Reddy Y.V., and M.S. Fox, (1982), "Knowledge Representation in Organization Modeling and Simulation: A Detailed Example", Modeling and Simulation, Vol. 13, W.G. Vogt & M.H. Mickle (Eds.), pp. 685-691, Research Triangle Park NC: Instrument Society of America.

Knowledge Representation in Organization Modeling and Simulation: A Detailed Example[1]

Y.V.Reddy[2] & Mark S. Fox

The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

1. Introduction

A companion paper (Fox and Reddy, 1982) in this proceedings introduced the technique of Knowledge Representation and how it can be used to build Organizational Models. This paper illustrates the knowledge-based approach to modelling (hereafter referred to as KBS) by constructing a model of a PCBOARD production plant. Before a detailed presentation of the model, the concept of selective model instrumentation is discussed.

2. Model Instrumentation

The purpose of executing a simulation model is to gather data representing the performance of the system under study. Most simulation systems provide a standard set of statistics. KBS provides two approaches to data gathering and analysis. The first approach is similar to other systems. A library of routines are provided to do post simulation analysis. The second approach allows user-specified, selective instrumentation of models.

In the model, data gathering and analysis can be specified by identifying the schemata and slots in which the data resides. Rules can be associated with slots. These rules will be evaluated whenever the contents of a slot change. Queue sizes, processing times, etc. can be recorded by associating data gathering rules with the appropriate slots.

Figure 2-1 shows how **machine1** can be instrumented to determine how many orders were processesed. The *if-added* facet is filled with a rule which computes the required statistics. The contents of an *if-added* facet are evaluated whenever the value of a slot is altered. This modification will ensure that whenever the value of a slot STATE o

machine1 changes the rule **count-orders** will be executed.

```
{{ machine1
    { INSTANCE discrete-machine
      STATE:
          if-added: count-orders } }}
```

Figure 2-1: Data Gathering Rules

Model instrumentation allows the experimenter to selectively analyze the simulation. For example, if only **machine1** i of interest, then an *if-added* rule is placed in the appropriate slot of **machine1**. If all **machines** are to be analyzed, the an *if-added* rule is to be place in the slot of the **machine** schema, and all sub-types and instances of **machine** wi automatically inherit the rule, enabling data to be gathered for all of them. Selective instrumentation reduces the amour of computation devoted to data gathering and analysis, when the question to be answered is restricted in nature.

In a batch oriented simulation model, the only output that is available is a listing of the statistics collected during th model execution. In an interactive model, one can watch the simulation model as it is executing. KBS provides a displa which may be divided into a number of windows each displaying a different aspect of the model. For example one windo may be displaying an event-trace while another displays the history of a machine. In addition to watching the simulatio model as it is running, one can interact with the model and change parameters and/or specify new information to b displayed.

The specification of what to display can be handled in much the same way as the specification of statistics using th *if-added* facet. For example, if we want to display the history of **machine1** we can modify the **machine1** schema as i figure 2-2. The next section presents a detailed model of a PCBOARD manufacturing plant.

```
{{ machine1
   { INSTANCE discrete-machine
      HISTORY:
      If-added: display-history-rule } }}
```

Figure 2-2: Display Instrumentation

3. PCBOARD Model

The organization model described in this paper may be abstracted as:

> The factory consists of a number of areas (work areas, service areas, offices etc.) where different activities take place. Different machines are located in work areas and perform individual operations. A circuit-board is produced by performing a series of operations on the raw material. All work pieces waiting for an operation wait in a queue in front of a collection of machines or in a centralized in-process storage. The flow of work is controlled by the "operation-lineup" associated with the product being manufactured. The operation-lineup specifies the sequence of operations which will be used to schedule the next operation to be performed on the work-piece. The factory is configured such that "work-pieces" flow to various work-areas on a centralized conveyor system. If there is no space for a work-piece in a given work-area, it is stored in a centralized in-process storage from which it could be recalled when needed.

Construction of a model reflecting this level of abstraction involves two steps:

- Instantiation of generic schemata from the model library.
- Construction of special functions for scheduling; "priming" the model and specification of desired performance statistics.

The model described in this section consisted of 17 work-areas, 48 machines (both discrete and continuous), 34 queues and 30 different operations.

A **work-area** (figure 3-1) defines an area of the factory which contains machines and/or operators. Several types of operations may be performed in a given work area.

```
{{ work-area
   MACHINES:
   OPERATIONS:
   LOCAL-QUEUES:
   CAPACITY:
   CURRENT-CONTENTS:
   HISTORY: }}
```

Figure 3-1: **work-area** Schema

This schema is used to monitor the activity in a given work-area. For example, by attaching an **if-added** rule to the slot: **current-contents** we can perform any function such as display whenever a work-piece enters or leaves a work-area.

The **machine** schema is used to describe a machine which loads and unloads work-pieces to perform operations. A **discrete-machine** (figure 3-2) is generic to the model. Actual machines are subtypes, e.g., nc-drill, or instances of it. Schemata representing individual machines inherit their event rules from schemata at higher levels in the schema hierarchy. The **discrete-load-rule** (figure 3-3) associated with the LOAD slot of a **discrete-machine** specifies the conditions that have to be satisfied for loading to take place and the associated actions that follow.

The **operation** (figure 3-4) schema is used to specify details about individual operations. Operations can form a


```
{{ discrete-machine

   { IS-A machine
      LOAD: discrete-load-rule
      UNLOAD: discrete-unload-rule
      P-UNLOAD: p-unload-rule
      SETUP: setup-rule
      START: start-rule
      MAINTENANCE: maintenance-rule
      BREAKDOWN: breakdown-rule
      SERVICE-TIME:
         Restriction: (TYPE is-a FUNCTION)
      INPUT-RULE:
         Default: fcfs-input-rule
      LAST-MAINTAINED:
         Default: 0
      LAST-BREAKDOWN:
         Default: 0
      MTBF:
         Default: INFINITE
      CONTENTS:
      STATE:
         Restriction: (OR ready free busy stopped under-maintenance)
      STATISTICS: statistics-rule
      HISTORY: history          } }}
```

Figure 3-2: **discrete-machine** Schema

di-graph by linking them via their PREVIOUS-OPERATION and NEXT-OPERATION slots. For each order (i.e. a collectio

```
{{ DISCRETE-LOAD-RULE
   { INSTANCE rule
      IF:    state = free & input-source = not empty
      THEN:  ask input-rule &
             update contents slot &
             change state to busy &
             execute statistics-rule } }}
```

Figure 3-3: Load Rule for Discrete Machines

circuit boards), the **scheduler** assigns the first operation to be performed. The machine that can perform this opera is ascertained from the **operation** schema. Subsequent operations are determined by looking at the NEXT-OPERA slot of the current operation.

In addition to the instantiation of various schemata described above, we need a set of functions which are specific given model. These include functions for the "prime-event", "scheduling" and "service-time" computations. The pri event function specifies initial actions that should take place and scheduling function specifies the type of schedu used in the given model. Service-time function specifies how to compute the operation time for each machine function of the "work-order".

The database constructed using these schemata is used to collect performance statistics by selectively instrumen the model. The performance statistics collected include:

- Machine utilization
- Congestion measure for each work-area
- History of each machine

{{ operation
OPERATION-NAME:
PERFORMED-BY:

PRECONDITION:
POSTCONDITION:
CO-CONDITION:

PREVIOUS-OPERATION:
NEXT-OPERATION:
SUB-OPERATION:

INTERRUPT-RULE:
Comment: (What to do if this is interrupted) }}

Figure 3-4: Operation Schema

- Current production

A number of facilities are provided for monitoring the progress of the simulation, and for performing queries of machine and order status. Figure 3-5 shows the display screen before starting the simulation. The display screen is divided into three parts. The top window shows the available commands. The middle window shows the current default settings such as which order will be tracked. These defaults can be changed using the parameters command shown in the top window. The db-query command will permit perusal of the model database. The start-sim command will start simulation execution. The bottom window is used for command entry. At any stage a carriage return will provide help, and exit command will return the program to the previous level in the command hierarchy.

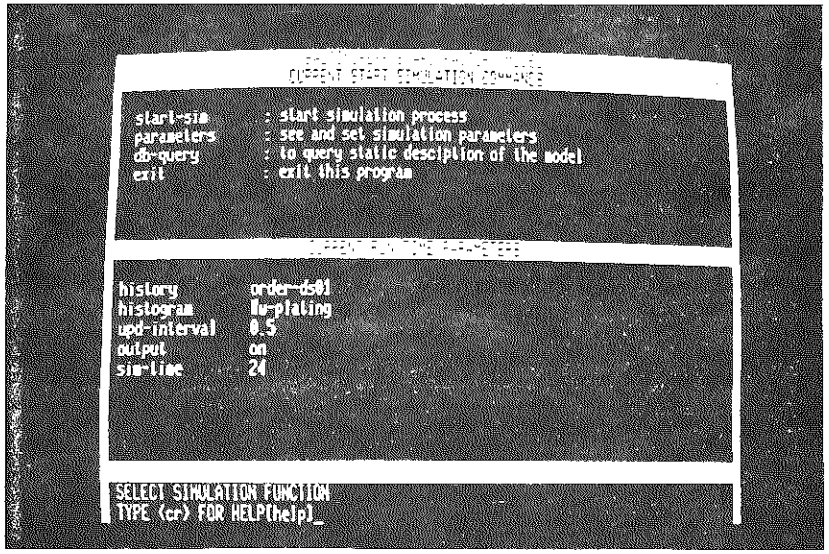

Figure 3-5: Initialization of Simulation Display

Figure 3-6 shows a snap-shot of simulation. In this the display is divided into six windows. The top left window shows the applicable commands. The top right window provides an event trace. The bottom left window provides the history of a selected order, i.e., the history of loads and unloads of order-ds01. The bottom right window provides a histogram of traffic in the selected work area. The bottom most window is used for selecting commands.

Figure 3-7 shows the display screen when a model query is selected. The top window shows the various types of

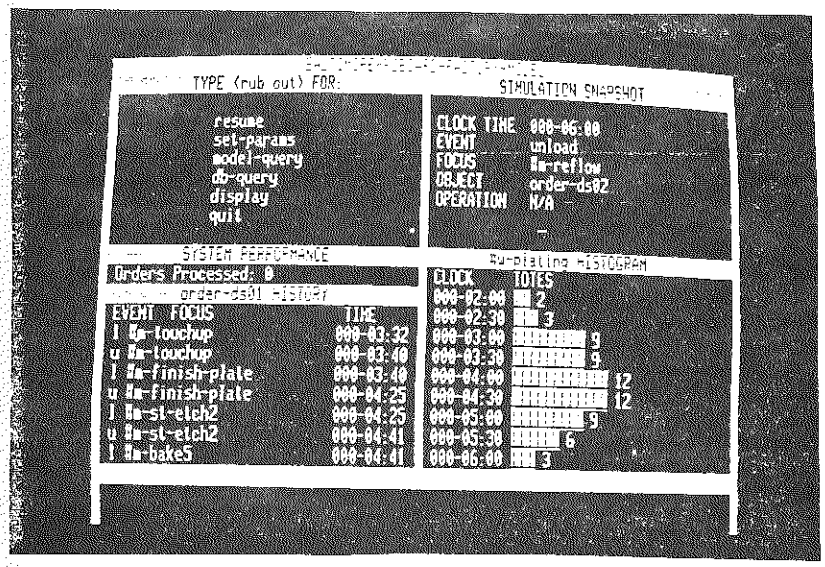

Figure 3-6: Simulation Snap-shot

queries that can be performed. The queries can be of the report type or of an individual type. The report type query displays statistics about a class of objects such as machines, where as individual type queries display information about an individual object. The bottom window displays a machine report consisting of busy time, percent utilization and number of orders processed. Figure 3-8 displays a query about an individual order. Individual queries can also refer to machines to find the order they are currently processing. Apart from the simulation process a separate graphics display process may also be concurrently executed to show the state of simulation by changing the color of the machines as they

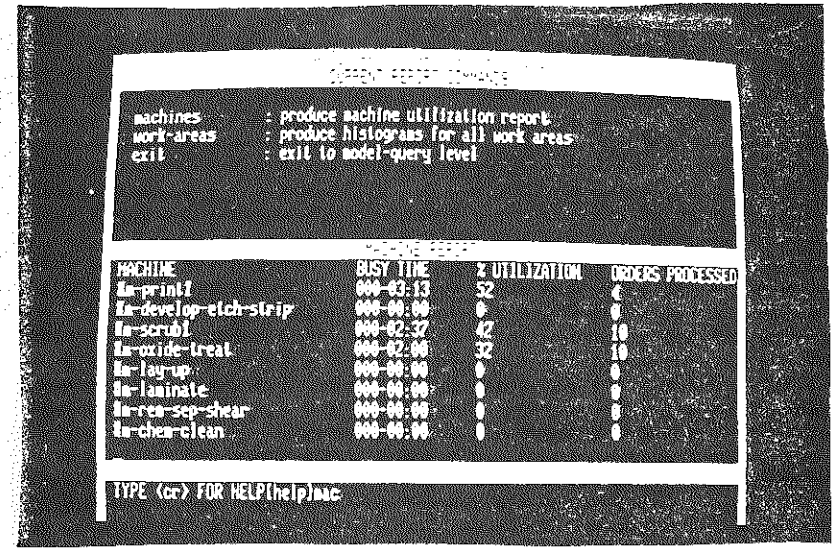

Figure 3-7: Model Query

change their state. It can also display the number of orders present in any work area. Figure 3-9 shows the layout of the entire factory. Figure 3-10 shows the close-up of one work area and the machines loacated in that area.

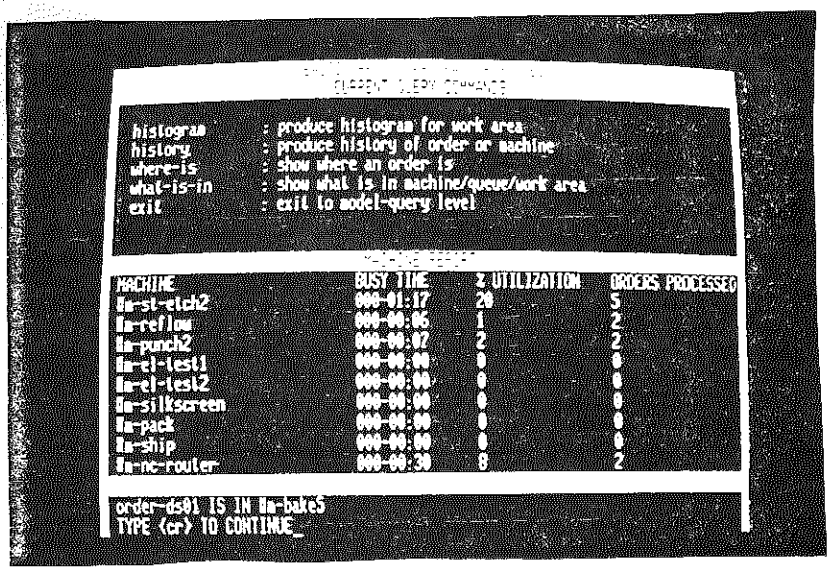

Figure 3-8: Display of An Order Query

4. Conclusion

In the KBS approach, we took the view that a simulation model need not be explicitly constructed, but rather be derived from a knowledge base. We also show that a suitable knowledge base can be constructed using the SRL knowledge representation facility. It also demonstrates that model acquisition can be accomplished by the instantiation of generic schemata found in a library for a specific domain. Events are represented as rules associated with schemata which

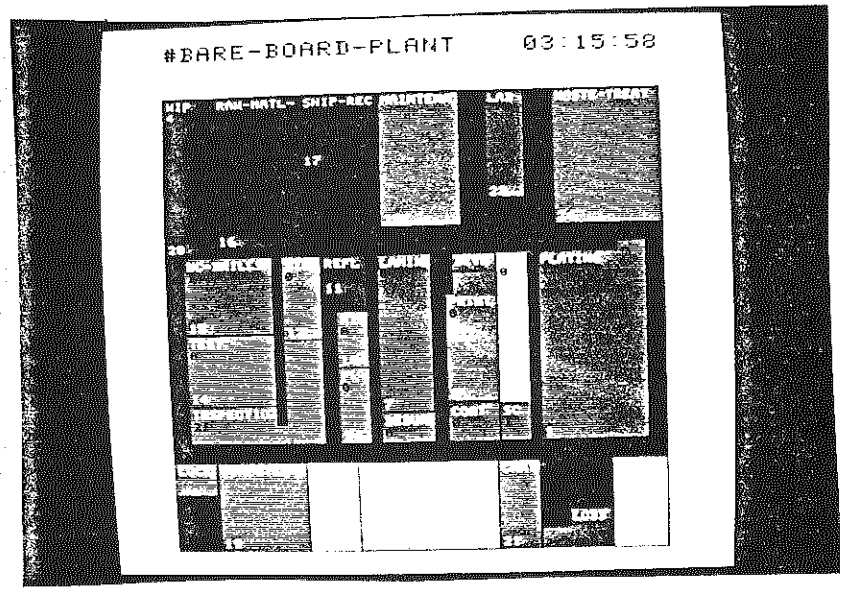

Figure 3-9: PCBOARD Factory Layout

provide a convenient method for specifying various actions in the model. KBS also provides a convenient method for selectively instrumenting the model for collection of performance statistics. Because of the schema representation of the model, model perusal, database query and integration with other modules such as a graphics display can easily be accomplished.

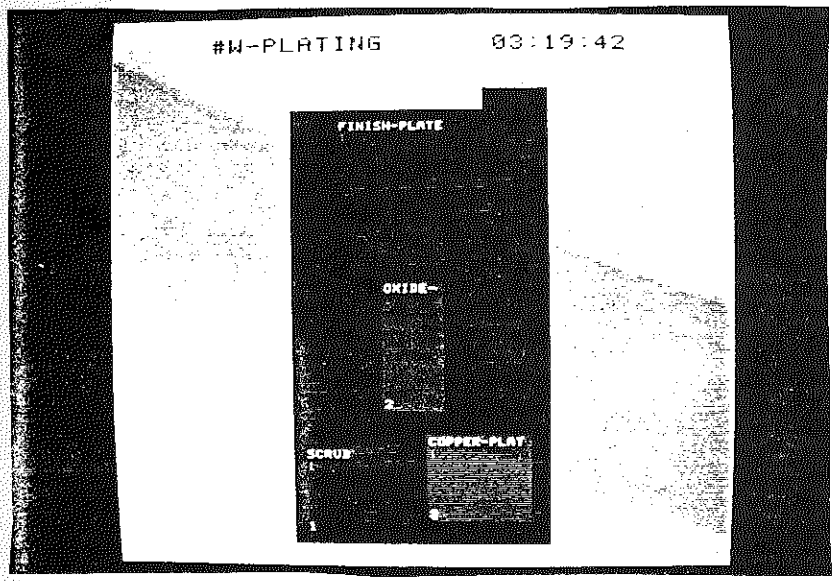

Figure 3-10: Layout of a single Work Area

[1]This research was supported in part by the Westinghouse Corporation, and by the CMU Robotics Institute.

[2]West Virginia University Morgantown, W.Va 26506