use of Petri Nets for modelling a batch system. They provide a case study showing using Petri Nets in a chemical batch process plant. Mazziotti and Horne (1997) present the use of DES used for scheduling in the clothing industry. They Describes the organization of files and model for a clothing manufacturer. Sims (1997) presents an overview of using DES for scheduling with examples from manufacturing and service industries. Vaidyanathan, Miller, and Park, (1998) examines a coffee roasting and packaging operation using SIMAN. They look at alternative schedules for each of the operating stages using DES, with the goal to improve utilization while meeting delivery requirements. Williams and Narayanaswamy, (1997) use AutoMOD to develop a scheduling and planning tool for a heavy industrial operation. Three different crane choices were evaluated to determine which could meet the desired schedule. Production rate, material delivery, and delivery sequence were studied.

A classical DES approach is in simulation and modeling of a system.

## Equipment Sizing Of A Material Handling System

When designing a new material handling system or looking at modifying an existing system there is always that nagging doubt in the back of your mind until you actually run material through, “is it sized properly?” The conventional approach to sizing equipment for a material handling system has used the design material balance and rules of thumb to provide a system capable of handling fluctuations. This system sizing uses engineering judgment and experience factors. Engineering judgment and experience are very important, and I have successfully sized many systems this way.

An alternative approach is to use a DES to model the system and provide a better understanding of the system capacity. By simulation of the proposed coarse ore system, the effect on though put of various equipment capacities can be modeled. This

can give the designer extra insight into the system and allow modifications to maintain desired capacity while working to achieve a lower installed cost.

The operator can also use this tool to determine where bottlenecks exist when looking at proposed increased capacity alternatives. Under-sizing surge bins, conveyors and stockpiles can lead to delays and lost production. Oversized systems are expensive, both in capital and later in operating and maintenance costs. As an example the cost of components for a 3 meter belt versus a 1 meter belt can double the capital cost of a conveyor system.

DES has been used extensively in manufacturing to simulate assembly lines, distribution systems, and similar. This same work has direct application in mining particularly in equipment sizing and system capability analysis. DES systems can create relatively detailed model and simulations of the material handling system.

The manufacturing industry has been a major user of simulation to reduce in process storage and surge capacity, with the goal of increasing throughput without increasing facility size. Benjaafar (2002) describes the use of a process oriented DES (ARENA) to study congestion in manufacturing material processing systems. The purpose being to reduce in process storage and surge capacity.

For similar reasons this topic has been looked at by several mining operations.

Giacaman, Medel, and Tabilo (2002) describe the impact on capacity of changes in truck and loader size. Brunner, Yazici and Baiden (1999) used Automod to evaluate the capacity and life of a mine under varying conditions and to evaluate the impact of changing system details (size and capacity) on the total mine life and throughput.

## Project Definition

This analysis is based on a study I worked on several years ago for open pit copper operation in the Southwest United States. This was the rehabilitation of an older mine that had been closed 10 years previously. The production rate for the proposed new concentrator has been set at 21.5 ktons per day. The operation was to include a primary crusher, overland conveyor, coarse ore stockpile and reclaim system, semiautogenous (SAG) and ball mill grinding, flotation, tailings disposal, concentrate filtering and load out facilities. This review is only looking at the coarse ore system (Figure 2).

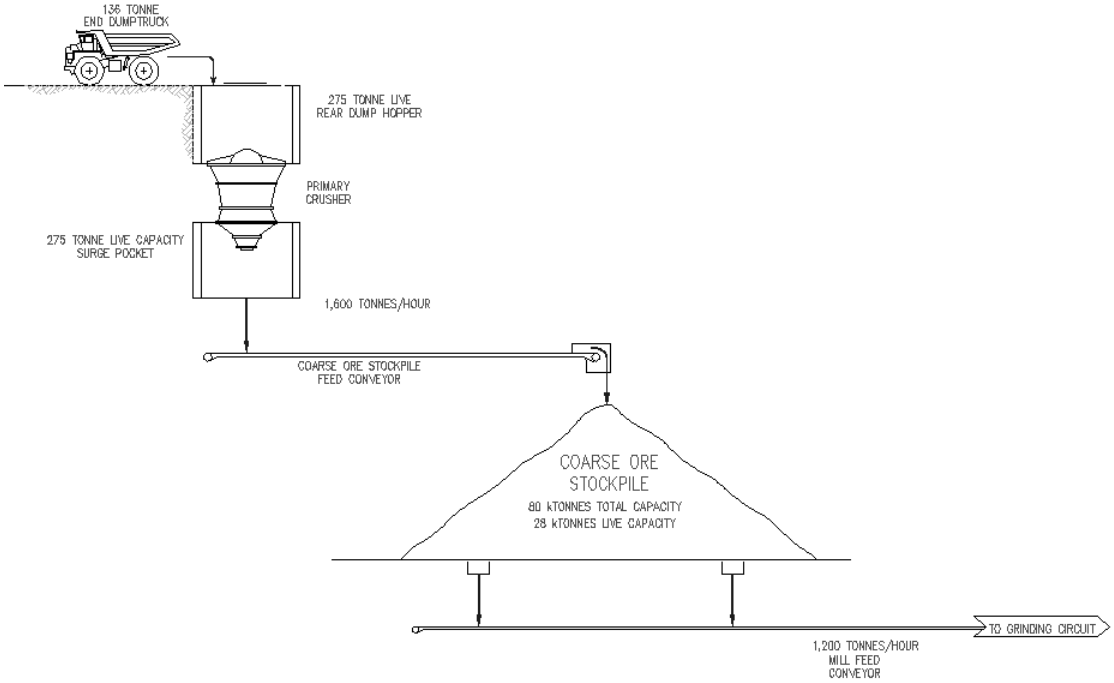



**Figure 12: Initial Process Flow sheet**

This example uses an event graph based DES (SIGMA©) to model a truck dump, primary crusher, surge bin, stockpile feed conveyor, coarse ore stockpile, and reclaim as an example. The nominal capacity is set at 21,500 tonne/day. The system runs 24 hours/day. The goal is to assess the impact of alternative equipment sizing on the stockpile requirements. Large stockpiles take more area, and longer and higher conveyors, which directly increase capital and operating costs. The factors that are looked at are how many reclaim points should be used, what width and speed the

conveyors should be, and what size surge bins makes sense. The process design criteria is shown in Table 1.

**Table 17: Process Design Criteria**

| | | |
|---|---|---|
| Ore to process facilities | | |
| dry tons/day | 21.5 | kt/d |
| dry tons/year | 7.85 | Mt/y |
| moisture | 2.50% | |
| density | 2.8 | g/cc |
| Dump Hopper | | |
| feed method | 136 | ton end dump truck |
| Truck Feed rate | 10 | trucks/hr (max) |
| Truck Feed rate | 6.6 | trucks/hr (ave) |
| Truck Feed rate | 3.6 | trucks/hr (min |
| dump pocket size | 275 | tonnes (live) |
| availability | 65% | |
| type of crusher | | gyratory |
| nominal treatment rate | 1600 | t/hr |
| maximum feed size | 1065 | mm |
| product size (p90) | 203 | mm |
| storage above stockpile belt | 275 | tonnes (live) |
| Crushed ore stockpile | | |
| feed method | | conveyor; single point discharge |
| capacity | 1600 | t/hr |
| capacity | 28 | kt (live) |
| | 80 | kt (total) |
| Plant Feed | | |
| drawdown angle | 55 $^{o}$ | |
| feed method | | conveyor; single point discharge |
| capacity | 1200 | t/hr |

BUILDING THE MODEL

The model for the system shown in figure 3.

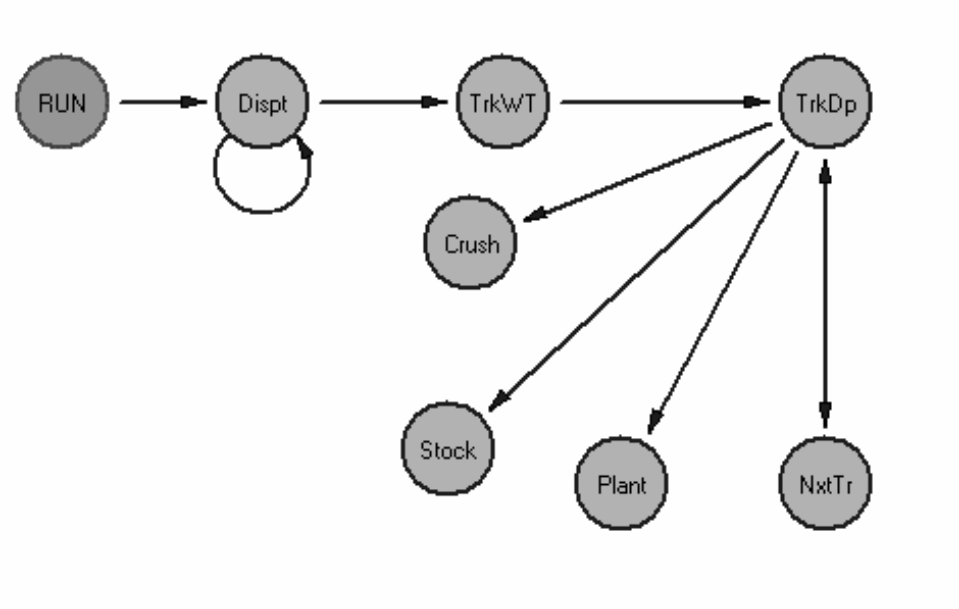



**Figure 13:SIGMA© Event Graph Model**

## Simulation Results

Starting with the initial values in Table 2 the model was run three times each for a simulated 30 day period. For each run the amount in the dump hopper, the conveyor surge bin and the stockpile were recorded.

The initial sizing and final sizing based on the results at the end of run 3 are summarized in Table 2

**Table 18 : Simulation Results**

| | | Initial | Final | % change |
|---|---|---|---|---|
| Dump Hopper Size (live) | tonnes | 275 | 175 | 36% |
| Primary crushing nominal rate | t/hr | 1600 | 1050 | 34% |
| Crushed ore stockpile feed conveyor capacity | t/hr | 1600 | 1200 | 25% |
| Plant Feed conveyor capacity | t/hr | 1200 | 900 | 25% |

Reducing the size of the hoppers, and reducing the capacities of the crusher and conveyors did not increase the truck wait time, or significantly alter the amount of

material in storage. Required capacities of the dump hopper and surge bin can be reduced from 275 tonnes to 175 tonnes each. The crusher capacity can be reduced from 1600 tonnes/hr to 1050 tonne/hr. The stockpile conveyor can be reduced from 1600 tonne/hr to 1200 tonne/hr. And the plant feed conveyor can be reduced from 1200 tonne/hr to 900 tonne/hr. All of this without impacting the system through put.

These results do not look at maintenance down time, or other problems. These can be modeled also, and would be the next step to add in the analysis.

Other applications can be more complex and lead to DSS systems.

## Decision Support System for Process Changeover

A special case existis in process changeovers. Many chemical process operations produce different products using the same equipment. This requires doing a process changeover from one product to another. When changing from one product to another it is necessary to clean the system and prepare it for the next production run. Depending on what product was being produced and what product will be produced, the steps and timing varies. When a new product is introduced or being evaluated for production this sequence needs to be determined. Also it is often advisable to evaluate the existing sequences to verify that best allocation of resources is being followed, especially when a different sequence of products is being run. Using a DES in a DSS will allow production planning personnel to prepare process changeover plans that utilizes the minimum resources (manpower) and accomplishes the changeover in the minimum time.

Criterion Catalysts & Technologies (CC&T) Pittsburg Operation produces over 16 different types of catalysts, for these products the same equipment is used, but the order and type of material addition is different. To prevent cross contamination and to achieve a high first time quality it is necessary to assure that the system is clean before changing to another product. This is especially important when changing from one

major type to another (e.g. acid to base or reverse), but also necessary when changing between different catalysts of the same solution type.

Pittsburg operates 7 days a week 24 hours a day. Production runs can be from 8 hours to 100+hours, most being in the range of 60 to 72 hours. Changeover can be accomplished in as little as 2 hours or can take up to 16 hours, minimizing changeover time is important. Depending on the tasks involved, maintenance manpower can be required. With maintenance working 8 hours/day 5 days/week, changeovers can require overtime and or emergency call outs, minimizing their required time is also important.

In an effort to better understand the changeover requirements the CC&T Process Planning personnel used MS Project to prepare a schedule of the Base Side to Acid Side process change (see figure 4 below) in the fall of 2003. This provided some help in identifying the items that were taking the most time. The use of this tool proved cumbersome and not very helpful in reducing the time required.

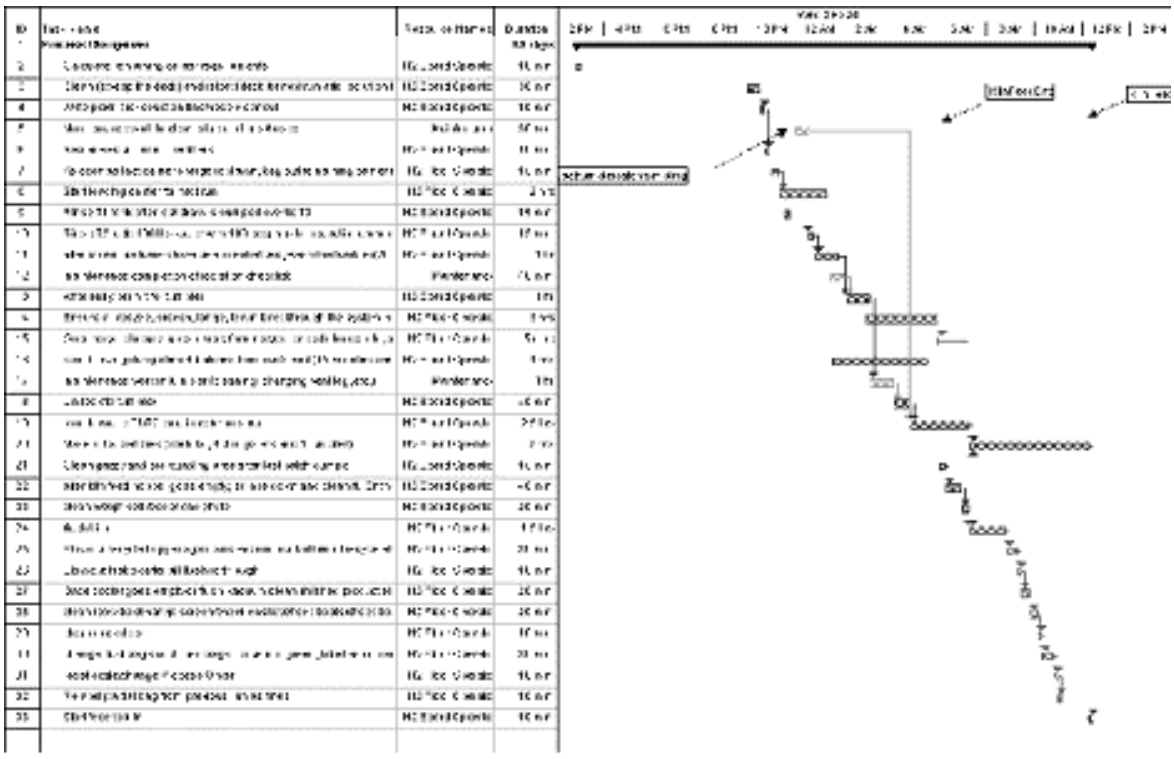


**Figure 14: MS Project Schedule for Base Side to Acid Side Changeover**

With the difficulties found using MS Project this project was to develop a DSS for this problem based on a DES system.

## Structure

The DSS is a two-stage system combining a spreadsheet for user interface and data storage with a task oriented discrete event simulation (DES) package to perform analysis and modelling.

The system includes the following:

- Display (dialog) – User interface is by use of Excel, for its ability to interface with other routines and ability to provide output for more formal reports. Also the potential users are quite familiar with Excel and its uses.
- Database – Excel is also used for the database of tasks and required resources.
- Problem analysis and modeling – Modeling is done using Sigma95 a simulation package.

For this project the software used is a graphical task oriented package, SIGMA© (Simulation Graphical Modeling and Analysis) by Custom Simulations (Sigma@CustomSimulations.com) in Berkeley, California. SIGMA© is a simulation modeling environment, based on using event graph modeling. The SIGMA© software allows the development of models in a graphical environment and includes the use of animations and statistical tools. SIGMA© is a fairly easy to use task oriented system that is suited for projects that have specific tasks or events that occur such as truck dumping, feeding a crusher, feeding a conveyor belt, or loading a stockpile. The manual “Event Graph Modeling Using SIGMA for Windows” (Schruben 1995) can be used for as an introduction.

Determining the steps for each changeover is traditionally done using a GANTT style bar chart to determine the resources needed and time required. One draw back to this is that to use scheduling software well requires practice and continual use. Also, most scheduling programs (such as MS Project) are fine for outlining the tasks and allocating

resources, but have a problem when performing “what if” analysis to determine minimum time and resource allocations. Simulation software is excellent for “what if” analysis, but also takes time to learn, and the output can be difficult to analyze, especially when being used to prepare operating procedures for the plant floor.

The first step was to define what events had to occur for the process changeover. Starting with the MS Project schedule above a list of the events and the tasks required for these events was generated. This is shown in:

**Table 19: changeover Event and Task List**

| Event No. | Event Description | Task No | Task Description |
|---|---|---|---|
| 1 | Last Carrier Charge Drawn | 1 | Calculate remaining carrier requirements |
| 2 | Clean Carrier Hopper | 2 | As soon as last carrier charge is drawn, bag out remaining carrier if necessary |
| 3 | Reload Carrier Hopper | 3 | Start loading carrier for next run |
| 4 | Drum Dump Hoppers Empty | 4 | Calculate remaining powder requirements |
| 5 | Last solution Pumped to tumbler | 5 | Final Powder fed to tanks |
| | | 6 | Final Carrier Fed to Tumble |
| 6 | Last Tumbler dump | 7 | Last Tumbler dump |
| 7 | Last Batch dumps | 8 | Last Batch is made |
| 8 | Kiln Feed Ends | 9 | Ensure all recycle, breech, longs, is run back through the system before kiln feed hopper goes empty |
| 9 | Kiln Feed system is Clean | 10 | Prepare to Clean Kiln Feed |
| | | 11 | Clean grizzly and surrounding area after last batch dumps |
| | | 12 | After kiln feed hopper goes empty, scrape down and clean it. Entry may be required if a large brick is in hopper due to wet batches. |
| | | 13 | Clean weigh belt feeder and chute. |
| 10 | Last Product Bag Off | 14 | Last Product Bag Off |
| 11 | Kiln Flush ends | 15 | Prepare to Flush Kiln |
| | | 16 | Once recycle hopper goes empty from recycle, breech, longs, etc, blow down before kiln feed hopper goes empty |

| | | | |
|---|---|---|---|
| | | 17 | Flush kiln |
| | | 18 | Blow out recycle hopper again and vacuum out bottom of recycle elevator. |
| 12 | Tanks Clean | 19 | Prepare to Clean Tanks |
| | | 20 | Rinse T1 tank after last draw is pumped over to T3 |
| | | 21 | Rinse T2 tank after last draw is pumped over to T3 |
| | | 22 | Rinse T3: add 100 lbs water with 100 lbs phos to neutralize ammonia…. Pump to tumbler after last batch dumped, tumble for 5 minutes. |
| 13 | Tumbler clean | 23 | After ammonia fumes have been neutralized, wash the tumbler(20 min) |
| 14 | Dump Hoppers Clean | 24 | Start blowing dumpsters 4 batches from run's end (15 minutes per assuming dumpsters aren't removed and chipped clean due to wet batches) |
| 15 | Writer Permits | 25 | Write permit for eduction line/hose cleanout |
| | | 26 | Write entry permit for tumbler |
| 16 | Complete Maintenance | 27 | Prepare for Maintenance |
| | | 28 | Complete isolation checklist and write permit for isolation to prepare tumbler for entry (40 minutes) |
| | | 29 | Maintenance completion of isolation checklist |
| | | 30 | Maintenance work to clear eduction lines/hoses |
| | | 31 | Maintenance work in tumbler (cleaning, changing vent leg, etc.) |
| | | 32 | Replace eduction in-line filters |
| 17 | Prepare the Deck for Next Run | 33 | Clean (sweep the deck) and reload deck for next run after solution tank drum feed hoppers go empty |
| 18 | Start New Solutions | 34 | Un-isolate tumbler |
| | | 35 | Start to make T1/T2 solutions for next run |
| | | 36 | Tanks Clean |
| 19 | Make Four Batches | 37 | Dump Hoppers Clean |
| | | 38 | Make initial batches (five total, 4 dumpsters and 1 tumbler) |
| | | 39 | Draw carrier charge to tumbler |
| 20 | Product System cleaned | 40 | Prepare to Clean Product System |
| | | 41 | Blow out firebox after all flush is through |

| | | | |
|---|---|---|---|
| | | 42 | Once cooler goes empty of flush, vacuum/clean finished product elevator |
| | | 43 | Clean Rotex box/change screen/check gasket/check boots/check balls |
| | | 44 | Clean Rotex deck |
| 21 | Prepare for Next Product | 45 | Change dust bags and fines bags if blue into green, label as contaminated. |
| 22 | Set up Packaging for Next Run | 46 | Last Product is Bagged Off |
| | | 47 | Prepare for next product packaging |
| | | 48 | Reset scale/change Process Order |
| | | 49 | Relabel partial bag from previous run as fines |
| 23 | Start Kiln Feed | 50 | Kiln Feed system is Clean |
| | | 51 | Make Four Batches |
| | | 52 | Prepare for Next Product |
| | | 53 | Packaging Set up |

From this a flow diagram of the changeover (figure 2) was developed (note figure 5 includes some additional events required for the simulation).

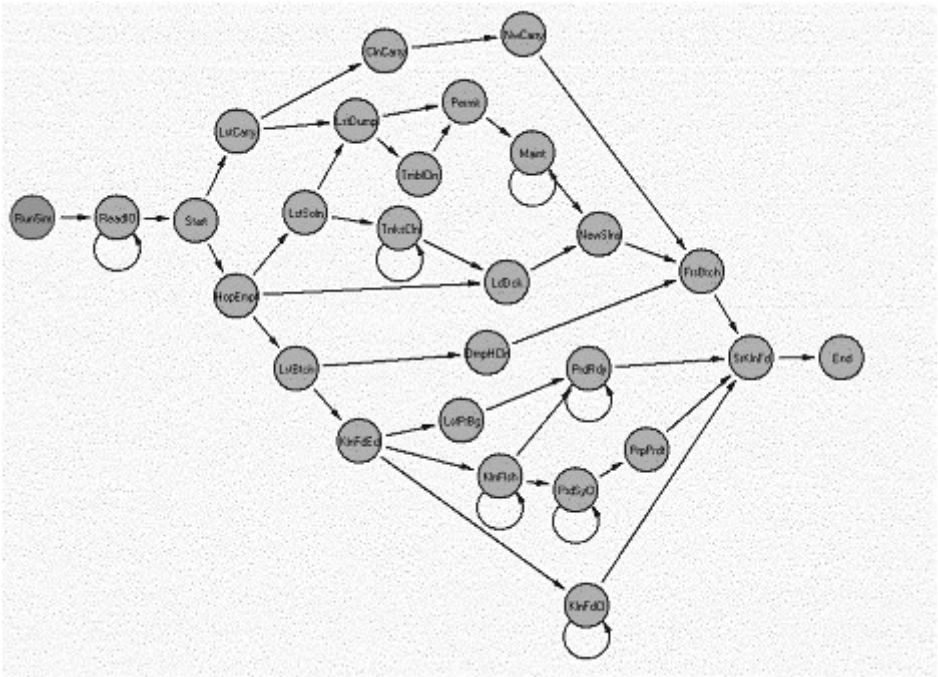



**Figure 15: Network Analysis of H2 Process Changeover (SIGMA organization)**

The task duration's were established and are being refined by operations. They are included in the spreadsheet.

## System Operation

The system uses both Excel Spreadsheet and Sigma95 for the DSS. The database of information that is verified is stored in Excel. The user can adjust the task duration, the resources used, and the task priority.

Opening RunH2changeover.xls will bring up the first screen that describes what can be changed and how the basic system operates. The system uses macros that must be enabled for the program to operate.

The DSS is run from an Excel spreadsheet (Figure 6).

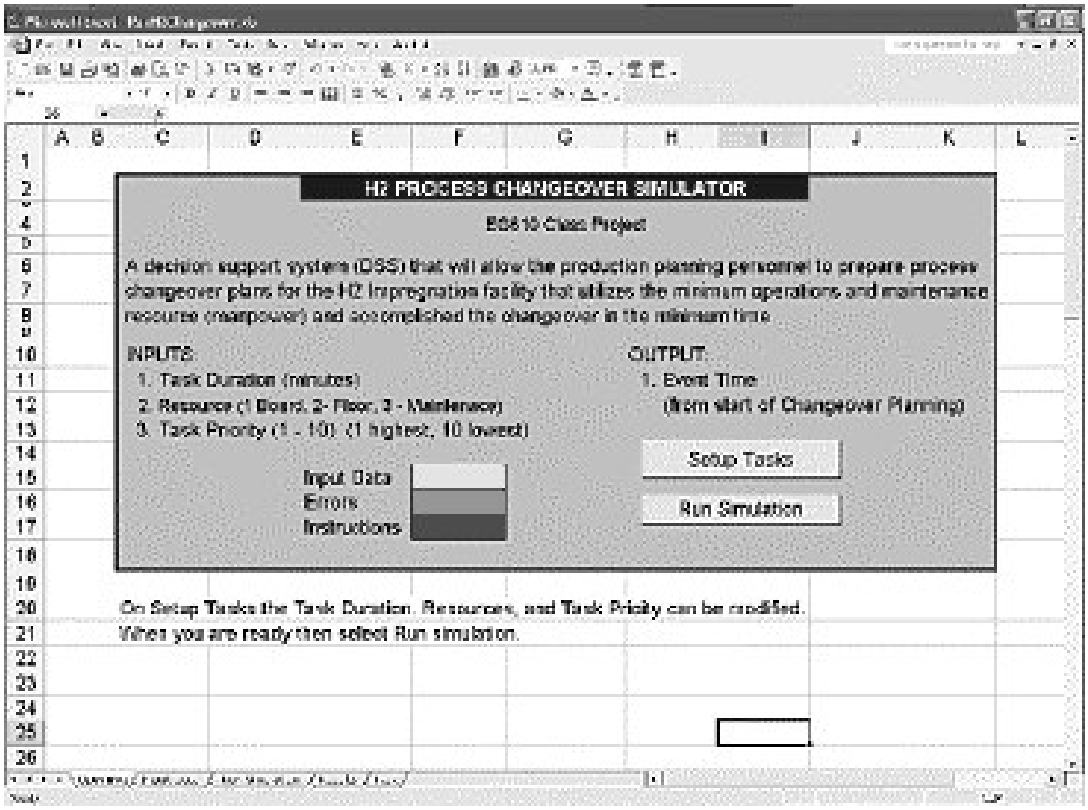
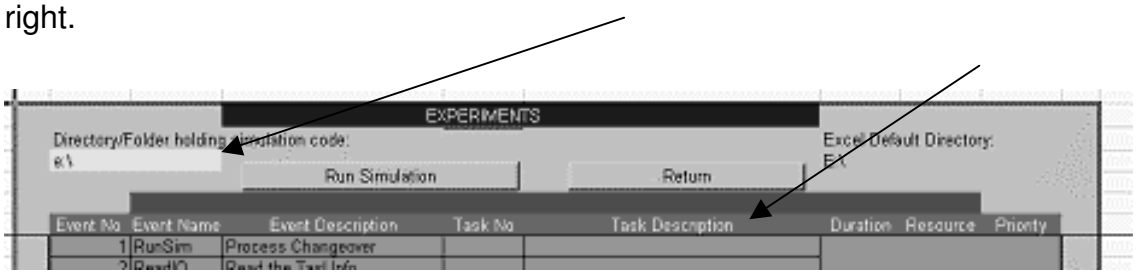



**Figure 16: System Access Screen**

The first step is to go to the Setup Tasks screen (click the Setup Tasks button). From the Setup Tasks sheet the Duration, Resources, and Priority can be changed. The

system requires that all the files be in the current directory. If the information is already correct go to Run simulation.

Make sure that the directory listed in the upper left is the same as the one in the upper right.




The simulation and output are obtained from the Run simulation sheet. (Figure 7).

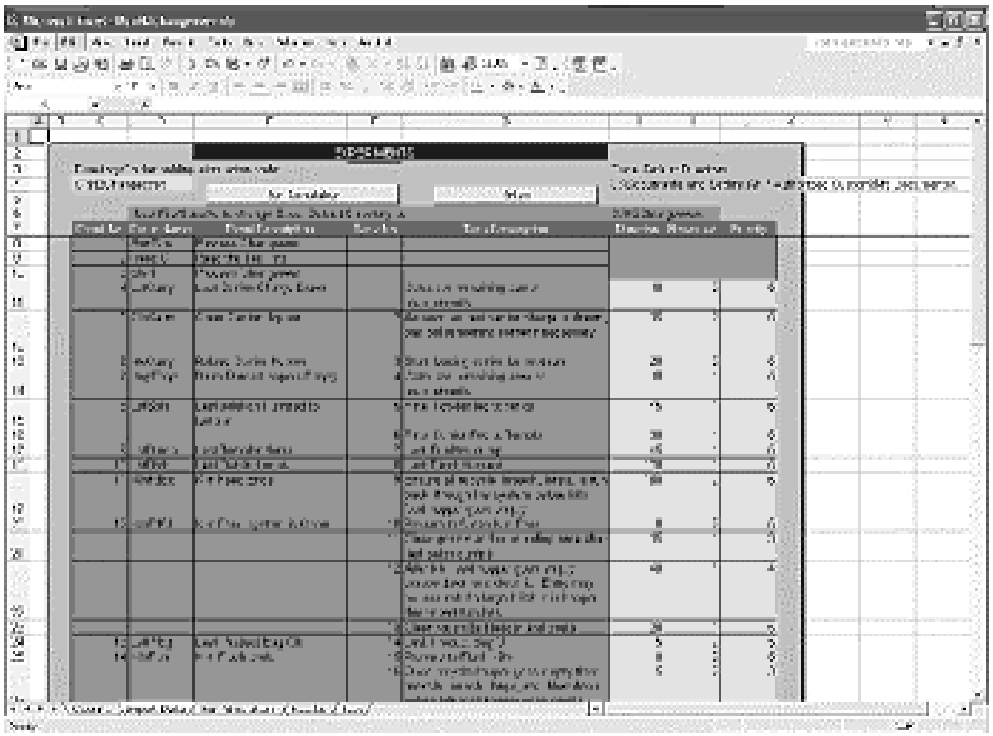


**Figure 17: Event & Tasks Lists**

Once all the task information is correct click the Run simulation button to go to the Run Simulation screen (figure 8). The simulation is run from this screen.

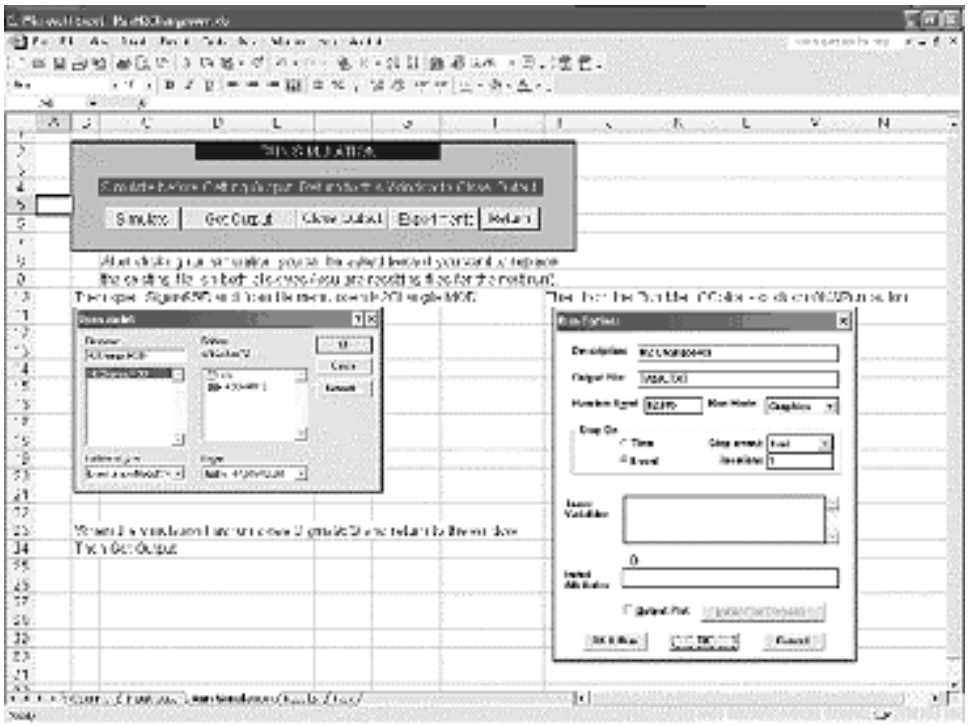
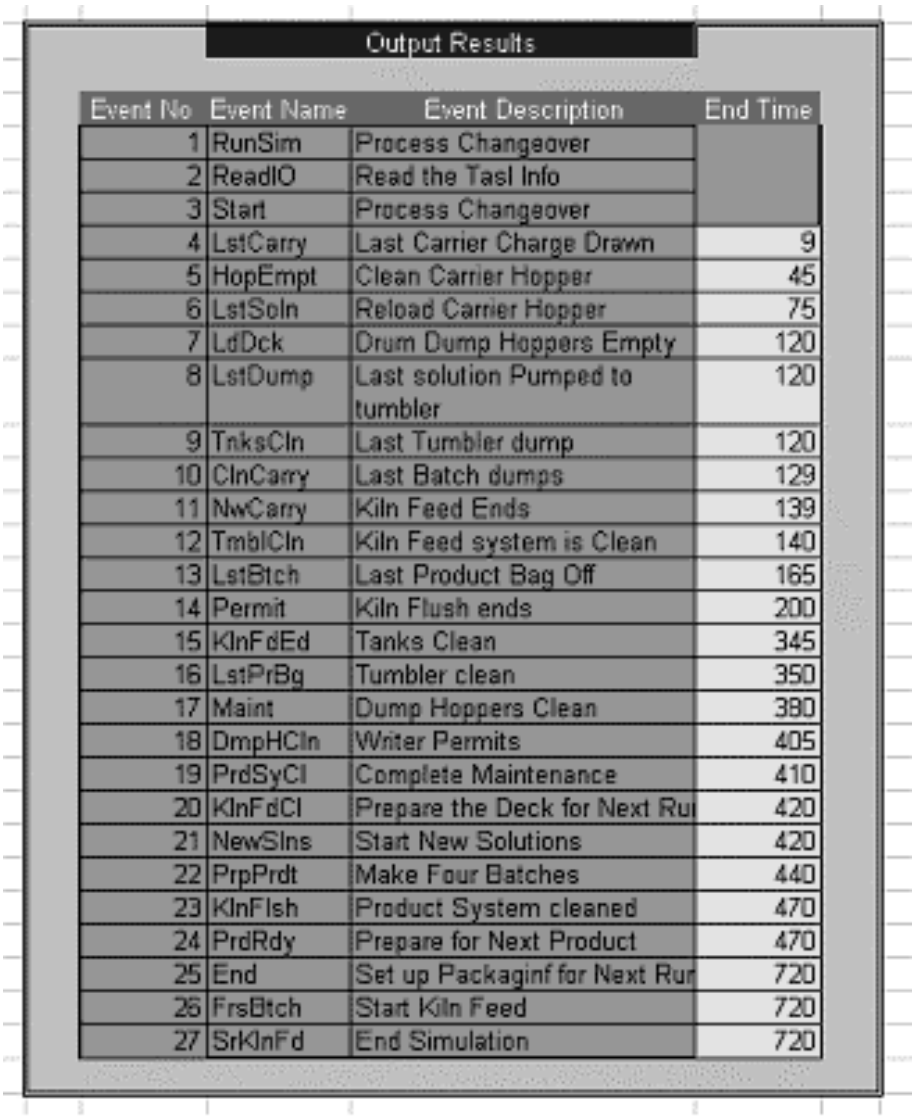


**Figure 18:Simulation run Screen**

Sigma95 will create an output file  which is then read by the spreadsheet and the results displayed

## Results

The DSS will calculate the end time for the last replication of each main event (figure 9).



Output Results

| Event No | Event Name | Event Description | End Time |
|---|---|---|---|
| 1 | RunSim | Process Changeover | |
| 2 | ReadIO | Read the Tasl Info | |
| 3 | Start | Process Changeover | |
| 4 | LstCarry | Last Carrier Charge Drawn | 9 |
| 5 | HopEmpt | Clean Carrier Hopper | 45 |
| 6 | LstSoln | Reload Carrier Hopper | 75 |
| 7 | LdDck | Drum Dump Hoppers Empty | 120 |
| 8 | LstDump | Last solution Pumped to tumbler | 120 |
| 9 | TnksCln | Last Tumbler dump | 120 |
| 10 | ClnCarry | Last Batch dumps | 129 |
| 11 | NwCarry | Kiln Feed Ends | 139 |
| 12 | TmblCln | Kiln Feed system is Clean | 140 |
| 13 | LstBtch | Last Product Bag Off | 165 |
| 14 | Permit | Kiln Flush ends | 200 |
| 15 | KlnFdEd | Tanks Clean | 345 |
| 16 | LstPrBg | Tumbler clean | 350 |
| 17 | Maint | Dump Hoppers Clean | 380 |
| 18 | DmpHCln | Writer Permits | 405 |
| 19 | PrdSyCl | Complete Maintenance | 410 |
| 20 | KlnFdCl | Prepare the Deck for Next Ru | 420 |
| 21 | NewSlns | Start New Solutions | 420 |
| 22 | PrpPrdt | Make Four Batches | 440 |
| 23 | KlnFlsh | Product System cleaned | 470 |
| 24 | PrdRdy | Prepare for Next Product | 470 |
| 25 | End | Set up Packaginf for Next Ru | 720 |
| 26 | FrsBtch | Start Kiln Feed | 720 |
| 27 | SrKlnFd | End Simulation | 720 |

**Figure 19: Output results Table and Changeover Time**

In addition it will calculate the net time between the end of Kiln Feed and the start of the New Feed to the Kiln (figure 6)

This will be the net lost production time.

The development of this DSS has proven to be very complex, more so then the operations and planning personnel knew. While no surprises have been uncovered, the detail and inter-relation of the many tasks needed to perform the changeover is extensive.

The results have been enlightening, but have resulted in a partially complete project. The current DSS is a work in progress. The basic logic has been defined and tested, but implementation of alternative process changeovers needs to be included. It is also planned to include resource allocation to determine that available resources can accomplish the changeover in the optimal time. Once the model has been completely verified and validated, the goal is to set up the actual simulation as C++ file that would be run seamless from within Excel.

## Hybrid Simulation

Both analytical models and DES have advantages and disadvantages in modeling a system (Byrne and Bakir (1999)). A purely analytical method can have issues when dealing with queues and and alternating production rates. At the same time DES has an inherent limitation in handling continuous events. Because of these work towards a hybrid system, which combines discrete and continuous functions in one simulation run has seen some effort. This is either done by using a stochastic system to generate input for a DES or by including truly continuous aspects to a DES. Using a stochastic system to generate input is often a Decision Support System approach. Incorporating continuous functions in a DES can lead to a hybrid language.

Hybrid Simulation has several different meanings depending on who is using the term.

Duse (1994) in his dissertation defined the term as a combination of standard DES and a Fast simulation where Fast means a model that does not maintain an events list. Again this is treated as a standard DES for this review.

The Defense Modeling and Simulation Office define Hybrid Simulation as simulation with live and virtual components. While differing from creating a DES model and running it under varying conditions, this can be considered more of a game approach, and is not specifically treated in this review.

## Stochastic/DES

Some refer to discrete event/stochastic (stochastic conditions determine DES conditions) as hybrid (Venkateswaran and Son (2005)). In the early days of DES, most SPL's had the ability to handle discrete events only. Y events occurred at X time intervals, but today most packages allow for the events to occur at random intervals, using alternative frequency distributions. But as most DES SPL's and packages are capable of stochastic event conditions this is not separated out from general DES in this analysis. In stochastic/DES simulations you have the ability to do stochastic simulation on a discrete system.

## Discrete/Continuous

Discrete/continuous applications have both a discrete and continuous nature. In the real world problems often have both a discrete and a continuous component; hybrid models cover the recognition that some applications have both a discrete and continuous nature. Examples of hybrid languages have also been described.

For this review the definition as proposed by Shanthikumar and Sargent (1983) is used:

> "Hybrid systems are systems that evolve with both discrete and continuous behaviors".

They went on to describe four (4) types of hybrid systems combining analytical portions and simulation portions.

> I: Results are obtained by alternating between independent analytical and simulation models; after the solution for each is derived the solutions are then combined for an overall solution.
>
> II: An analytical and simulation model operate in parallel and their output is compared together.
>
> III: A simulation model is used to feed an analytical model.
>
> IV: An analytical model is used to feed a simulation model.

The majority of the work in discrete/continuous simulation has involved types III and IV.

A major reason for the use analytical models is that they can be lower in development cost then simulations and are usually low in cost to run once developed. Simulation models, though, give more detail and are usually more realistic. Combining them in to discrete/continuous systems can give strong benefits of both. Type I or II are used if the time dependent nature of the system can be broken out, with type I being used if the time dependent nature can be completely separated and type II if their is some interdependency. These types are also used when the simulation model is used for validating the analytical models output. Type III is used when a simulation of system or subsystem is used to determine some or all of the values used in the analytical model. Type IV is used when an analytical model is used to determine some or all of the inputs for the simulation.

Petropoulakis and Giacomini (1998) present a system for modeling a manufacturing process using both continuous and discrete model components. They present that few manufacturing systems are purely discrete or continuous, but rather a combination of both. They propose a combined discrete and continuous system. Their system uses Simple++ as the DES component and SAM for the continuous portion. The DES is top level system which calls the continuous portion as needed. This would compare to a type IV system as described by Shanthikumar and Sargent (1983).

By inference a fifth type (type V) would be where an analytical model feeds a simulation, which is then fed to an analytical model. This gets even closer to true artificial intelligence.

## A Combined Analytical DES Model

Some of the earliest work on a combined (hybrid) system was by Cellier (1979). Cellier (1979) in his dissertation, describes techniques for simulating systems with complex

structures by use of a digital computer, as well as the requirements of tools (simulation languages) to cope with the problem in a user-friendly way. He describes three (3) classes of simulation problems

(i) continuous time systems described by ordinary differential equations

(ii) continuous time systems described by partial differential equations

(iii) discrete time systems described by difference equations, sequence of time events or by mixtures of both

For type i and ii, modeling languages have been available for some time, but (at the time of this paper) type ii models are still difficult, he comments that many problems are truly combined i & iii. He then proposes a structure of an SPL to handle combined type I & iii.

(a) a discrete part consisting of elements from known discrete event simulation
(b) a continuous part consisting of elements from known continuous system simulation
(c) an interface part describing the conditions when to switch from (a) to (b) and vice-versa

Cellier then describes the conditions and characteristics for part (c). His proposed system is essentially an executive program that manages the activities of a DES and a continuous simulator and keeps track of which one should function next.

To examine the use of DES in a hybrid simulation case, the work of Byrne and Bakir (1999), Kim and Kim (2001), and Byrne and Hossain (2005) has been examined. The three papers examine a multi-period multi product (MPMP) production scheduling problem. The problem is based on a three (3) product, three (3) period case. The system has four (4) processing areas (Machine Centers) each with one (1) production unit. Each unit has a maximum capacity of 2400 minutes (40 hours) per period.

The basic system and input data for all three is the same. The variation is in the LP model. They each iterate around a linear programming model interfaced with a DES. Their hybrid solution process is as follows:

Step 1. Generate optimum production plan by the LP model

Step 2. Assign optimum production plan from LP model as input to the simulation model

Step 3. Run simulation model subject to operational criteria

Step 4. Check capacity constraints: if capacity permits go to step 7 if capacity does not permit go to step 5

Step 5. Calculate adjusted capacity

Step 6. Go to step 1

Step 7. Generate production schedule for shop floor based on generated unit load size

Step 8. Stop

This is shown graphically as:

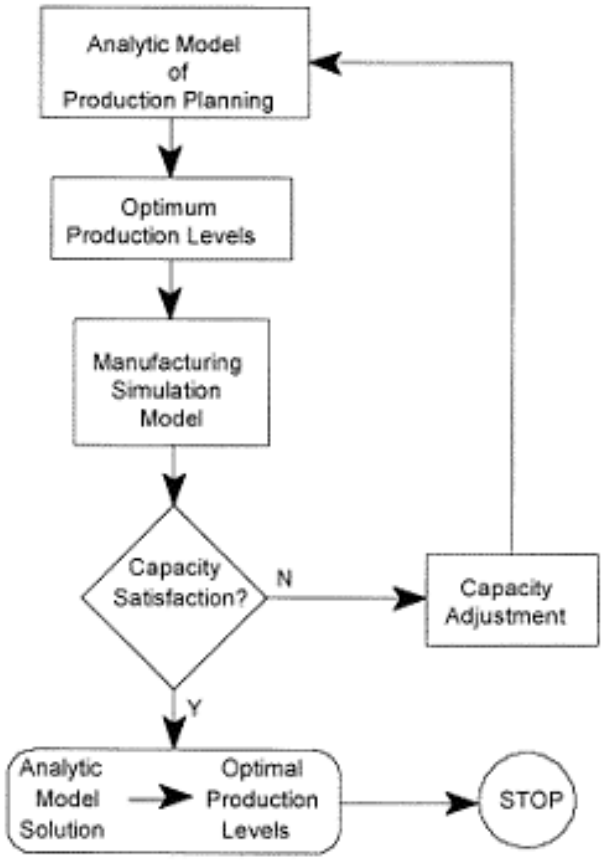



**Figure 20: Hybride Simulation Proceedure**

The model data is also the same, which is shown in tables 1-4 below.

**Table 20: Cost components**

| | Unit production cost | | | Inventory holding cost | | | Shortage cost | | |
|---|---|---|---|---|---|---|---|---|---|
| Periods: | 1 | 2 | 3 | 1 | 2 | 3 | 1 | 2 | 3 |
| Product | | | | | | | | | |
| 1 | 100 | 100 | 100 | 25 | 25 | 100 | 400 | 400 | 400 |
| 2 | 150 | 150 | 150 | 30 | 30 | 150 | 450 | 450 | 450 |
| 3 | 125 | 125 | 125 | 35 | 35 | 200 | 500 | 500 | 500 |

**Table 21: Demand**

| Product | Periods | | |
|---|---|---|---|
| | 1 | 2 | 3 |
| 1 | 150 | 125 | 160 |
| 2 | 100 | 150 | 150 |
| 3 | 125 | 165 | 125 |

**Table 22: Process Times**

| | Process Time (minutes) | | | | |
|---|---|---|---|---|---|
| Machines centers | MC1 | MC2 | MC4 | MC3 | total |
| Product | | | | | time |
| 1 | 5 | | 10 | 4 | 19 |
| 2 | 7 | 7 | | 5 | 19 |
| 3 | 7 | 6 | | 10 | 23 |
| Total | 19 | 13 | 10 | 19 | |

**Table 23: Process Routings**

| Process routings | | | |
|---|---|---|---|
| Product | Machine visit order | | |
| 1 | MC1 | MC4 | MC3 |
| 2 | MC1 | MC2 | MC3 |

3 MC1 MC2 MC3

Byrne and Bakir (1999) model is interfaced with a DES (SIMAN). The LP model from Byrne and Bakir is:

$$Min \sum_{t=1}^{T} \sum_{i=1}^{N} \left( c_{it} X_{it} + h_{it} I_{it} + \pi_{it} B_{it} \right)$$

$$subject\_to$$

$$\sum_{i=1}^{N} a_{ik} X_{it} \le C_{kt}$$

$$I_{it} - B_{it} = I_{it-1} - B_{it-1} + X_{it} - d_{it}$$

$$all\_variable \ge 0$$

Where:

Indices

i index of product, i =1,2,…,N, where N is the number of products.

p,t indices of planning periond, p=1,2,…,T and t=1,2,…,T where T is the planning horizon

k machine type, k=1,2,…,K, where K is the number of machines.

Parameters

$a_{ik}$ processing time to produce a unit of product i at machine type k

$C_{kt}$ capacity (work-hours) of machine k in period t

$d_{it}$ demand for product i in period t

$c_{it}$ unit cost of producing product i in period t

$h_{it}$ inventory holding cost for product i in period t

$\pi_{it}$ backorder cost of product i in period t

Variables

$X_{it}$ quantity of product i produced in period t

$I_{it}$ inventory level of product i in period t

$B_{it}$ backorder level of product i in period t

Kim and Kim (2001) reprise the work of Byrne and Bakir (1999) using a different methodology which modifies both the left- and right-hand sides of the capacity constraints in the LP model. Their main goal is to look at:

(1) how much work each machine performs in each period for the current production mix and volume, and

(2) how much of the full capacity is actually consumed by the current plan by each machine in each period.

They use a C++ derived simulation routine (type unstated). Kim and Kim's extended LP model is based on machine loading

Total workload at machine k in period t $= \sum_{i=1}^{N} \sum_{j=1}^{M_i} \sum_{p=1}^{t} e_{ijk(p,t)} a_{ijk} X_{ip}$

Where the following changes to Byrne and Bakir (1999) are made.

Additional parameters

$a_{ijk}$ processing time to produce a unit of product i in process j at machine type k

$e_{ijk(p,t)}$ effective loading ratio of product i in process j on machine k in period t from starts in period p (indices as before).

$u_{kt}$ effective utilization of machine k in period t

$M_i$ the number of process of product i

Variables

$X_{it}$ quantity to be started (or released into the shop) for production in each period

$Y_{it}$ the output quantity of product i in period t (this variable is redundant, but employed to simplify the presentation).

In the following LP, it can be substituted by the following:

$$Y_{it} = \sum_{p=1}^{t} \sum_{k=1}^{K} e_{iM_ik(p,t)} X_{ip}$$

∴ Their LP Model becomes:

$$Min \sum_{t=1}^{T} \sum_{i=1}^{N} \left( c_{it} Y_{it} + h_{it} I_{it} + \pi_{it} B_{it} \right)$$

$$subject\_to$$

$$\sum_{i=1}^{N} \sum_{j=1}^{M_i} \sum_{p=1}^{t} e_{ijk(p,t)} a_{ijk} X_{ip} \le u_{kt} C_{kt}$$

$$I_{it} - B_{it} = I_{it-1} - B_{it-1} + Y_{it} - d_{it}$$

$$all\_\text{var}iable \ge 0$$

Byrne and Hossain (2005) continue the work of Byrne and Bakir (1999), and Kim and Kim (2001) again using a different LP formulation and present a system that iterates around a linear programming model interfaced with a DES (AutoMod). They use the same hybrid solution process as described in Byrne and Bakir (1999).

Their LP model is:

$$Min \sum_{t=1}^{T} \sum_{i=1}^{N} \left( c_{it} Y_{it} + h_{it} I_{it} + \pi_{it} B_{it} \right)$$

$$subject\_to$$

$$\sum_{i=1}^{N} \sum_{j=1}^{M_i} e_{ijk} a_{ijk} \alpha_i G_{it} \le P_a C_{kt} c$$

$$I_{it} - B_{it} = I_{it-1} - B_{it-1} + Y_{it} - d_{it}$$

$$X_{it} = \alpha_i G_{it}$$

$$G_{it} \le ULS \le PC_i$$

$$all\_\text{var}iable \ge 0$$

Where:

Indices

i index of product, i =1,2,…,N, where N is the number of products.

p,t indices of planning period, p=1,2,…,T and t=1,2,…,T where T is the planning horizon

k machine type, k=1,2,…,K, where K is the number of machines.

Parameters

$a_{ijk}$ processing time to produce a unit of product i in process j at machine type k

$C_{kt}$ capacity (work-hours) of machine k in period t

$d_{it}$ demand for product i in period t

$e_{ijk(p,t)}$ effective loading ratio of product i in process j on machine k in period t from starts in period p (indices as before).

$c_{it}$ unit cost of producing product i in period t

$h_{it}$ inventory holding cost for product i in period t

$\alpha_i$ the number of unit loads of product i

$\pi_{it}$ backorder cost of product i in period t

$u_{kt}$ effective utilization of machine k in period t

Variables

$X_{it}$ quantity to be started (or released into the shop) for production in each period

$Y_{it}$ the output quantity of product i in period t (this variable is redundant, but employed to simplify the presentation).

In the following LP, it can be substituted by the following:

$$Y_{it} = \sum_{p=1}^{t} \sum_{k=1}^{K} e_{iM_jk(p,t)} X_{ip}$$

$I_{it}$ inventory level of product i in period t

$B_{it}$ backorder level of product i in period t

$G_{it}$ the unit input quantity of product i in time t;

ULS the unit load size,

PCi the container capacity of product i;

Pa the probability of availability of machine hours based on the previous machine history;

## Worked Hybrid Example

The work by Byrne and Bakir (1999), Kim and Kim (2001) and Byrne and Hossain (2005) of working towards having an analytical model feed a simulation, which is then fed to an analytical model (Type V) can lead to enhanced knowledge networks. The following follows their work using a LP interfaced with a DES model (in Sigma).

### LP Model

The original work by Byrne and Bakir (1999) assumed that backordering did not exist (any orders unfilled in a period were a net loss), this was modified by Kim and Kim (2001) and Byrne and Hossain (2005) to allow backorders to be filled in later periods (except period 3 orders). The system modeled for this exercise used the original Byrne and Bakir (1999) LP but allowed for backordering.

$$Min \sum_{t=1}^{T} \sum_{i=1}^{N} \left( c_{it} X_{it} + h_{it} I_{it} + \pi_{it} B_{it} \right)$$

$$subject\_to$$

$$\sum_{i=1}^{N} a_{ik} X_{it} \le C_{kt}$$

$$I_{it} - B_{it} = I_{it-1} - B_{it-1} + X_{it} - d_{it}$$

$$all\_\mathrm{var}iable \ge 0$$

Where:

Indices

i index of product, i =1,2,…,N, where N is the number of products.

p,t indices of planning periond, p=1,2,…,T and t=1,2,…,T where T is the planning horizon

k machine type, k=1,2,…,K, where K is the number of machines.

Parameters

$a_{ik}$ processing time to produce a unit of product i at machine type k

$C_{kt}$ capacity (work-hours) of machine k in period t

$d_{it}$ demand for product i in period t

$c_{it}$ unit cost of producing product i in period t

$h_{it}$ inventory holding cost for product i in period t

$\pi_{it}$ backorder cost of product i in period t

Variables

$X_{it}$ quantity of product i produced in period t

$I_{it}$ inventory level of product i in period t

$B_{it}$ backorder level of product i in period t

Adjustments were made (according to the articles) to the unit capacity, but the specific unit capacities were never described. The number of machines per machine center was set at 1 each, the maximum available working time per period was set at 2400 minute (40 hours), and the unit production rate by product was given (table 4 above). Analysis of alternative adjustment methods determined that the maximum available working was adjusted (reduced).

**Simulation Model**

The work by Byrne and Bakir (1999), Kim and Kim (2001) and Byrne and Hossain (2005) stressed the LP model and various modifications to that LP to improve results. But they all glossed over the DES model and it's structure. In addition they each used a different SML and SMP.

The simulation model for this example was created and run in Sigma95. The graphic model is:

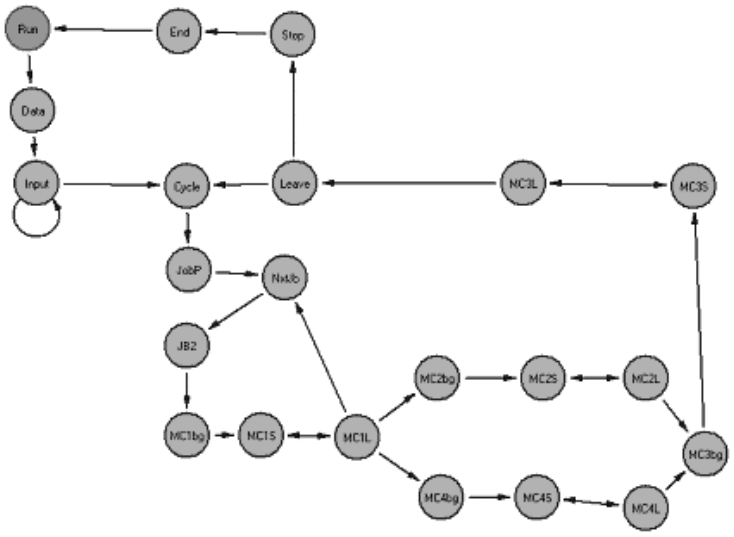



**Figure 21: Sigma Model for MPLP Problem**

The work discussed in Byrne and Bakir (1999), Kim and Kim (2001), and Byrne and Hossain (2005) did not describe the actual simulation to any extent. In particular, the actual simulation study was not discussed. As each of the alternative products takes a different amount of time to produce, alternative schedules could produce the same output with different production times, and different production outputs can be produced with similar productin times. The three products production time is shown in figure 12 below.

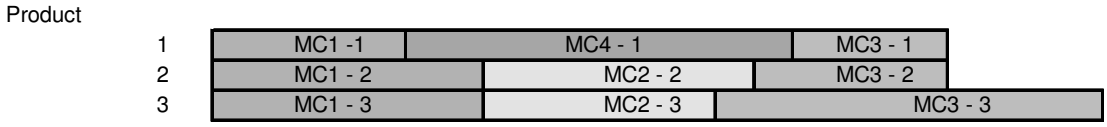



**Figure 22: Production Time by Product**

An initial look was performed on the impact of alternative production schedules. And it was noted that there was a time advantage to one versus another (timeline shown figure 13 below). To overcome this each simulation run was replicated 36 times, using different random number seeds, and the averages of these runs was used for the LP adjustment factors.

**Iterations**

A similar methodology as used by Byrne and Bakir (1999), Kim and Kim (2001), and Byrne and Hossain (2005) was followed, and a stopping point was determined to be when two iterations agreed closely.

An initial LP solution was found and that production level was input into the simulation model. The actual production and run times for each machine center were calculated. An adjustment factor for simulated times compared to full capacity times was calculated. This was used to adjust the LP capacities and another LP solution was obtained. The convergence of the LP models occurred at seven (7) iterations, similar to the work of Byrne and Bakir (1999).

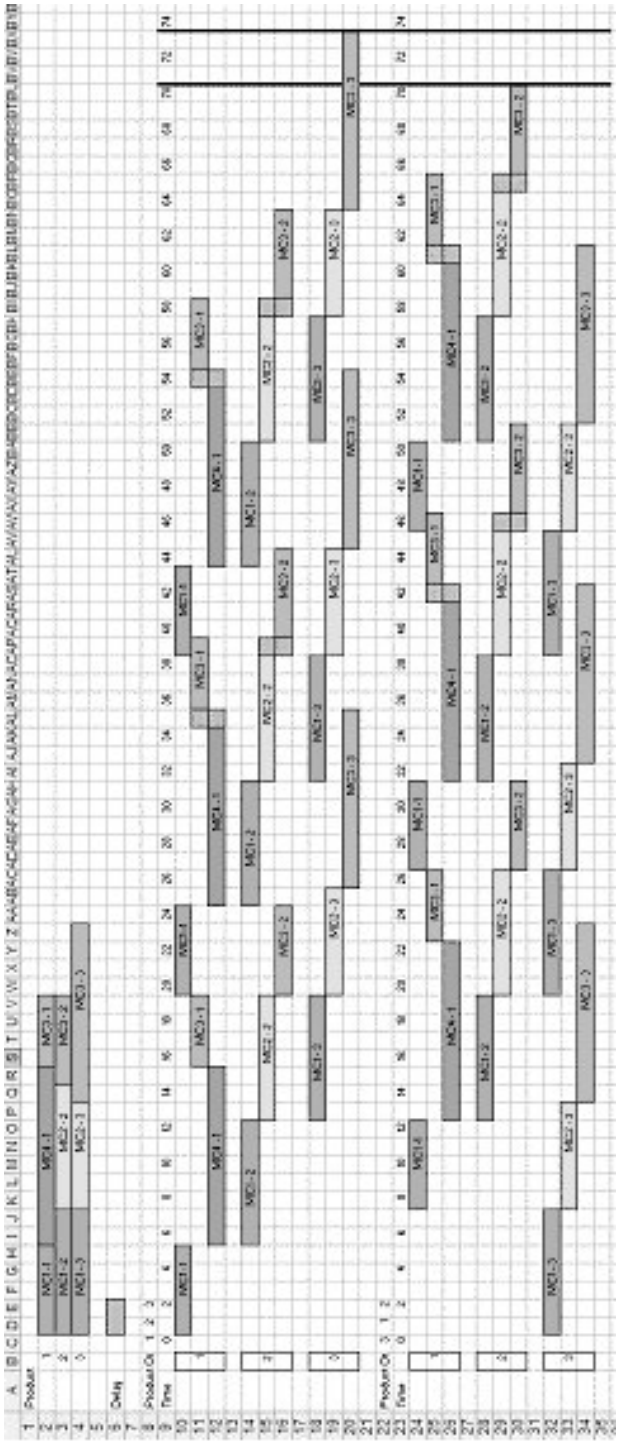


**Figure 23: Alternative Schedules**

**Table 24: Hybride Example Iteration Results**

| Iteration: | 1 | | | Periods (t) | | |
|---|---|---|---|---|---|---|
| | | Product (i) | 1 | 2 | 3 | Total |
| citXit | $141,750.00 | 1 | 151 | 123 | 160 | |
| hitlit | $355.00 | 2 | 111 | 129 | 104 | |
| pitBit | $71,500.00 | 3 | 124 | 126 | 124 | |
| Minimize | $213,605.00 | Total | 386 | 378 | 388 | 1152 |

| Iteration: | 2 | | | Periods (t) | | |
|---|---|---|---|---|---|---|
| | | Product (i) | 1 | 2 | 3 | Total |
| citXit | $138,250.00 | 1 | 135 | 136 | 164 | |
| hitlit | $600.00 | 2 | 120 | 129 | 96 | |
| pitBit | $101,800.00 | 3 | 119 | 110 | 115 | |
| Minimize | $240,650.00 | Total | 374 | 375 | 375 | 1124 |

| Iteration: | 3 | | | Periods (t) | | |
|---|---|---|---|---|---|---|
| | | Product (i) | 1 | 2 | 3 | Total |
| citXit | $136,100.00 | 1 | 126 | 149 | 160 | |
| hitlit | $660.00 | 2 | 122 | 127 | 85 | |
| pitBit | $114,750.00 | 3 | 120 | 100 | 120 | |
| Minimize | $251,510.00 | Total | 368 | 376 | 365 | 1109 |

| Iteration: | 4 | | | Periods (t) | | |
|---|---|---|---|---|---|---|
| | | Product (i) | 1 | 2 | 3 | Total |
| citXit | $135,350.00 | 1 | 114 | 162 | 159 | |
| hitlit | $715.00 | 2 | 123 | 127 | 74 | |
| pitBit | $120,100.00 | 3 | 125 | 91 | 130 | |
| Minimize | $256,165.00 | Total | 362 | 380 | 363 | 1105 |

| Iteration: | 5 | | | Periods (t) | | |
|---|---|---|---|---|---|---|
| | | Product (i) | 1 | 2 | 3 | Total |

| | | | | | | |
|---|---|---|---|---|---|---|
| citXit | $135,050.00 | 1 | 102 | 172 | 161 | |
| hitIit | $1,090.00 | 2 | 134 | 116 | 62 | |
| pitBit | $123,700.00 | 3 | 127 | 91 | 140 | |
| Minimize | $259,840.00 | Total | 363 | 379 | 363 | 1105 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Iteration: | 6 | | | Periods (t) | | |
| | | Product (i) | 1 | 2 | 3 | Total |
| citXit | $134,000.00 | 1 | 89 | 185 | 161 | |
| hitIit | $1,245.00 | 2 | 138 | 112 | 50 | |
| pitBit | $132,300.00 | 3 | 128 | 88 | 148 | |
| Minimize | $267,545.00 | Total | 355 | 385 | 359 | 1099 |

| | | | | | | |
|---|---|---|---|---|---|---|
| Iteration: | 7 | | | Periods (t) | | |
| | | Product (i) | 1 | 2 | 3 | Total |
| CitXit | $133,825.00 | 1 | 89 | 186 | 160 | |
| HitIit | $1,195.00 | 2 | 134 | 115 | 49 | |
| PitBit | $132,750.00 | 3 | 130 | 86 | 149 | |
| Minimize | $267,770.00 | Total | 353 | 387 | 358 | 1098 |

This type V, or hybrid system using a simple LP model can then lead to more advanced concepts with simulation into complex systems and knowledge networks.

## Simulation and Modeling for Complex systems

Almost all systems can be considered complex and adaptive. One of the most complex adaptive systems is human life. Human reproduction takes 8 to 9 months and considerable effort. A complex manufacturing assembly plant can be revamped in the same time frame. When modeling Complex (Adaptive) Systems the use of knowledge rules leads to a DSS that comes close to reasoning (Artificial Intelligence (AI)), or at least can solve simple problems and suggest alternative options (Knowledge Networks).

## Potential Research Areas

From this review some potential research areas have been identified in the general area of simulation in operations support. This area has several potential research possibilities. In should be noted that there is significant overlap between them.

## Operations Control

Many chemical process operations produce different products using the same equipment. They may have several operating units working independently or in combination. While the process equipment and manpower is relatively fixed, the actual materials and material flows can change depending on the product. In a multi-unit, multi-product semi-batch operation the optimized scheduling of component delivery can significantly impact production and quality. The production impact is fairly obvious due to delays from waiting for components being handled by shared resources. Similarly in a chemical process system, the delivery of the correct amount of material at the right time can impact quality.

Manufacturing operations are increasingly being impacted by higher material costs and increased efforts to achieve high first time quality levels, solving production upsets is critical. Simulation can show where bottlenecks and interferences exist.

An example is at Criterion Catalysts & Technologies (CC&T) Pittsburg Operation. Pittsburg produces over 16 different types of catalysts in three parallel and one series process. Normally the three parallel processes are producing different products. A portion of the raw material handling system is common for all three. To prevent cross contamination and to achieve a high first time quality it is necessary to assure that the system is clear before changing to another product for a different unit. Getting the wrong amount can lead to not meeting the product specifications. Delivering too much of a component, while also effecting quality, can have a significant cost impact. The use of a simulation model can help identify operational and control problems

.

## Model/DSS Reusability

Much work has been done on the tools to develop DSS, but equally important is having a DSS that is easy to use. In relating to V&V, one area that was mentioned in the GAO (Fosset, Harrison, Winthrop, and Gass, (1991) ) work was the number of models developed and never used. If the DSS is not easy to use and flexible, it will not be used. Also, simulation models are usually built to analyze a particular problem and optimized for that use. Work (analysis of execution rate above) has shown that adding additional features can impact the models execution. Changing the desired output of the model or its use can be a time consuming effort. Model reuse and or component based architecture can improve this and increase flexibility. Utilizing java based simulations modeling frameworks (such as Ptolemy II) can lead to faster revisions.

## Hybrid simulation in real world applications.

In the real world problems often have both a discrete and a continuous component; hybrid models cover the recognition that some applications have both a discrete and continuous nature. Tasks commonly thought of as discrete events (such as discharging a bin) have a continuous component. Actually bin discharge takes a finite and varying period of time (angle of tipping can change rate of discharge, etc.). Recognizing an inherent limitation of DES to handling continuous events, work towards a hybrid system, which combines discrete and continuous functions in one simulation run has seen some effort. This is either done by using a stochastic system to generate input for a DES or by including truly continuous aspects to a DES. Using a stochastic system to generate input is often a Decision Support System approach. Incorporating continuous functions in a DES can lead to a hybrid system. Going beyond a simple hybrid system can then lead to more advanced concepts with simulation into complex systems and knowledge networks.

Almost all systems can be considered complex and adaptive. One of the most complex adaptive systems is human life. Human reproduction takes 8 to 9 months and considerable effort. A complex manufacturing assembly plant can be revamped in the

same time frame. When modeling Complex (Adaptive) Systems the use of knowledge rules leads to a DSS that comes close to reasoning (Artificial Intelligence (AI)), or at least can solve simple problems and suggest alternative options (Knowledge Networks).

# Appendix A: Current (2007) Simulation Software

| | Software | Vendor |
|---|---|---|
| 1 | adevs | University of Arizona |
| 2 | ALPHA/Sim | BAE Systems/Alphatech |
| 3 | Amino | Open source |
| 4 | AnyLogic 6.0 | XJ Technologies |
| 5 | Arena | Rockwell Software |
| 6 | AutoMod Suite | Brooks Automation |
| 7 | AWESIM | INFOR |
| 8 | C++SIM | Department of Computing Science, University of Newcastle upon Tyne |
| 9 | CD++ | Wainer, Carelton Univeristy, |
| 10 | Chi | Systems Engineering Group of the Eindhoven University of Technology, Department of Mechanical Engineering |
| 11 | CPN Tools | University of Aarhus |
| 12 | CSIM 19/CSIM java | Mesquite Software |
| 13 | CYclone | University of Michigan CEM |
| 14 | DEMOS | various |
| 15 | DES | MelbourneTech |
| 16 | Design/CPN | University of Aarhus |
| 17 | DESMO-J | University of Hamburg |
| 18 | DEVS/HLA | University of Arizona |
| 19 | DEVS-JAVA | University of Arizona |
| 20 | DEx TK | dextk.org |
| 21 | Dymola | Dassault Systèmes/Dynasim |
| 22 | eM-Plant | Tecnomatix Technologies Inc. |
| 23 | Enterprise Dynamics | Incontrol Enterprise Dynamics (US: Production Modeling Corporation) |
| 24 | Extend | Imagine That, Inc. |
| 25 | EZSIM | Behrokh Khoshnevis |
| 26 | Flexsim | Flexsim Software Products, Inc. |
| 27 | ForeTell-DSS | DecisionPath, Inc. |
| 28 | GPSS World for Windows | Minuteman Software |
| 29 | GPSS/H | Wolverine |
| 30 | GPSS/PC | Minuteman Software |
| 31 | JavaSim | University of Newcastle upon Tyne |
| 32 | JiST | Cornell University |
| 33 | KAMELEON | PE International |
| 34 | Lean-Modeler | Visual8 |
| 35 | MAST | CMS Research Inc |

| | | |
|---|---|---|
| 36 | Micro Saint | Micro Analysis & Design, Inc. |
| 37 | Modline | SIMULOG |
| 38 | MODSIM III | CACI - no longer supported |
| 39 | NeST | Columbia University |
| 40 | nsnam (was: Berkeley NS) | UCB/LBNL/VINT |
| 41 | OMNeT++ | Omnetpp Org (A. Vargas) |
| 42 | OOPM/MOOSE /SIMPACK | P. A. Fishwick, Univeristy of Florida |
| 43 | OpEMCSS | JOHN R. CLYMER,CSU Fullerton |
| 44 | PARSEC / MAISIE | UCLA Parallel Computing Laboratory |
| 45 | PASION Simulation System | Stanislaw Raczynski |
| 46 | POSES++ | GPC |
| 47 | ProModel | ProModel Solutions |
| 48 | Proplanner | Proplanner |
| 49 | PSIM3C PSIMJ | Jose M. Garrido |
| 50 | PSM++ Simulation System | Stanislaw Raczynski |
| 51 | Ptolemy II | CHESS EECS Dept University of California at Berkeley |
| 52 | QNAP2 | SIMULOG |
| 53 | QUEST | Delmia |
| 54 | Real 5.0 | S. Keshav, Cornell University |
| 55 | RePast | Open source |
| 56 | Sage | Highpoint Software Systems, LLC |
| 57 | SansGUI | ProtoDesign Inc. |
| 58 | SDI Industry | Simulation Dynamics, Inc. |
| 59 | Shift | California Path |
| 60 | ShowFlow | Showflow US |
| 61 | SIGMA | Custom Simulations |
| 62 | Silk | ThreadTec |
| 63 | SIMAN | Rockwell Software |
| 64 | SIMAS II | CIMPACT |
| 65 | SimCAD Pro | CreateASoft, Inc. |
| 66 | simjava | University of Edinburgh |
| 67 | Simkit | Moves/NPS |
| 68 | SimPack | P. A. Fishwick, Univeristy of Florida |
| 69 | Simple 1 | Sierra Simulations & Software |
| 70 | SIMPLE++ | Tecnomatix Technologies Inc. |
| 71 | SIMPLORER | ANSOFT |
| 72 | SIMPROCESS | CACI Products Company |
| 73 | SimPy | SimPy Developer Team |
| 74 | SIMSCRIPT II.5 | CACI Products Company |

| | | |
|---|---|---|
| 75 | SIMUL8 | SIMUL8 Corporation |
| 76 | SLX | Wolverine |
| 77 | Speedes | METRON, INC., HIGH PERFORMANCE COMPUTING DIVISION |
| 78 | Stoboscope | University of Michigan CEM |
| 79 | Supply Chain Guru | LLamasoft |
| 80 | Swarm | Swarm Development Group |
| 81 | Systemflow 3D Animator | Systemflow Simulations, Inc. |
| 82 | Technomatix | UGS |
| 83 | Ventsim | Ventsim |
| 84 | Visual Simulation Environment (VSE) | Orca Computer, Inc. |
| 85 | Visual SLAM | INFOR |
| 86 | WebGPSS | FLUX Software Engineering |
| 87 | WITNESS 2007 | Lanner Group |

# Appendix B: Simulation Software Evaluation I

The following are taken from Hlupic, Irani, and Paul (1999)

| General features | | | | |
|---|---|---|---|---|
| 1 | Type of package | Data driven simulator | Data driven w/ some programming capability | Simulation language |
| 2 | Type of simulation | Discrete event | Continuous | Both |
| 3 | Purpose | General purpose | Manufacturing oriented | Other special purpose |
| 4 | Terminology | General terminology | Application specific terminology | |
| 5 | Modelling approach | Process based | | |
| | | Activity based | | |
| | | Event based | | |
| | | Three phase | | |
| | | Combination | | |
| 6 | Formal logic | Required | Not Required | |
| 7 | Representativeness of models | High | Medium | Low |
| 8 | Ease of conceptualisation of simulation logic | | | |
| 9 | Modelling transparency | High | Medium | Low |
| 10 | Hierarchical model building | Yes | No | |
| 11 | Run-time applications | Yes | No | |
| 12 | Conceptual model generator | Yes | No | |
| 13 | Versions of software for different operating systems | UNIX | | |
| | | Windows 3.1 | | |
| | | Windows 95 | | |
| | | Windows NT | | |
| | | OS/2 | | |
| 14 | The length of entity name | Long (e.g. unlimited) | Medium | Short (<=3 characters) |
| 15 | Entity name | User defined | System defined | |
| 16 | Experience required for software use | None | Some | Substantial |

| | | | | |
|---|---|---|---|---|
| 17 | Formal education in simulation required for software use | None | Some | Substantial |
| 18 | User friendliness | High | Medium | Low |
| 19 | Ease of learning | Yes | No | |
| 20 | Ease of using | Yes | No | |
| 21 | Initialisation | Yes | No | |
| 22 | Specification of time units | Yes | No | |
| 23 | Integration of operations (data gathering, simulation model development, output data analysis) | Yes | No | |
| 24 | Real-time simulation models | Yes | No | |
| 25 | Distributed simulation on network environment | Yes | No | |
| 26 | Specification of length measures | Yes | No | |
| | | | | |
| Visual aspects | | | | |
| 1 | Animation | Yes | No | |
| 2 | Type of animation | Full animation | Semi-animation (state-to-state) | |
| 3 | Timing of animation | Concurrent animation | Post-processed animation | |
| 4 | Type of graphical display | Icons | Symbols | Characters |
| 5 | 3D graphics | Yes | No | |
| 6 | Integrity of graphics | Integrated to the package | Separate | |
| 7 | Animation layout development | Concurrent with model development | | |
| | | Before model development | | |
| | | After model development | | |
| | | Flexible | | |
| 8 | Multiple screen layout | Yes | No | |
| 9 | Facility for customising the view of the model | Yes | No | |
| 10 | Playback mode | Yes | No | |
| 11 | Importing graphics and multimedia elements | CAD packages | Media Control Interface | Bitmap |
| 12 | Animation with visual clock | Yes | No | |
| 13 | Icon editor | Yes | No | |
| 14 | Screen editor | Yes | No | |
| 15 | Ease of icon development | Yes | No | |

| | | | | |
|---|---|---|---|---|
| 16 | Ease of using screen editor | Yes | No | |
| 17 | Types of icons | Bit mapped | Pixel based | |
| 18 | Icon library | Yes | No | |
| 19 | Merging icon files | Yes | No | |
| 20 | Resizing of icons | Yes | No | |
| 21 | Rotating of icons | Yes | No | |
| 22 | Changing the colour of the icons | Yes | No | |
| 23 | Zoom function | Yes | No | |
| 24 | Panning | Yes | No | |
| 25 | Switching on/off the graphic | Yes | No | |
| 26 | Switching between screens | Yes | No | |
| 27 | Switching between character and icon graphics | Yes | No | |
| 28 | Print screen facility | Yes | No | |
| 29 | Virtual screen | Yes | No | |
| 30 | Indication of the element status | Yes | No | |
| 31 | Changing the colour of the element status display | Yes | No | |
| 32 | Limitation on number of displayed icons | Yes | No | |
| 33 | Number of icons stored in icon library | Large | Medium | Small |
| 34 | Change of icons during simulation | Yes | No | |
| 35 | Icons with multiple colours | Yes | No | |
| 36 | Virtual reality features | Yes | No | |
| 37 | Easy copying of icons | Yes | No | |
| | | | | |
| Coding aspects | | | | |
| 1 | Programming flexibility | Yes | No | |
| 2 | Program generator | Yes | No | |
| 3 | Access to source code | Yes | No | |
| 4 | Readability of source code | High | Medium | Low |
| 5 | Readability of added code | High | Medium | Low |
| 6 | Self-documentation of added code | High | Medium | Low |
| 7 | Precision of added code | High | Medium | Low |
| 8 | Comprehensiveness of added code | High | Medium | Low |
| 9 | Link to a Lower language | Yes | No | |
| 10 | Data storage, retrieval and manipulation facilities | Yes | No | |
| 11 | Quality of data storage, retrieval and manipulation facilities | High | Medium | Low |
| 12 | Built-in functions | Yes | No | |
| 13 | User functions | Yes | No | |

| | | | | |
|---|---|---|---|---|
| 14 | Global variables | Yes | No | |
| 15 | Names of functions, variables and attributes: defined by: | User | System | |
| 16 | Writing comments for logical elements | Yes | No | |
| 17 | Type of time variable | Real | Integer | |
| 18 | Type of translation | Compilation | Interpretation | |
| 19 | Text/code manipulation | Yes | No | |
| 20 | Length of the lines in coding editor | Large | Medium | Small |
| 21 | Support of programming concepts | Yes | No | |
| 22 | Quality of the support for programming concepts | High | Medium | Low |
| 23 | Interface to user written programs | | | |
| 24 | Object oriented programming concepts | Yes | No | |
| | | | | |
| Efficiency | | | | |
| 1 | Robustness | High | Medium | Low |
| 2 | Level of detail | High | Medium | Low |
| 3 | Number of elements in the model | Large | Medium | Small |
| 4 | Model reusability | Yes | No | |
| 5 | Model status saving | Yes | No | |
| 6 | Automatic saving | Yes | No | |
| 7 | Interaction | Yes | No | |
| 8 | Adaptability to model changes | High | Medium | Low |
| 9 | Multitasking (i.e. Performing several tasks at the same time) | Yes | No | |
| 10 | Model chaining (i.e. linking outputs from different models) | Yes | No | |
| 11 | Exit to the operating system within the package | Yes | No | |
| 12 | Compilation time | Long | Medium | Short |
| 13 | Model execution time | Long | Medium | Short |
| 14 | Case sensitivity | Yes | No | |
| 15 | Conversion of numbers (real vs. integer) | Yes | No | |
| 16 | Various queuing policies | Yes | No | |
| 17 | Number of queuing policies | Large | Medium | Small |
| 18 | Time scale for model building | Large | Medium | Small |
| 19 | Reliability | High | Medium | Small |
| 20 | Pre-existing generic models | Yes | No | |
| 21 | Merging of models | Yes | No | |
| 22 | Editing partially developed models | Yes | No | |

| | | | | |
|---|---|---|---|---|
| 23 | Automatic model building | Yes | No | |
| 24 | Ease of model editing | Yes | No | |
| 25 | Interactive handling of parameters during experimentation | Yes | No | |
| 26 | Specification of part flow by amouse | Yes | No | |
| | | | | |
| Modelling assistance | | | | |
| 1 | Prompting | Yes | No | |
| 2 | Quality of prompting | High | Medium | Small |
| 3 | Modularity | Yes | No | |
| 4 | Model and data separation | Yes | No | |
| 5 | Facility for designing reusable user defined elements | Yes | No | |
| 6 | Libraries and templates of simulation objects | Yes | No | |
| 7 | Warning messages which for operations which affect the model file (e.g. overwriting, closing file not saved) | Yes | No | |
| 8 | Warning messages for operations which affect model currently developed | Yes | No | |
| 9 | Context sensitive prompt to facilitate model development | Yes | No | |
| 10 | Undo/redo commands | Yes | No | |
| 11 | Automatic connection between elements | Yes | No | |
| 12 | Use of mouse | Yes | No | |
| 13 | One-line help | Yes | No | |
| 14 | Quality of on-line help | High | Medium | Small |
| 15 | Search facilities within help | Yes | No | |
| 16 | Help on system messages | Yes | No | |
| 17 | Printing help text | Yes | No | |
| 18 | Documentation notes for inserting comments concurrently with model development | Yes | No | |
| 19 | Quality of facility for documentation notes | High | Medium | Small |
| 20 | Text editor as integral part of the package | Yes | No | |
| 21 | Automatic editing of data | Yes | No | |
| | | | | |
| Testability | | | | |
| 1 | Logic checks | Yes | No | |
| 2 | Interactive error messages | Yes | No | |

| | | | | |
|---|---|---|---|---|
| 3 | Quality of error messages | High | Medium | Small |
| 4 | Moment of error diagnosis | Model entry | | |
| | | Compilation | | |
| | | Model execution | | |
| | | Combination | | |
| 5 | Ease of debugging | Yes | No | |
| 6 | Display of function values | Yes | No | |
| 7 | Display of attributes | Yes | No | |
| 8 | Access to attributes | Yes | No | |
| 9 | Display of variables | Yes | No | |
| 10 | Display of element's state | Yes | No | |
| 11 | Dynamic display of capacity | Yes | No | |
| 12 | Display of the workflow path | Yes | No | |
| 13 | Display of events on the screen | Yes | No | |
| 14 | Display of part position within element | Yes | No | |
| 15 | Facility for immediate user actions | Yes | No | |
| 16 | List files (list of model entities and parameters) | Yes | No | |
| 17 | Echo | Yes | No | |
| 18 | Trace files (showing events and entity status) | Yes | No | |
| 19 | Explode function (showing a state of an element) | Yes | No | |
| 20 | List of used elements | Yes | No | |
| 21 | Backward clock | Yes | No | |
| 22 | Step function (event to event jumping) | Yes | No | |
| 23 | FLow analysis | Yes | No | |
| 24 | Interactive debugger | Yes | No | |
| 25 | Display of parts flow tracking record collected during simulation run | Yes | No | |
| 26 | Audible alarms | Yes | No | |
| 27 | Rejection of illegal inputs | Yes | No | |
| | | | | |
| Software compatibility | | | | |
| 1 | Integration with spreadsheet packages | Yes | No | |
| 2 | Integration with statistical packages | Yes | No | |
| 3 | Integration with word processors | Yes | No | |
| 4 | Integration with computer-aided design software | Yes | No | |
| 5 | Integration with database management system | Yes | No | |

| | | | | |
|---|---|---|---|---|
| 6 | Integration with expert systems | Yes | No | |
| 7 | Integration with manufacturing requirements planning software | Yes | No | |
| 8 | Integration with scheduling software | Yes | No | |
| | | | | |
| Input/output | | | | |
| 1 | Menu driven interface | Yes | No | |
| 2 | Pull down menus | Yes | No | |
| 3 | Type of menu selection | By mouse | By keys | Other |
| 4 | Selection buttons | Yes | No | |
| 5 | Dialogue boxes | Yes | No | |
| 6 | Multiple inputs | Yes | No | |
| 7 | Model input | Interactive | Batch mode | |
| 8 | Database maintenance for input/output | Yes | No | |
| 9 | Multiple outputs | Yes | No | |
| 10 | General output reports | Yes | No | |
| 11 | Static graphical output | Yes | No | |
| 12 | Dynamic graphical output | Yes | No | |
| 13 | Types of graphical display | Bar graphs | | |
| | | Histograms | | |
| | | Level graphs | | |
| | | Pie charts | | |
| | | Line graphs | | |
| | | Scatter diagrams | | |
| | | Time series | | |
| | | Area graphs | | |
| 14 | User defined output | Yes | No | |
| 15 | Automatic rescaling of histograms and time series | Yes | No | |
| 16 | Quality of output reports | High | Medium | Low |
| 17 | Understandability of output reports | High | Medium | Low |
| 18 | Periodic output of simulation results | Yes | No | |
| 19 | Availability of results before end of simulation | Yes | No | |
| 20 | Input data reading from files | Yes | No | |
| 21 | Writing reports to files | Yes | No | |
| 22 | Writing reports to printer | Yes | No | |
| 23 | Writing reports to plotter | Yes | No | |
| 24 | Snapshot reports | Yes | No | |
| 25 | Summary reports for multiple runs | Yes | No | |

| | | | | |
|---|---|---|---|---|
| Experimentation facilities | | | | |
| 1 | Automatic batch run | Yes | No | |
| 2 | Warm-up period | Yes | No | |
| 3 | Independent replications of experiments | Yes | No | |
| 4 | Re-initialisation | Yes | No | |
| 5 | Re-start from non-empty state | Yes | No | |
| 6 | Breakpoints | Yes | No | |
| 7 | Speed adjustment | Yes | No | |
| 8 | Experimental design capability | Yes | No | |
| 9 | Quality of experimental design facility | | | |
| 10 | Accuracy check | Yes | No | |
| 11 | Automatic determination of run length | Yes | No | |
| | | | | |
| Statistical facilities | | | | |
| 1 | Theoretical statistical distributions | Yes | No | |
| 2 | Number of theoretical statistical distributions | Large | Medium | Small |
| 3 | User-defined distributions | Yes | No | |
| 4 | Random number streams | Yes | No | |
| 5 | Number of different random number streams | Large | Medium | Small |
| 6 | User-specified seeds of random number streams | Yes | No | |
| 7 | Antithetic sampling | Yes | No | |
| 8 | Distribution fitting | Yes | No | |
| 9 | Goodness-of-fit tests | Yes | No | |
| 10 | Output data analysis | Yes | No | |
| 11 | Quality of data analysis facility | High | Medium | Low |
| 12 | Confidence intervals | Yes | No | |
| | | | | |
| User support | | | | |
| 1 | Documentation (manuals) | Yes | No | |
| 2 | Quality of documentation | High | Medium | Low |
| 3 | Reference card | Yes | No | |
| 4 | Glossary | Yes | No | |
| 5 | Technical and promotional information material (e-mail, bulletin board) | Yes | No | |
| 6 | Discussion groups on the Internet | Yes | No | |
| 7 | Lecturer's guide for educational licences | Yes | No | |

| | | | | |
|---|---|---|---|---|
| 8 | Demo disks | Yes | No | |
| 9 | Tutorial | Yes | No | |
| 10 | Training course (basic, advanced) | Yes | No | |
| 11 | Custom tailored training course | Yes | No | |
| 12 | Duration of training courses | Long | Medium | Short |
| 13 | Frequency of training courses | Frequent | Not frequent | |
| 14 | Demo models | Yes | No | |
| 15 | Help-line | Yes | No | |
| 16 | User group meetings | Yes | No | |
| 17 | Frequency of user group meetings | Frequent | Not frequent | |
| 18 | Newsletter | Yes | No | |
| 19 | Package maintenance | Yes | No | |
| 20 | Consultancy | Yes | No | |
| | | | | |
| Financial and technical features | | | | |
| 1 | Portability | Yes | No | |
| 2 | File conversion | Yes | No | |
| 3 | Price | High | Medium | Low |
| 4 | Installation costs | High | Medium | Low |
| 5 | Ease of installation | Easy | Not Easy | |
| 6 | Hardware requirements | High | Medium | Low |
| 7 | Availability of package on standard hardware | Yes | No | |
| 8 | Availability of package on standard operating systems | Yes | No | |
| 9 | Version of software for network | Yes | No | |
| 10 | Virtual memory facility | Yes | No | |
| 11 | Security device | Yes | No | |
| 12 | Free software trials | Yes | No | |
| 13 | Free technical support | Yes | No | |
| 14 | Types of contracts available | Many | Not many | |
| 15 | Educational discount | Yes | No | |
| 16 | Quantity discount | Yes | No | |
| 17 | Life cycle maintenance costs | High | Medium | Low |
| 18 | Price of training course | High | Medium | Low |
| 19 | Consultancy fees | High | Medium | Low |
| 20 | Frequency of update | Frequent | Not frequent | |
| 21 | Comprehensiveness of update | High | Medium | Low |
| | | | | |
| Pedigree | | | | |
| 1 | Age | New | Medium | Old |
| 2 | Genealogy - - - - - - - - | | | |

| | | | | |
|---|---|---|---|---|
| 3 | Spread | High | Medium | Low |
| 4 | Success | High | Medium | Low |
| 5 | Availability of references | High | Medium | Low |
| 6 | Software maturity | High | Medium | Low |
| 7 | Reputation of supplier | High | Medium | Low |
| 8 | Sources of information about the package | Other users | | |
| | | Literature | | |
| | | Supplier | | |
| | | Demonstration | | |
| | | Combination of several sources | | |

Following Hlupic, Irani, and Paul (1999)

# Appendix C: Simulation Software Evaluation II

Evaluation framework for software process simulation models' quality from Ahmed, Hall, and Wernick (2003).

| Quality Aspect | Goal | Model Properties | Means |
|---|---|---|---|
| Syntactic Quality | Syntactic Correctness | Defined Syntax | 1. Manual checking of the influence model (the diagram must include only factors and links between factors and there must be no circular links) |
| | | | 2. Manual checking of algebraic equations (to ensure that the equations are well formed and that intermediate values are not required before they are calculated) |
| | | | 3. Checking that each diagram elements appears in an equation and that inputs and outputs are consistent with the direction of the links |
| | | | 4. Testing the algebraic equations with selected predefined values |
| Semantic Quality | Feasible validity Feasible completeness | Traceability to domain | 1. Inspection aimed at checking the model includes: |
| | | | • Definition |
| | | | o Verification of model inputs and outputs |
| | | | o Qualitative assessment of reasonableness of model outputs |
| | | | • Detail Information |
| | | | • Scope Information |
| | | | • Other domain features related to the bidding process, problems associated with the bidding process and the intended solutions to the problems. (this has been replaced by face validity) |
| | | | • Face validity |
| | | | • Objectivity |

| | | | |
|---|---|---|---|
| | | | o Reference Only Objectivity |
| | | | o Subjective objectivity |
| | | | o Partial Objectivity |
| | | | o Completely Objective |
| | | | 2. Sensitivity analysis – identify unnecessary features |
| | | | 3. Consistency checking (aimed at ensuring the model is internally consistent) |
| Pragmatic Quality | Feasible comprehension | Understanding | 1. Means to enable comprehension including Visualization, Explanation Filtering. |
| | | Ability to use (user properties) | 2. Means to assess comprehension for example empirical study of understanding achieved by audience group (interviews or self-administered questionnaires) |
| | Feasible understandability | Structured expressive economy | Documenting guidelines and standards covering format and content. Contents standard should define elements that must be included in model documentation such as |
| | | | • Interface definition (Ease of use) |
| | | | • Model input-output explanation (Constructiveness) |
| Test Quality | Feasible test coverage | Executability | 1. Simulation studies based on pre-defined scenarios (Fidelity) |
| | | | 2. Simulation studies related to input value manipulation (Sensitivity and Stability) |
| | | | 3. Post-mortem validation |
| | | | 4. Verification of inputs and outputs |
| | | | 5. Qualitative assessment of reasonableness of outputs |
| | | | 6. Special case validity |
| Maintainability | Practical Maintenance | Structuredness | 1. Means to enhance maintainability |
| | | Modularity | 2. Conformance with Standards |
| | | Complexity | 3. Quality of documentation |
| | | | 4. Understandability |

| | | | |
|---|---|---|---|
| | | | 5. Parsimony |

Model's value evaluation:

| Evaluation Aspect | Goal | Model Properties | Means |
|---|---|---|---|
| Value | Practical utility | | 1. Means to enable model use including appropriate user interface design (Ease of use), use manuals and training. |
| | | | 2. Assessment of goals achievement |
| | | | 3. Means to evaluate model value for example empirical study of model users view of using the model (experiment, interviews or self-administered questionaire) |

# Appendix D: Modeling Package Evaluation

| | | | | Modeling Package | | | |
|---|---|---|---|---|---|---|---|
| CATEGORY Element | | Description | Factor | Arena | Extend | Sigma | Ptolemy II |
| A. Modeling Environment | | | | | | | |
| A | 1 | Model development environment provided – to support modeling process. | 1 | 4 | 4 | 4 | 5 |
| A | 2 | Alternative model creation methods | 1 | 1 | 1 | 2 | 5 |
| A | 3 | Visual (drag and drop) modeling | 1 | 5 | 5 | 5 | 5 |
| A | 4 | Degree of detail – the level of description – controllable within the model | 1 | 2 | 4 | 4 | 5 |
| A | 5 | Text editor as integral part of the package | 1 | 5 | 2 | 5 | 3 |
| A | 6 | Accessability of model code | 1 | 2 | 2 | 5 | 5 |
| A | 7 | Readibility of basic model code | 1 | 2 | 2 | 3 | 4 |
| A | 8 | Structured – to guide user in model development. | 1 | 4 | 4 | 2 | 4 |
| A | 9 | Animation – model animation provided without burden to the modeler. | 1 | 4 | 4 | 4 | 3 |
| A | 10 | Virtual reality features | 1 | 0 | 0 | 0 | 0 |
| | | CATEGORY A TOTAL Maximum = | 50 | 29 | 28 | 34 | 39 |
| B. Model Documentation and Structure | | | | | | | |
| B | 1 | Model description ranges from a very high to a very low level. | 1 | 5 | 5 | 3 | 5 |
| B | 2 | Operating system independence | 1 | 1 | 1 | 1 | 5 |
| B | 3 | Computer architecture independence – sequential, parallel and distributed | 1 | 1 | 1 | 1 | 5 |
| B | 4 | Model reuse – support a model database for component reuse. | 1 | 4 | 4 | 2 | 4 |
| B | 5 | Hierarchical capability – to facilitate the modeling of complex systems. | 1 | 5 | 5 | 2 | 5 |
| B | 6 | Model chaining (i.e. linking outputs from different models) | 1 | 2 | 2 | 1 | 5 |
| B | 7 | Alternative worldviews | 1 | 1 | 1 | 2 | 5 |
| B | 8 | Ease of communication – conceptual model(s) are easy to communicate to others. | 1 | 3 | 3 | 2 | 4 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| B | 9 | Efficient translation to executable form | 1 | 3 | 3 | 4 | 4 |
| B | 10 | Compatibility with, extant simulation programming languages | 1 | 2 | 2 | 1 | 3 |
| | | CATEGORY B TOTAL Maximum = | 50 | 27 | 27 | 19 | 45 |
| C. Verification & Validation | | | | | | | |
| C | 1 | Facilitates the verification and verification of simulation models. | 1 | 4 | 4 | 3 | 3 |
| C | 2 | Ease of model validation – supports both conceptual and operational validity. | 1 | 3 | 3 | 3 | 3 |
| C | 3 | Provides interactive model checking | 1 | 3 | 4 | 4 | 4 |
| C | 4 | Quality of error messages is high | 1 | 3 | 3 | 2 | 2 |
| C | 5 | Trace file provided (showing events and entity status) | 1 | 1 | 1 | 1 | 4 |
| | | CATEGORY C TOTAL Maximum = | 25 | 14 | 15 | 13 | 16 |
| D. Experimentation facilities | | | | | | | |
| D | 1 | Automatic batch run | 1 | 3 | 3 | 5 | 3 |
| D | 2 | Warm-up period | 1 | 5 | 5 | 2 | 3 |
| D | 3 | Independent replications of experiments | 1 | 5 | 5 | 4 | 4 |
| D | 4 | Re-initialisation | 1 | 5 | 5 | 4 | 2 |
| D | 5 | Re-start from non-empty state | 1 | 5 | 4 | 4 | 4 |
| D | 6 | Breakpoints | 1 | 5 | 4 | 3 | 4 |
| D | 7 | Speed adjustment | 1 | 5 | 4 | 4 | 5 |
| D | 8 | Experimental design capability | 1 | 5 | 4 | 3 | 3 |
| D | 9 | Accuracy check | 1 | 5 | 4 | 3 | 2 |
| D | 10 | Automatic determination of run length | 1 | 5 | 4 | 4 | 3 |
| | | CATEGORY D TOTAL Maximum = | 50 | 48 | 42 | 36 | 33 |
| E. Statistical facilities | | | | | | | |
| E | 1 | Alternative statistical distributions | 1 | 4 | 4 | 5 | 5 |
| E | 2 | User-defined distributions | 1 | 2 | 2 | 2 | 5 |
| E | 3 | Random number streams | 1 | 2 | 2 | 2 | 5 |
| E | 4 | User-specified seeds of random number streams | 1 | 5 | 5 | 5 | 3 |
| E | 5 | Antithetic sampling | 1 | 4 | 3 | 2 | 4 |
| E | 6 | Distribution fitting | 1 | 4 | 4 | 3 | 2 |
| E | 7 | Goodness-of-fit tests | 1 | 5 | 5 | 2 | 2 |
| E | 8 | Output data analysis | 1 | 5 | 5 | 3 | 3 |
| E | 9 | Quality of data analysis facility | 1 | 4 | 4 | 2 | 2 |
| E | 10 | Confidence intervals | 1 | 4 | 4 | 1 | 1 |
| | | CATEGORY E TOTAL Maximum = | 50 | 39 | 38 | 27 | 32 |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| F. | User support | | | | | | |
| | F 1 | User manuals (Documentation) | 1 | 5 | 3 | 4 | 3 |
| | F 2 | Quality of documentation | 1 | 4 | 3 | 3 | 4 |
| | F 3 | Technical and promotional information (e-mail, internet discussion groups) | 1 | 4 | 3 | 1 | 4 |
| | F 4 | Lecturer's guide for educational licences | 1 | 5 | 2 | 5 | 2 |
| | F 5 | Tutorial | 1 | 4 | 4 | 4 | 3 |
| | F 6 | Training course (basic, advanced) | 1 | 3 | 3 | 3 | 2 |
| | F 7 | Demo models | 1 | 4 | 4 | 4 | 4 |
| | F 8 | Package maintenance | 1 | 4 | 4 | 2 | 4 |
| | F 9 | Portability | 1 | 3 | 3 | 3 | 5 |
| | F 10 | File conversion | 1 | 3 | 3 | 4 | 4 |
| | | CATEGORY F TOTAL Maximum = | 50 | 39 | 32 | 33 | 35 |
| G. | Financial and technical features | | | | | | |
| | G 1 | Package Cost | 1 | 3 | 3 | 1 | 5 |
| | G 2 | Ease of installation | 1 | 4 | 4 | 4 | 4 |
| | G 3 | Life cycle maintenance costs | 1 | 2 | 2 | 2 | 4 |
| | G 4 | Hardware/system requirements | 1 | 4 | 4 | 4 | 5 |
| | G 5 | Frequency and comprehensiveness of update | 1 | 3 | 3 | 2 | 5 |
| | | CATEGORY G TOTAL Maximum = | 25 | 16 | 16 | 13 | 23 |
| | | TOTAL Maximum = | 300 | 212 | 198 | 175 | 223 |

Score = 0 to 5, with 0 = NO or not availabe, and 5 = YES or completely available
In conditions where it is a comparative ranking (e.g. Price) the most desirable would = 5
and the least desirable would = 0.