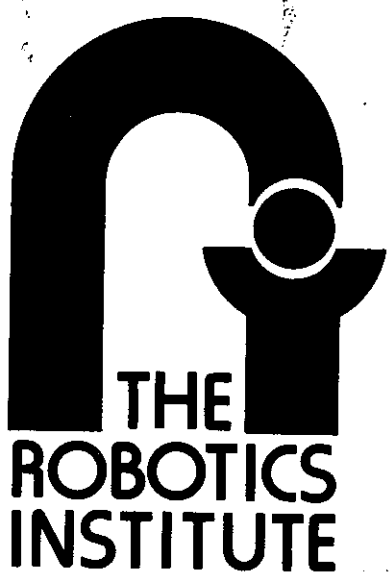
Carnegie-Mellon University

THE INTELLIGENT MANAGEMENT SYSTEM
AN OVERVIEW

Mark S. Fox

Intelligent Systems Laboratory
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213



CMU-RI-TR-81-4

The Intelligent Management System
An Overview

Mark S. Fox

Intelligent Systems Laboratory
The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

11 August 1981

Abstract This report describes the Intelligent Management System (IMS) project, which is part of the Factory of the Future project in the Robotics Institute of Carnegie-Mellon University. IMS is a long term project concerned with applying artificial intelligence techniques in aiding professionals and managers in their day to day tasks. This report discusses both the long term goals of IMS, and current research. It describes research in the modelling of organizations, job-shop scheduling, organization simulation, user interfaces, and system architecture. Examples of working systems are provided.

This research was supported by the Robotics Institute, Carnegie-Mellon University, and, in part, by the Westinghouse Corporation.

Table of Contents

List of Figures

1. Introduction

An important change is taking place in industry and industrial organizations. Products and services are increasing in complexity. The technology to produce them is becoming correspondingly more complex. Machine automation of physical processes progressively reduces the number of machine tenders and tedious low-skill jobs while simultaneously increasing the requisite number of high-skill, more challenging jobs. Highly skilled individuals cannot be interchanged as readily as the less skilled, leading to a more complex organizational structure with more demanding management tasks. Organizations, whether factories, ships, or hospitals, seem more difficult to manage. Managers typically respond to the situation by creating additional, specialist management jobs. Because of this new-found complexity, serious communication problems among the levels and subdivisions of management arise. Long range planning is often slighted because the complexity makes it difficult to pull all of the pieces together.

Classical research in the area of factory automation has been concerned more with production processes than with management. Yet, it has been observed that in many factories, white collar labor accounts for more than 50% of the cost of producing goods. In high volume, large batch-size production, such as lamp manufacturing, white collar labor costs account for much less than 50%. But in small batch size production, it can run as high as 75%. Contrary to popular belief, small batch-size production accounts for 50%-75% of the dollar value of goods produced in the United States. In metal-cutting, job-shop production environments, it has been found that only 20% of the time an order is in a factory, is it actually mounted on a machine. And during only 5%-10% of its time on the machine, are value-adding operations being performed. There are (at least) two approaches to dealing with this problem. The first is to discover new methods of producing products that do not suffer from these inefficiencies. The second approach is to increase the effectiveness of professional and managerial personnel. The project described herein is concerned with the latter. In the summer of 1980 we began the design and construction of what we call an *Intelligent Management System* (IMS). It is no longer the case that simple, single-function management systems, providing information access capabilities, are sufficient. Management systems must be more effective in the tasks they perform. They must integrate and communicate the knowledge and skill of the whole organization, making them available for management decisions. More importantly, they must aid profesionals and managers in carrying out tasks. Management systems must become more *intelligent*.

The goal of the IMS project is to aid supervisory personnel in their day to day decision making. It is concerned with both the type and level of functionality required by supervisory personnel, and with the cost of creating, maintaining, and adding new functionality. It is not sufficient to increase the effectiveness of one part of an organization, e.g., managerial, by increasing costs in another, namely programming and system support. Yet much of the software constructed today, while providing increased functionality, also requires increased programming support. Research and development must be concerned not only with functionality but with adaptability.

2. The Intelligent Management System

The goals of the Intelligent Managment System Project are to

1. Provide expert assistance in the accomplishment of professional and managerial tasks.

2. Integrate and coordinate the management of the organization.

To accomplish this the Intelligent Management System should:

- *Sense*: Automatically acquire state data. Sense the location of objects, state of machines and status of activities both on the plant floor and in supervisory departments.

- *Model*: Model the organization at many levels of abstraction. For example, machines, people, materials, orders, departments, need to be modelled in detail from both an attribute and a process view, including their interactions, authority, and communication.

- *Operate*: Provide expert assistance in the accomplishment of complex professional tasks within the organization.

- *Manage*:

  - Analyse and manipulate the model to schedule production and resource utilization, and answer short and long term state and planning questions. The system in this role is *passive* in that it responds to user initiated queries.

  - *Actively* monitor the organization and inform responsible personnel when important events occur. For example, when a machine break down occurs, not only is the foreman informed, but also maintenance, and the salesman who must inform the client that the order will be delayed.

- *Analyse-Optimize*: Analyse how the structure and the processing of the organization should be changed to further optimize some criteria such as cost, throughput, and quality.

Such a system, if it is to succeed in a business environment, must have the following characteristics:

Accessability: Interfaces to computer systems are usually idiosyncratic and difficult to learn and use. Also, systems that change require that their users be continually re-educated. Our goal for IMS is to enable all personnel to meaningfully communicate with it. The interface will gracefully interact with the user and provide guidance and help in deciding what the user needs.

Accountability: A major obstacle to computer acceptance is that users are unable to question how and why output was generated. Our goal is to construct an explanation system which will allow IMS to explain its actions at various levels of detail.

Adaptability: Currently, software systems are tightly coupled, requiring extensive re-programming whenever changes are required. This has resulted in the growth of the number of programmers needed to create and maintain computer systems. The goal of our research is to construct a theory of system design which will allow the users to modify the model, analysis, and processing functions without the aid of programmers. The end user introduces changes via dialogue.

Reliability: The system will not fail if one of its parts fails. No component is critical.

Reactability: The system, via its sensors and data monitoring will be able to detect changes in the organization. Detrimental changes are corrected if possible. Interested personnel are informed of the change, but receive only the information they require.

The construction of IMS requires the integration of a variety of technologies, many which require further research. While the total concept of IMS may take 10-15 years to reach fruition, many useful results will appear in the near future.

Research began in the spring of 1980. Three Westinghouse Corporation plants were visited[1]. Extensive analyses were carried out to determine problem areas and possible solutions. Three problems were chosen: job-shop scheduling, factory simulation, and process diagnosis. Systems were demonstrated for the first two applications in December of the same year. Research in all these areas and others, described later in the paper, continues.

In the following sections, what IMS looks like from a user's point of view is described. The remaining sections then describe the various areas of research underway in the project.

3. User's View

What the architecture of this system will look like, depends upon the type of organization. But there are many features that are organization invariant. Logically, the system architecture is distributed (figure 3-1). Most employee's will have a User Interface Process (UIP) that will act as an intelligent aide. A UIP is composed of a personal computer, graphics display, keyboard, microphone, and network interface (e.g., a SPICE machine (CMU-CSD, 1979)). The UIP will have either voice or typed natural language input. It will act as an aide in the sense that it will interpret and implement user requests and queries. All UIPs will be inter-connected via a communication network allowing them to cooperatively interact to solve problems and communicate information. The UIP will also carry out many of the employees well-structured tasks automatically. Each machine will have a Machine Interface Process (MIP) which monitors and controls it. It is also connected to the network, and can reply to queries and commands initiated by other MIPs or UIPs on the network. Lastly, there are Task Interface Processes (TIP). A TIP provides the focus for task management. It does most of the mundane task monitoring and control, freeing managers to do the more complex decision making tasks.

Examples of the types of interactions we would like to see are described in the following:

"Tell me when ..."

The marketing manager is under heavy pressure to get a rather large order out of the factory. He wants to be informed the minute it is shipped. He turns to his terminal and types the message: "Inform me when order X is shipped." His UIP translates the request into a rule "IF order X is Shipped

[1] A turbine component plant, a printed circuit board plant, and a flourescent bulb plant.

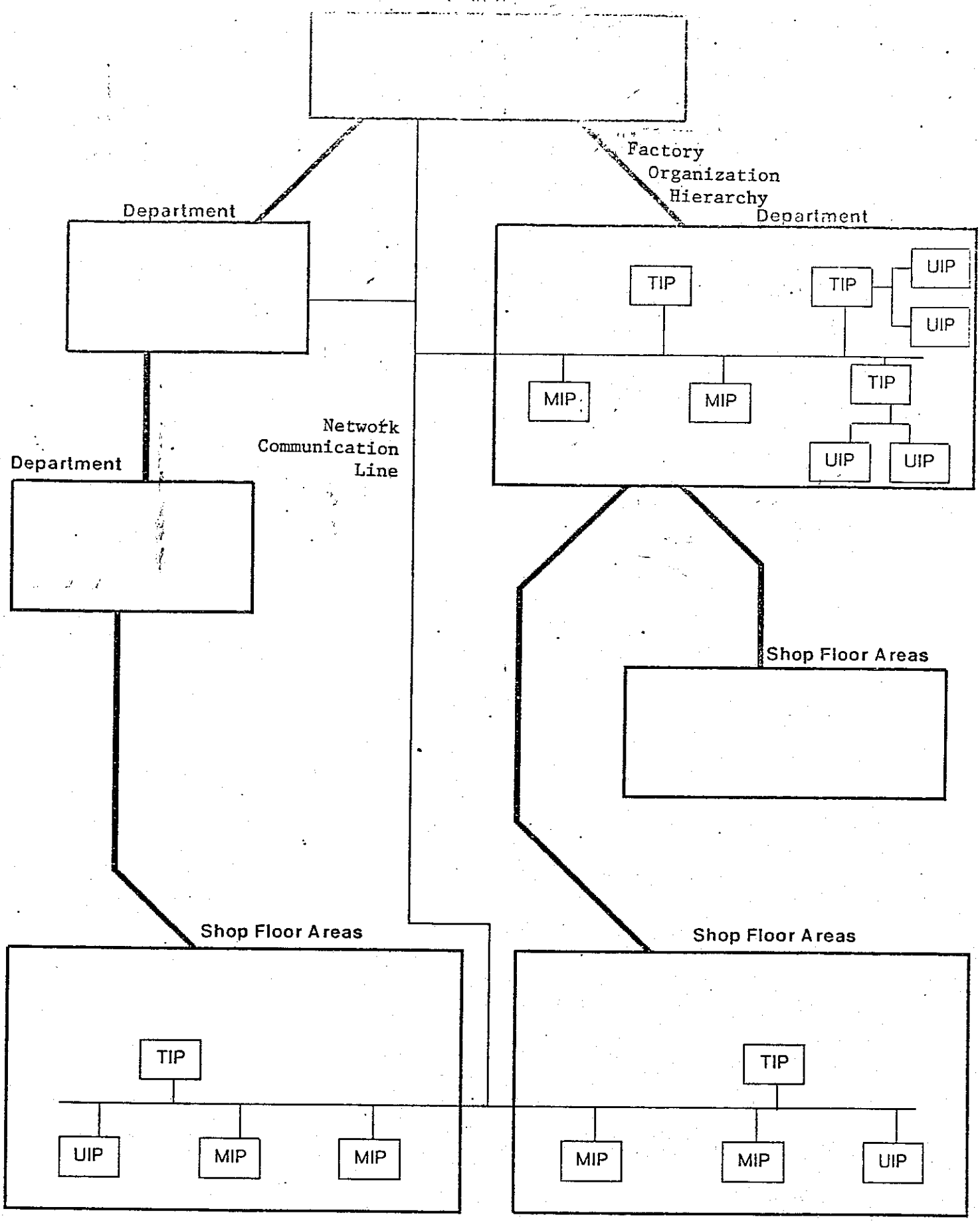


Figure 3-1: IMS System Architecture

THEN send message to manager Y", and sends it to the shipping TIP. The TIP monitors the system to determine when the rule condition occurs. When the order is shipped, the TIP interprets the rule, resulting in the shipping message being sent.

"You've got problems ..."

The milling machine breaks a tool while cutting a high priority order. The machine and order are damaged. The machine's sensors transmit the information to its MIP. The MIP analyses the problem, and shuts down the machine. It then informs the floor supervisor and the scheduler TIP of the breakdown; the scheduling TIP re-routes orders. A message is also sent to the maintenance TIP which allocates a maintanence person to fix the machine. Lastly, the MIP checks the importance of the order, and informs marketing and other personnel of the problem, if it affects their tasks.

"What if ..."

The manager in charge of production is considering the problem of a continually large back-log of orders. Should another machine be bought, or should the orders be subcontracted? He/she turns to his/her terminal and types in: "What are the effects on orders over the next six months if we buy another machine X?" The system then enters into a dialogue with the manager determining other information required to analyse the question. The UIP then scans the system wide functions that may help answer the problem. It finds a simulation module that can analyse structural changes in an organization. It gathers the initialization data and alters the factory model. It then runs the simulation, analyses the output and provides the manager with the answer, and further explanations.

4. User Requirements

In determining the functionality IMS is to exhibit, over 30 managers from three plants in the Westinghouse Corporation were asked to record the types of questions they frequently are faced with during the performance of their jobs. The following is a sample of their replies:

- What is the effect of changes in engineering specifications: designs, materials, process specifications, etc?
- What is the effect of defective work?
- What is the proper inventory level at various levels, i.e., raw, work in progress, finished parts, etc?
- What is the productivity level of employees in all categories and how does it compare to expected levels and trends?
- When should preventive maintenance be scheduled for a particular facility?
- What if the jig grinder goes down with maintenance problems?
- Based on downtime, cost of maintenance and cost of replacement parts, how do I determine when to buy a new piece of production equipment?

- What equipment should be added?

- Charlotte moves one (1) BB72 rotor two (2) months ahead of schedule and Lester moves two (2) BB22 rotors back in schedule.

    1. How are all promised dates for next six months affected?

    2. What manpower changes are needed by cost center to accomplish changes?

    3. What material delivery changes need to be made?

- I am a tool planner. How do I decide what tools are required to manufacture a new blade?

- How do I select the process to be used? What if the shop changes the plan and needs to perform a particular job on a different machine from the standard?

- Given orders and time frame information, it is difficult to plan whether to produce the product in-house or to subcontract it, where to acquire materials, or when to schedule production.

- Material X is not available, what schedule changes are required and develop new schedule?

- Internal Communication. Not enough information concerning the current state of production on the floor, orders in process, problems etc. is communicated quickly and succinctly to interested parties within the plant.

- Engineering Feedback: changes to products or tools by the engineering division are not adequately communicated and coordinated with changes taking place in the factory.

- Tooling Problem. The plant contains about thousands of tools scattered throughout Operator and machine time is wasted searching for tools necessary to carry out a task. Tools are "lost" by losing track of their location. Scheduling cannot be accurately done without knowledge of tool availability.

- Machine, materials, schedule, maintenance, personnel Status.

- If I have a database containing lamp prices, sales volume, ics (internal product cost) transportation cost, overhead allottment, .. for all lamp types. What will the profit be if I change transportaiton, or change the source of manufacturing.

- A priority of the entire division is to reduce the working capital. Amount of products in the channels, work-in-process, etc.. Want a system to speed the product through the distribution channels. High inventory, in transit, in process cost.

- Reduce the inventory levels by upgrading the forecast, reorder quantities, and lamps on demand.

- Analyse the effect of increasing throughput on profitability: include speed up of machines, increases shrinkage, wear and tear, greater inventory levels, etc.

- Analyse the effect on profitabiiilty of the increased sales of a certain product, including individual product margins, product promotions and scheduling.

- Keep records of: production, shrinkage, down time, absenteeism, safety purchases, perpetual inventory of materials and machine parts.

- Analyse down time by machine part number failure, shift, and by operator.

- Predict machine failures and prescribe preventative maintenance by sound viabration, machine timing, shrikiage trends, and end of normal part life.

- A program to determine correlations between gas, fill, temperature, stack up, and shrinkage.

The questions and problems listed above can be categorized into three broad classes:

1. Information access and communication.

2. Professional tasks.

3. System analysis.

Existing vendor systems can handle most of the information access problem, e.g., product tracking, reporting, billing, etc., by providing databases and appropriate access functions. But for many of the managerial level questions and problems, systems do not exist.

The following sections describe our first steps towards constructing an Intelligent Management System. Our approach has been to construct a solid system architecture and organization modelling base upon which to construct interesting functionality, whose choice has been motivated by the needs found in the plants visited. The problem of modelling organizations is discussed first because of the foundational role it plays in IMS. Next, sections on organization analysis and management are presented. They discuss the various analysis and management functions in IMS. How users interact with IMS is described in the section on User Interfaces. And finally the architecture of IMS is described in the section on integrating distributed systems.

5. Organization Modelling

5.1. Introduction

The purpose of organization modelling is to provide the information base upon which intelligent processes rely. What the contents of a model should be, and how it is represented is dependent upon the processes that use it. It is safe to say that an "Intelligent Management System" will require at least as much information as humans. The richness and variety of this information cannot be found in the databases of current management information systems. Nor is the form of the organization model related to current notions of database structures.

For example, a simulation system requires knowledge of existing processes including process

times, resource requirements, and its structural (routing) relation to other processes. It must also know when routings for products are static, or are determined by a decision process such as a scheduler. In the latter case, it must know when and where to integrate the scheduler into the simulation. If IMS is to generate the sequence of events to produce a new product, it must have knowledge of processes (e.g., machines) which includes the type of processing it can do, its operating constraints, the resources it consumes, and its operating tolerances. If data is to be changed in an interactive, possibly natural language mode, IMS must have knowledge of generic processes such as machines, tasks, and departments if it is to understand the interaction. It must also know what information is important and how it relates to other information in order to detect missing information and inconsistencies. Hence, the organizational model must be able to represent object and process descriptions (structural and behavioral), and functional, communication and authority interactions and dependencies. It must represent individual machines, tools, materials, and people, and also more abstract concepts such as departments, tasks, and goals.

Consider a process model of a florescent lamp production machine group. The purpose of a process model is to represent each physical process and the causal relations which link it with other processes. For example, the Lehr[2] process comprises of hundreds of subprocesses concerned with bulb grasping, positioning, heating, cooling, etc. Each is sequentially related with others in time. The performance of each is related to the performance of previous subprocesses. Each subprocess is described in terms of how it physically performs, its three dimensional movement, what and how it may grasp and heat an object, what object it expects, operating constraints, etc. The scope of the description is limited only by the uses to be made of it, and the information available.

One use of such a model is for machine diagnosis. Engineers spend a great deal of time watching a malfunctioning machine to determine what has gone wrong and what variables to alter to improve performance. One of the major stumbling blocks in the systematic analysis of such systems is the unavailability of process instrumentation for data collection. Yet the availability of the data solves only half of the problem. The other is the automatic analysis of data to find relations between system parameters and productivity. Statistical correlations are only a small part of the analysis. Statistics alone can only *suggest* relations. *Understanding* whether the relation is valid requires a thorough knowledge of the process itself and all its interactions. This cannot be performed without a sufficiently rich model of the process describing physical, functional, and time relations.

Another use of a model is to provide process cost analysis. Given a model, questions about the resource consumption and production of each process and subprocess can be answered. The individual process descriptions can be integrated across the group to provide summary cost information. But more importantly, because the model represents not only the process but the effects and relations among processes, questions related to process alterations can also be analysed. For example, what will the effect on cost be of changing a Lehr process to use microwave heating. And from a diagnosis point of view, we would want the system to determine whether the change would have any significant effect on shrinkage[3]. For example, the temperature of the glass may be lower

---

[2] Baking lacquer out of lamp phosphor.

[3] loss through defects.

after microwaving, hence the following process, end sealing, may not have glass at a high enough temperature.

In summary, the modelling system should provide:

1. A rich source of information.

2. A context within which all IMS modules can interact.

But if it is to be used in a changing environment and by non-programmers, it should also provide the following features:

Accessability: the model user should be able to easily peruse the model to determine its attributes and structure.

Extensibility: the model should be alterable by its users.

Consistency: when a model is created or extended, the modelling system should be able to determine when the model contains inconsistent information.

5.2. Basic Modelling System

The above discussion presents a case for a modelling system that supports a variety of functionality such as simulation, cost analyses, and process diagnosis, and a flexible user interface. Therein lies the question: does such a modelling system exist? Traditional, computer-based modelling systems fall into four categories:

1. Mathematical.

2. Discrete event, facility based.

3. Continuous.

4. Arbitrary program.

In looking at these modelling systems, we found:

- They are too problem specific, hence inflexible.
- The modelling techniques are difficult to learn by managers and engineers who are the prime users.
- Alteration may require substantial change to the model and related systems.
- Once constructed, the models are difficult to understand, peruse and verify.

Each use of the Intelligent Management System has at its core, a model of the organization. This model is shared by all subsystems to achieve their tasks. The modelling system provides the following features:

- The model is composed of declarative objects and relations which match the users conceptual model of the organization.

- The model incorporates a variety of representational techniques allowing a wide variety of organizations to be modelled (continuous and discrete). And it is extensible, allowing the incorportation of new modelling techniques.

- The user interactively defines, alters, and peruses the model.

- The model can be easily instrumented. For example, the model can be diagramatically displayed on a color graphics monitor at different levels of abstraction. The complete organization, or parts thereof, can be viewed with summaries (e.g., queue lengths, state).

- The modelling system is simple to learn to use because the modelling tools match the concepts people use to think about problems.

The modelling system is based on the knowledge representation system SRL (Fox, 1979a; 1981). SRL is a schema (frame-like) based representation language (Minsky, 1975; Bobrow & Winograd, 1977). One of its primary features is its allowance for the creation of user-defined (inheritance) relations. From a modelling point of view, this allows the model builder to define both objects and relations amongst objects that display inheritance and mapping properties that closely match the model building view of an organization. A schema describes the attributive, behavioral, and relational characteristics of a concept.

{{ **Machine**

CAPACITY:
QUEUE:
OPERATOR:
CONTENTS:
Restriction: (TYPE # is-a # product)
LOAD:
Restriction: (SET (TYPE # is-a # rule))
Default: # load-rule
UNLOAD:
Restriction: (SET (TYPE # is-a # rule))
Default: # unload-rule
}}

Figure 5-1: Machine Schema

In figure 5-1 the basic schema for a MACHINE is defined having many slots, e.g., CAPACITY, QUEUE. None of them are filled with values or rules, though restrictions and defaults for the values of some of the slots are specified. Figure 5-2 defines a CONTINUOUS-MACHINE which works much like a pizza

```
{{ Continuous-Machine
   { IS-A Machine
       USED-CAPACITY:
       LOAD: { # INSTANCE # rule
                 IF: USED-CAPACITY < CAPACITY
                 THEN: add 1 to USED-CAPACITY, add object to CONTENTS.
                 ELSE: add object to QUEUE }
       }
   }}
```

Figure 5-2: Continuous-Machine Schema

oven, it can be continuously filled up to capacity. A **Continuous-Machine** IS-A **Machine**. The IS-A relation between the two schemata allows **Continuous-Machine** to inherit attributes (slot names) and their values from the **Machine** schema. The LOAD slot defines the behavior of the machine when a load event ocurrs. The loading rule tests whether the machine has capacity, if so the object is placed in the machine, otherwise it is queued.

The **Machine** and **Continuous-Machine** schemata are generic schemata that form part of the basic modelling system. The system contains a variety of basic schemata which the model builder can use to model an individual organization. An organization model is constructed by instantiating the basic schemata with appropriate attribute values, e.g., capacity, and possibly, new behavioral rules. Schemata are structured (linked) through a user extendable set of relations: IS-A, INSTANCE, PART-OF, etc. For example, a circuit board baking oven (figure 5-3) can be instantiated as an INSTANCE of a **Continuous-Machine**. It has a CAPACITY of 10, and inherits its loading rule from **Continuous-Machine**. It is also part of a **Work-Area** in the plant called the **Baking-Shop**. The PART-OF relation allows the inheritance of locational information.

```
{{ PcOven

   { INSTANCE Continous-Machine
       CAPACITY: 10     }

   { PART-OF Baking-Shop  }

   }}
```

Figure 5-3: PcOven Schema

5.3. Model Accessability and Extensability

Current management information systems require the use of programmers when changes are made to the organization model. While such an approach is fine for domains where the model changes seldomly, the dynamics of a factory organization require continual updating of the model; both parametric and structural changes are constantly occuring. In lieu of employing a brigade of programmers to implement changes, the alternative is to construct a model acquisition system that allows the person initiating the change, to directly inform IMS of the modifications.

The IMS modelling system currently provides the following functionality:

- ***Accessability***: The user may interactively view each schema in the model using a variety of schema printing functions. Relational hierarchies can also be displayed. Color graphic display of the model is supported at various levels of abstraction.
- ***Extensibility***: An interactive schema editor allows the user to create and alter schemata in the model.
- ***Consistency***: A first version of a model consistency language has been developed. A consistency checker uses the consistency specifications to check a model and reports inconsistencies to the user.

Though the above mechanisms provide a good user interface to the modelling system, our ultimate goal is to allow managers to alter the model directly. By means of a natural language interface, the manager should be able to describe to IMS changes that have ocurred in his area of responsibility. To achieve this, the modelling system interface must:

1. Determine what part of the factory model is affected by the new information.
2. Check to see if the new information is consistent with what it already knows about the factory.
3. Determine the effect of the change on other parts of the model, and make the appropriate changes.
4. Query the manager when inconsistencies appear in the reconciliation of the new information with the existing model.

A natural language understanding and discourse modelling system to support model acquisition, factory layout, and job-shop scheduling is currently under development.

5.4. Flow-Shop Modelling

A complete model of a printed-wire-board (PWB) bareboard production plant[4] has been constructed to the machine level (August, 1980). Figure 5-4 provides a glimpse of part of the relational structure of the model without each schema's attributes. You will note that information on

[4] under design by the Westinghouse Corporation.

how to simulate and display the factory is embedded in the model. The model also includes schemata for operations, process sequences (lineups), products, personnel and others. The model has been used directly by our simulation system and to display the factory on a color graphics monitor. Figure 5-5 displays the complete layout of the plant. Each work-area is color coded according to type of processing. Under each name is the number of orders in process and queued for the work-area. At 1:15 there are 9 orders in the inspection area. The UIP allows the user to specify what part of the plant to display. Figure 5-6 shows a blow up of the inspection area. At 1:36 there are 6 orders in the inspect1 station and 8 orders in the touchup station.

Process flow is defined by a production (operation) lineup defined for each product type. When an order is unloaded from an object such as a machine, work-area, etc. the next operation is determined by information inherited by the object. For some objects access to a scheduling system is inherited via the PART-OF relation, for others the next operation is defined directly due to physical coupling of operations.

5.5. Job-Shop Modelling

A model of part of a turbine component production plant[5] has also been constructed to support simulation (section 6.2) and job-shop scheduling functions (section 7.2). The model contains information about machines, products, tools, work-centers, labor and cost data, and factory layout. Much of the schemata used to model the circuit-board plant were used in the turbine plant model. In some cases additional slots (attributes) were specified, e.g., cost data, and operation sequences were expanded to operation graphs to include alternate processsing routes in the plant. Job-shop scheduling was provided by inheriting a different scheduling system.

5.6. Machine Modelling

Our third modelling project, currently underway, is to construct a model of a flourescent lamp production line; including enough detail to support machine process diagnosis (section 7.4). This model focusses on the pre and post-conditions of individual machines in the production process, and on the causal relations that exist between and within the machines.

6. Organization Analysis

6.1. Introduction

The effectiveness of an organizational is determined by its ability to deal with environmental uncertainty and complexity. Environmental uncertainty and complexity coupled with organizational uncertainty and complexity results in sub-optimal, and even sub-satisficing organizational behavior. To produce requisite behavior, an organization must analyse its environment and adapt. But it is too often the case that management lacks the time or ability to carry out the analysis; hence the organization internalizes more rigid behavior. One of the most important but least understood aspects of an organization is its ability to adapt. Much of Organization Theory has been concerned

[5] Westinghouse Turbine Component Plant, Winston-Salem NC.

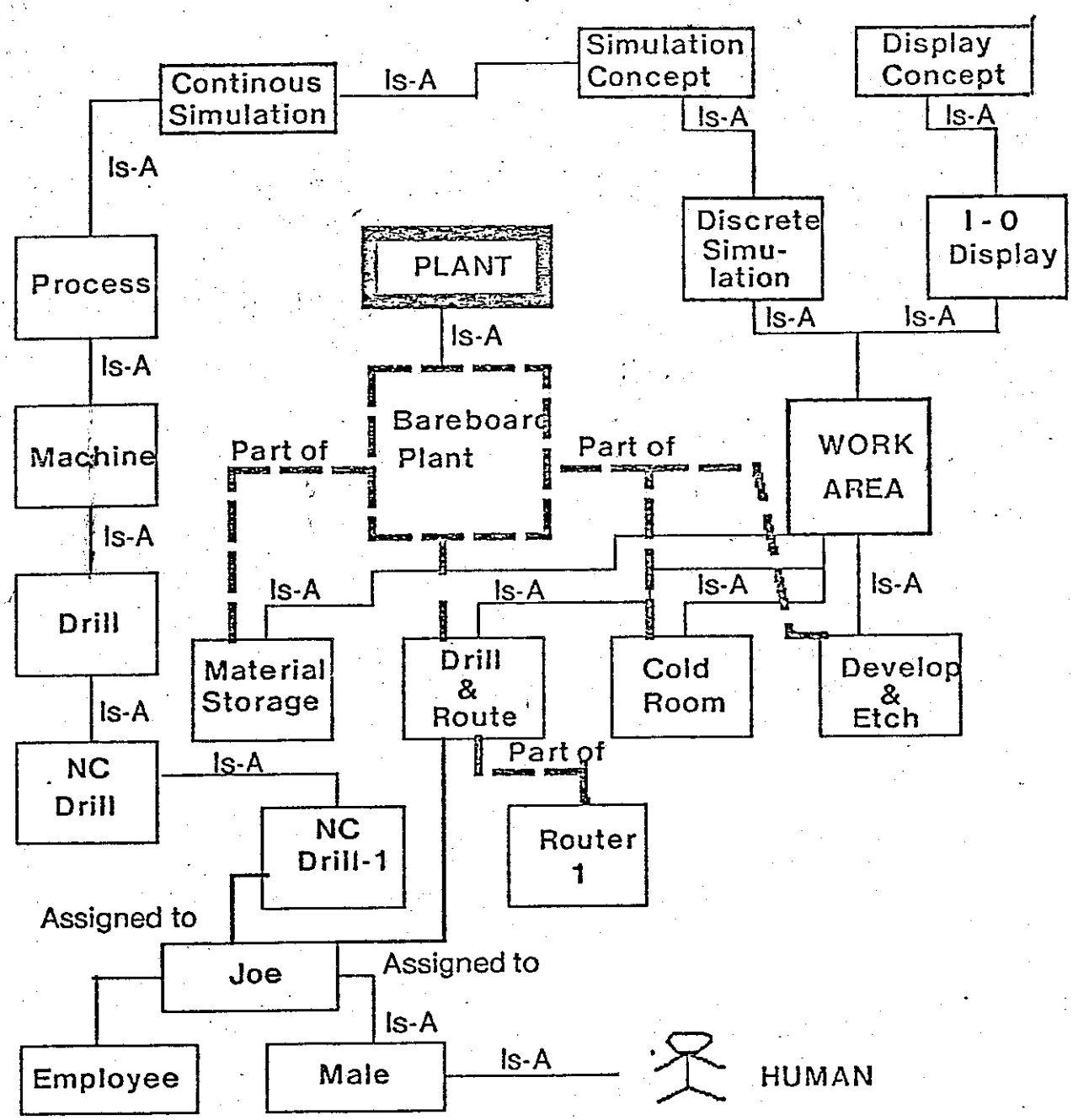


Figure 5-4: Flow-Shop Partial Model

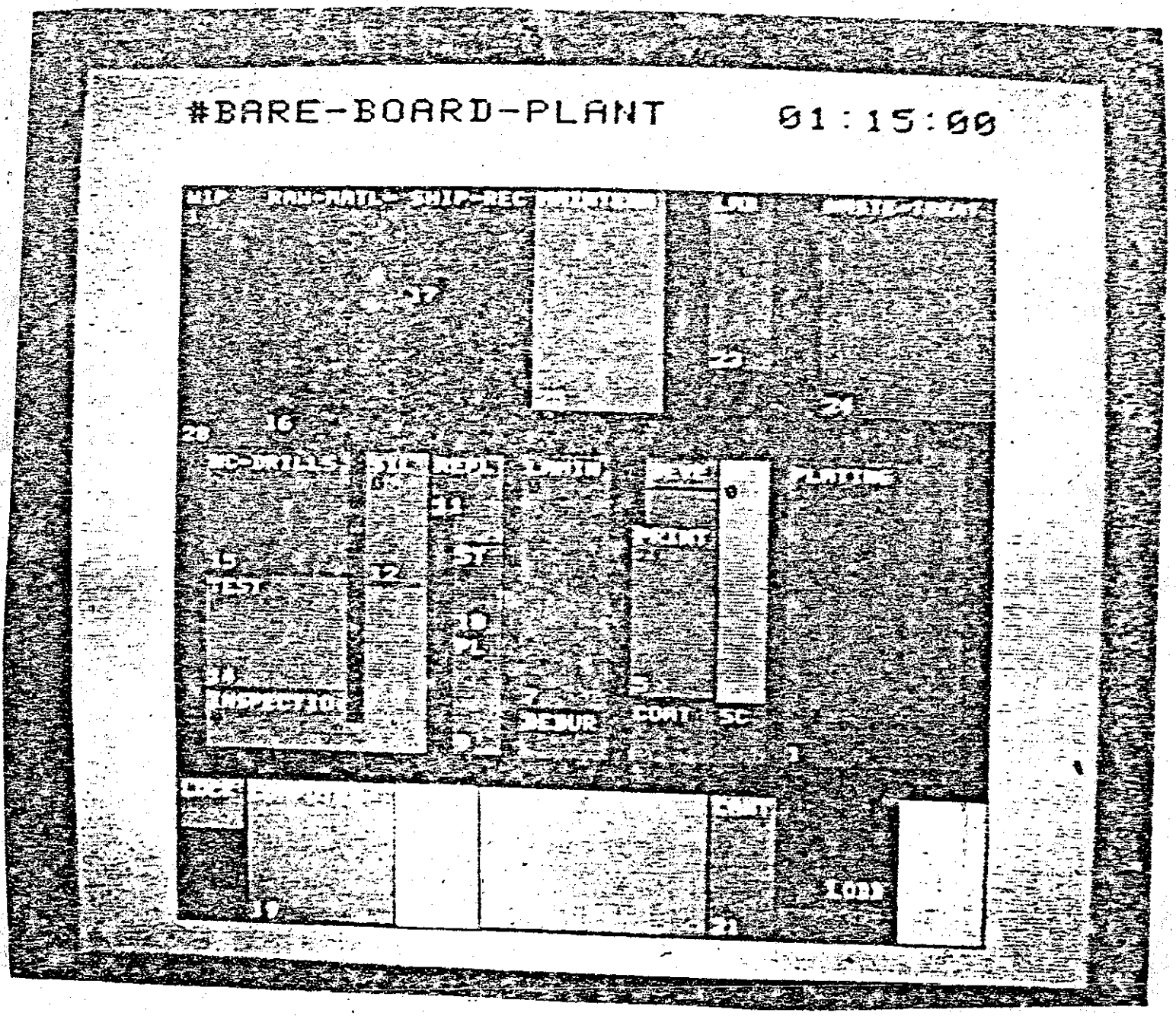


Figure 5-5: Monitoring the Factory

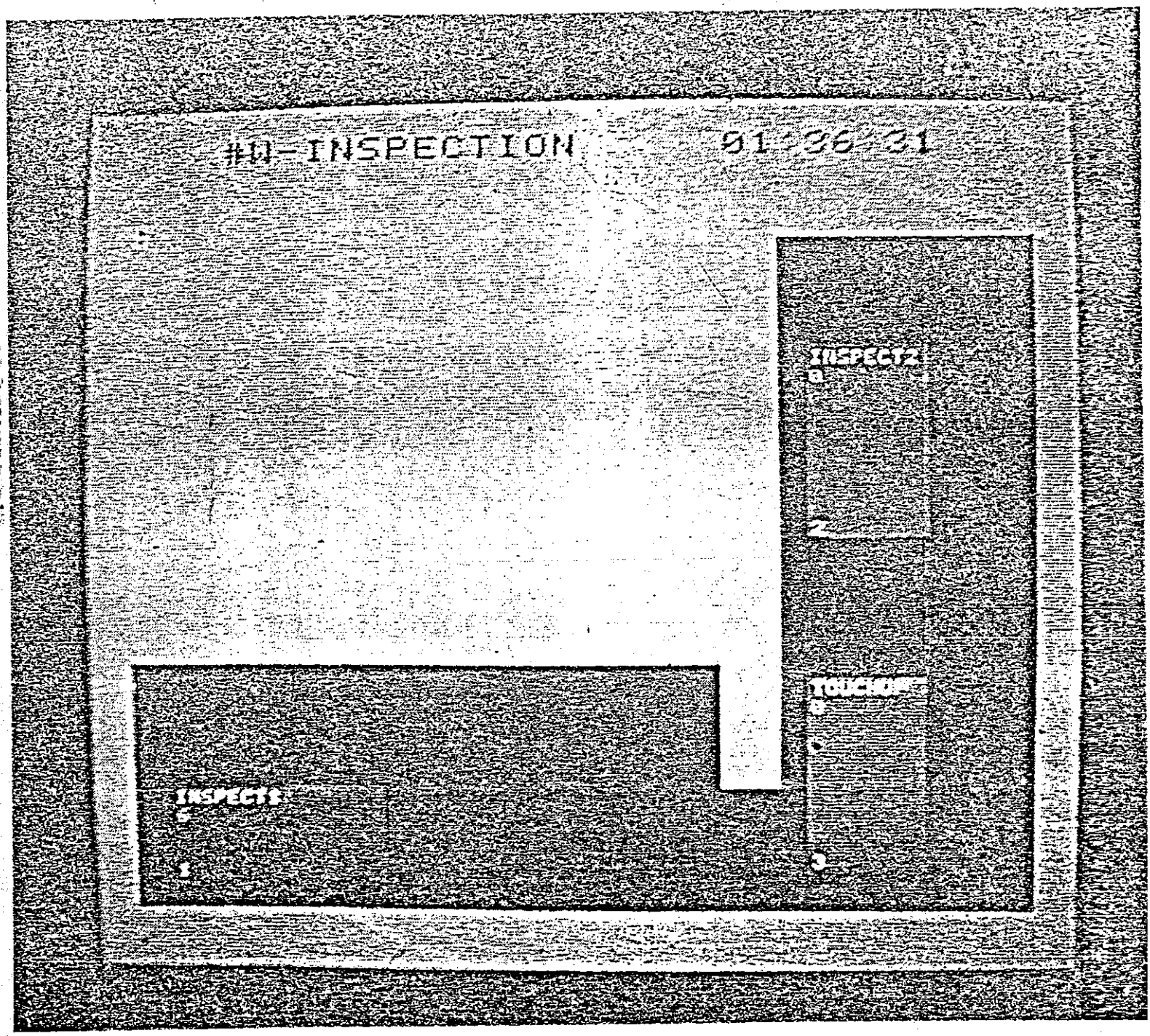


Figure 5-6: Monitoring the Inspection Area

with determining environmental and task characteristics that affect organization structure (March & Simon, 1957; Galbraith, 1973; Williamson, 1975). At best the results are descriptive (if not speculative).

Tools do exist that aid in the analysis of organizations. In particular, a simulation system allows one to test structure and processes. But the cost of constructing a simulation can be prohibitive, with the resulting system being of little use except for running the same or similar simulations again. Simple "what if" questions cannot be answered readily. A manager requires an intermediary, such as a system analyst, to answer the question. There is a definite need for more sophisticated tools for analysing organizations, and for providing usable tools directly to the managers and professionals. The following describes some of the research in IMS towards these goals.

6.2. Simulation

Many of the "what if" questions encountered in our analysis were concerned with the understanding of structural changes in the factory. For example, the decision to buy a more flexible but more expensive machine depends on its effect on the performance of the factory. In a job-shop, it is difficult to answer question analytically due to the complexity of the model. Simulations are used to predict the performance of complex systems. Hence they are a potentially useful tool to provide managers with. Why do research in this area when systems like GPSS, Simscript, etc. already exist? The reasons are many:

- Current systems are difficult to learn to use and require extensive computer programming skill.
- They suffer from the modelling problems described in the modelling section.
- Output is limited to the analyses provided by the system.
- Most systems are not interactive, and cannot be monitored easily while they run.

A discrete simulation system has been constructed that interprets the organization model directly (Reddy & Fox, 1981). The model was constructed to represent the complete organization, not just the information necessary to perform a simulation. The simulation uses as little or as much of the model as it requires. Hence the user only has to modify the general model without having to alter the simulation system. Some of the characteristics of the simulation system are:

- It is interactive, allowing the user to start, halt, and resume the simulation.
- The user can query the state of entities in the simulation as the simulation proceeds.
- The organization is displayed on a color graphics device with pertinent data (e.g., queue sizes) changing as the simulation proceeds. The user may view the organization at several levels of abstraction as the simulation runs.
- The user can interactively alter the model while the simulation is in progress and the simulation will continue with the alterations.

- The user can display a variety of analyses at the terminal.
- It accesses the *same* organization model that other functions use. Hence, only one model is maintained in IMS.

The simulation system proper is small. Its sole purpose is to manage the event queue. How events are executed is defined in the model. For example, a load event of the **Pcoven** (figure 5-3) would be accomplished by evaluating the contents of the **Pcoven**'s LOAD slot, which is inherited via the IS-A relation from the **Continuous-Machine** schema (figure 5-2). The contents of an event slot is a set of rules defining the event's behavior.[6]

Data gathering for analysis is accomplished by associating data gathering rules with the appropriate slots in the model. In particular, rules can be specified to be evaluated anytime the contents of a slot are changed.

Simulation monitoring is handled in the same manner as data gathering. Display rules are associated with slots in the model. When a slot value is changed, the associated rules are evaluated, resulting in display changes.

Figure 6-1 shows one of the interfaces. The upper right window displays each event as it ocurrs. It is maintained by display rules in the event scheduler. The lower right window monitors the queue of a user specified facility. The lower left window follows an order through the factory, printing each event as it ocurrs. The upper left window displays the available commands. Additionally, the user can monitor the factory via a graphics display (figures 5-5 and 5-6). The graphics display gives factory wide or close-up information, e.g., queue lengths, machine status, etc.

The simulation system is part of a general analysis system to be used by managers directly. Research is continuing to extend its capabilities (e.g., multi-level simulation, continuous system simulation, etc.) and making the interface simpler to use by managers.

6.3. Technology Choice Assessment

The last decade has seen an increase in the construction of flexible tools and machinery. Their flexiblity allows them to perform operations previously done by more specialized machines, at a lower or higher cost. While on the surface, the more flexible machine seems desirable, the issue is more complex. Introducing new technology may raise costs, further complicate scheduling and planning, require high operator skill and more education, and so on. The issue of technology choice is under investigation jointly with the Engineering and Public Policy Department of Carnegie-Mellon University (Ayers, 1980; Miller, 1981).

---

[6] Klahr & ?? (1980) use a rule-based approach to simulation.

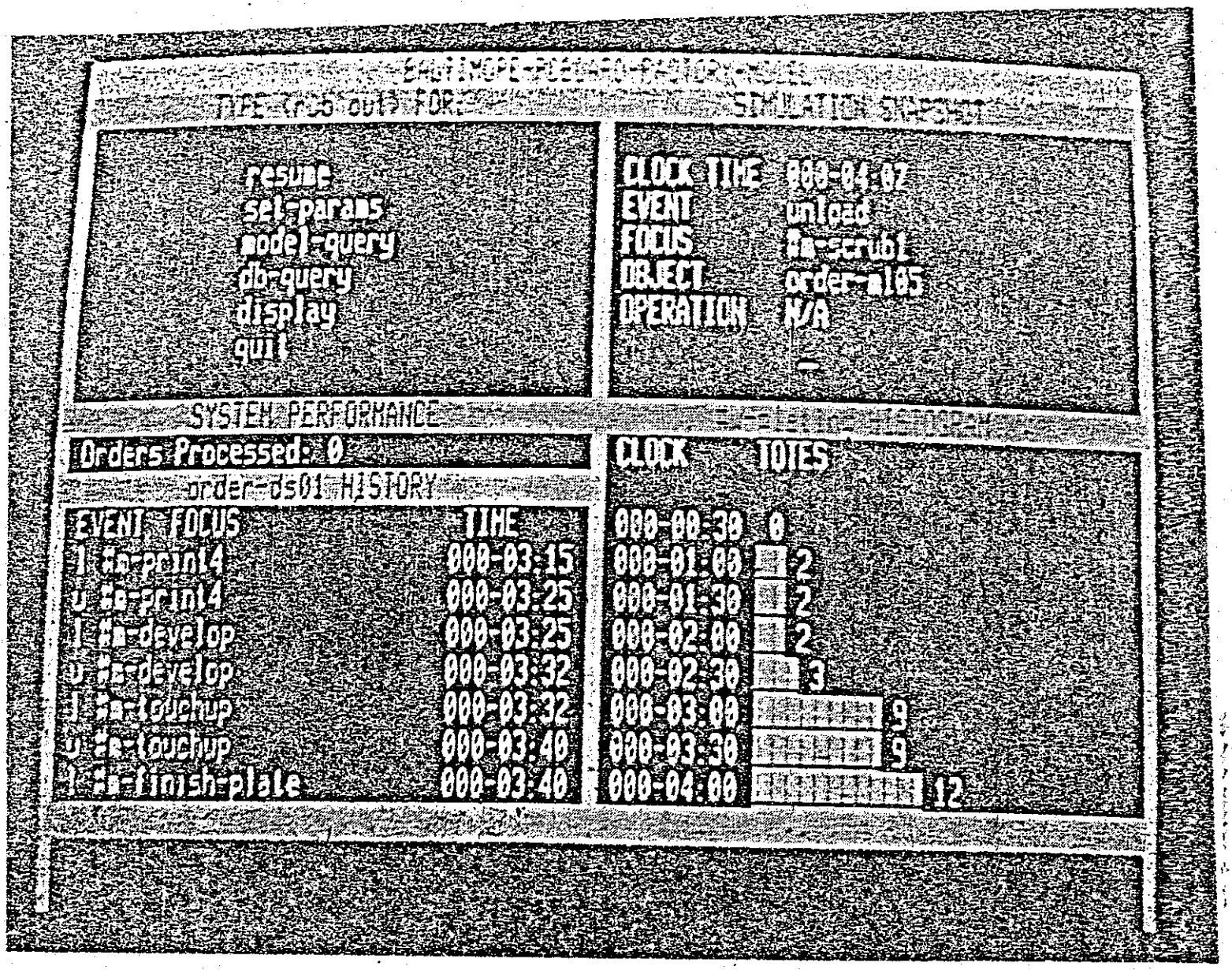


Figure 6-1: User Interface Display for Simulation

6.4. Structure Analysis

Whether generating a more optimal process, or diagnosing an existing one, the organization must be analysed. Analysis methodologies fall into two categories: analytical and experimental. The latter approach includes simulation, and is used when analytical techniques do not exist, are intractable, or the required information is missing. But analytical approaches are still very useful, especially in a local setting. Many organizational changes can be measured by analysing the causal relations that exist "around" the proposed change. By following causal links and measuring their change effects, most generated changes can be adequately tested. The generation of changes is dependent upon the existence of causal or dependeficy information that can denote high expense areas measured by some criteria.

In the laboratory, we are extending the organization modelling research to include organization structure analysis. Our goal is to construct a system that can analyse an organization model and suggest structural changes in order to improve performance as measured against some well defined criteria such as profit, reaction time, and quality of output.

7. Organization Operation and Management

7.1. Introduction

The purpose of our research in organization operation and management is to provide intelligent aids to managerial, professional and salaried staff. While the primary goal is to provide expert level functionality, other factors must be considered. First, the aid must be interactive to allow access in a timely fashion. It must be integrated with the rest of IMS so that it can access available information and functions, communicate with other users, their UIPS, TIPs, and MIPs. Just as important, the aid must be constructed in a manner that its declarative and procedural knowledge can be easily altered and expanded. In order to achieve these goals research from artificial intelligence, computer science, and management science have been used in the construction of these aids.

7.2. Job-Shop Scheduling

One of the most important functions in a factory is scheduling. Simply, scheduling is a resource assignment problem which is constrained by resource availability of one form or another and organizational goals in a job-shop. Scheduling in a job-shop is a problem because there is more than one way of producing an item, and that all orders are competing for the same resources (machines, personnel, materials, tools, etc.). In the factories we are working with, there are several thousand *distinct* product models. At any given time, there could be between one and several hundred duplicate units of a particular product model on the floor. There are several alternative routes that each product model could travel through the factory, depending on the availability of physical production resources, other active orders on the floor, and on the relative importance of time, cost and other cost constraints.

Although the physical number of machines in the plant, or the number of operators employed does not change too often, the set of machines, tools, and workers that are *actually* available at a given moment is constantly changing. Job-shop flexibilty is only part of the problem. Dynamically changing

states is another. Machines break down often, at unpredictable times. A worker may get hurt and have to leave the shop floor, leaving the machine unattended. Small tools may disappear because the operator forgot to return them to the tool crib, or lent them to someone else without recording it.

Existing scheduling systems, for the most part, are inflexible. They are based on a "static" model of critical production resources, based on the number of machines, tools and workers which are available under "ideal" conditions, and not on the set of production resources which are *actually* available at a given moment. These "static" models can not be modified at the rate at which changes occur. As a result, these systems are limited in their usefulness, since there are few times when the "static" model inside the computer matches actual operating conditions. Many factories which currently have these types of systems *still* have to schedule manually because of the frequent changes in operating conditions.

An interactive, real-time, job-shop scheduling system has been constructed which interprets the factory model directly[7]. It schedules orders on machines for complex job-shop environments where orders require a large number of operations, and there are many different ways of producing an order on machines under high contention. It allows the user to enter orders and specify scheduling constraints. The specifiable constraints are

- start and due dates
- operations costs
- operation alternatives
- machine alternatives
- machine down-time and maintenance
- order priorities
- order lottings
- lot priorities

The system will generate a schedule that attempts to satisfy the constraints. The system uses a knowledge-based heuristic search to construct forward (from start date) or backward (from due date) planned schedules. The scheduler is a factory floor level system which updates its factory model (e.g., machine, order, personnel, resource status) either by user input or through computer messages. And the system does a minimal rescheduling in reaction to any state change in the factory. Figure 7-1 depicts the scheduling system after it has constructed a schedule. The right window is the operation sequence chosen, the left is the machine reservations for the lot.

While the scheduling system takes into account more information than current scheduling systems, experience with the first version has shown that an interactive job-shop scheduling system must account for *all* the constraints that human schedulers use in manually constructing schedules. Our analysis of our interviews with schedulers and other factory personnel has shown that they receive information from more than 20 sources in the plant, such as forging, tooling, NC programming, and materials, marketing, and forecasting. Much of the scheduler's time is spent gathering these

[7] The system was demonstrated in december 1980 at CMU, a second version is under development and will be demonstrated in the fall of 1981.

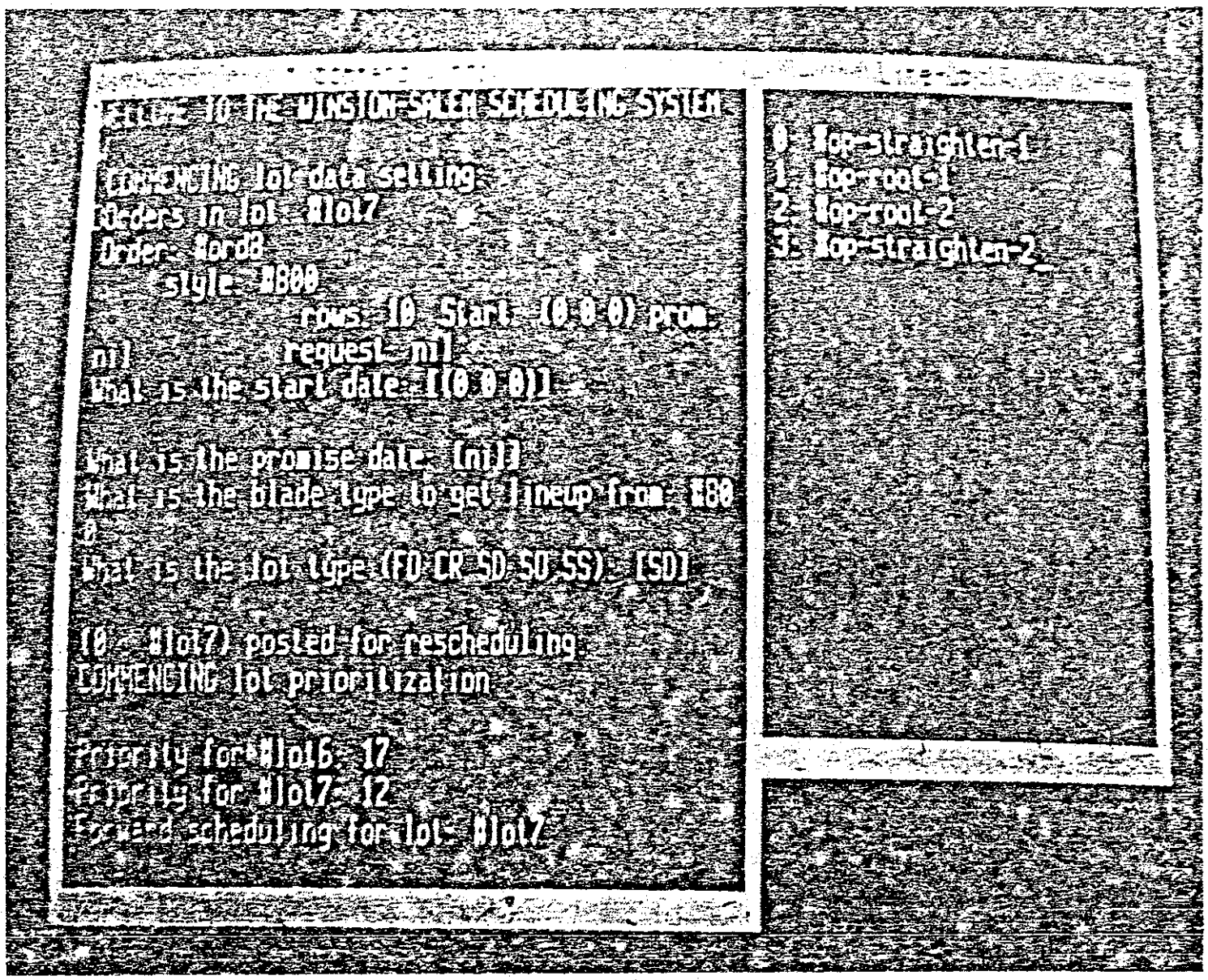


Figure 7-1: Scheduling System

constraints, and then constructing a schedule that satisifies as many of the constraints as possible. Hence, the problem of scheduling can be viewed as the problem of constraint satisfaction. But it is often the case that many of the constraints cannot be met. Under these conditions the human scheduler bargains with the constraint sources to have them altered. It follows that a scheduling system should not only take into account the large number of constraints, but should consider the conditions under which the constraints can and should be *relaxed* and/or *strengthened*. For example, if there are normally five workers available to perform a task on a given shift, and there is a rush order to get out, then the system should recognize the workers available constraint can be relaxed so that more workers are put on the the job. From a problem solving point of view, in addition to knowing what operators can achieve a goal (e.g., machinists), you must know what and how operators can be added, removed or modified. Hence an intelligent scheduling or constraint analysis system must:

- Consider all constraints that may impact a scheduling system.
- Attempt to satisfy them according to importance.
- Decide *when* constraints should be relaxed.
- Choose *what* constraints should be relaxed.
- Decide *how* to relax the constraint.

Accordingly, a second scheduling system is under development and is to be demonstrated in the fall of 1981. The research emphasis of this system is on the general representation and utlization of constraints, and on how these constraints can be dynamically relaxed. Figure 7-2 is an example of a general range constraint schema. It depicts the constraint on the value of a variable and the utility of changing its value. An order due-date can be viewed as a **#continuous-value-constraint** (figure 7-3) The utility of shipping early or late is depicted by the min and max val utility slots. While constructing a schedule, the system uses the constraint specification in proposing and evaluating schedules with alternative ship dates.

7.3. Factory Monitoring

As discussed in the section on simulation, the display facilities used to monitor the simulation are attached to the model. Hence, if the model is updated from real-time messages from the factory floor, as opposed to the simulation, figures 5-5, 5-6, and 6-1 can be used to monitor the real-time operation of an organization.

7.4. Process Diagnosis

Complex production environments may introduce a variety of defects into a product. The defect may not appear until later in the production process, where cumulative processing compounds the earlier induced defect to a point where signs of its manifestation appear. The purpose of process diagnosis is to understand each step of the production process to determine what type of problems can occur and the causal (cumulative) relations that can exist amongst sequential steps in the production process. The goal is to construct a system that monitors the production process, and

```
{{ #continuous-value-constraint
   { IS-A #constraint
       PRE-CONDITION:
           Comment: the conditions underwhich this value constraint
                    can be used. There may be many value constraints specified for a slot,
                    only one of which may apply.
       VALUE:
           Comment: desired value.
   MIN-VALUE:
           Comment: smallest value
       MIN-UTILITY-FUNCTION:
           Comment: takes the value as a parameter and returns the cost of approaching minvalue.
       MAX-VALUE:
           Comment: largest value.
       MAX-UTILITY-FUNCTION:
           Comment: takes the value as a parameter and returns the cost of approaching max-valu
   }}
```

Figure 7-2: Continuous Value Constraint Schema

```
{{ #due-date
   { IS-A #continuous-value-constraint
       (map VALUE --> DUE-DATE) }
   }}
```

Figure 7-3: Date Constraint Schema

when defects occur, analyses the problem to determine the possible points in the production process which could have caused it. A machine group[8]. monitoring, signalling, and control system has been designed, and the prototype simulation has been demonstrated. Our next step is to construct a training simulator for new operators, before completing the diagnosis system. The rational for this multi-step approach is both pragmatic and theorertical. The group does not contain adequate sensoring to support diagnosis at present, but new senors will be added over the life of the project. The simulator project, when completed, will provide the group model for the diagnosis project.

[8] A machine group is a set of machine integrated into a automated flow shop. The machine group produces flourescent bulbs.

8. User Interfaces

8.1. Introduction

An obstacle preventing the introduction of intelligent systems into organizations is the lack of reasonable interfaces. The keyword here is reasonable. The reasonableness of an interface is not absolute. It depends upon the person accessing the system: knowledge of the system, expectations, time constraints, etc.; the style of the interface: menu, touch screen, voice, typing, interactive, batch, and the functionality of the system. Successful systems tend to constrain the expectations of the user, and then optimize the interface for the constrained task (e.g., ZOG (Robertson et al., 1979), DAISY (Buneman et al., 1977)). This is sufficient when the task itself is naturally constrained, e.g., sales order entry, inventory control. But many of the ill-structured tasks that managers do, e.g., planning, forecasting, are not well enough understood, let alone highly constrained. If the goal is to introduce intelligent aids at the professional and managerial level, then the interface's acceptability will increase as the interface closely approximates typical human interactions.

The purpose of our user interface research is to explore the problem of interfacing people with machines by incorporating dialogue and problem-solving capabilities in the UIP. During the first year of the project, research focussed on modelling, managing and analysis functionality. Little was done in the area of user interfaces. Since then our emphasis has shifted to include interfaces. The following describes our accomplishments and our goals.

8.2. Information Display

In a multiple process environment, monitoring more than one process can be difficult when using only a single display. Previous solutions to the problem entailed programming each process to cooperate in the use of the display, essentially hard-wiring their interaction. Changes in processes, information to be displayed, etc. required each process's display code to be altered. We have solved the problem by providing a communication and display system that allows processes to communicate information for display, independent of who is to receive it and the device upon which it is to be displayed. This allows a process to dynamically communicate with any other process and have the communication appear on a screen in conjunction with information from other processes, without interference or recoding (see figure 6-1).

This is accomplished by defining each primitive type of information: text, histogram, bitmap, etc. as a message primitive (figure 8-1). A message primitive has attributes such as source and destination process so that it can be communicated between processes. Once a message is received at a process, a display-handler associates it with a logical window (figure 8-2) in a logical display. The window consumes the message causing the physical display to be altered. Each window of the display may be displaying messages from a different source (process). Each process may have many logical displays (figure 8-3) of which only one may appear on a physical device, i.e., the display is active. If a message is assigned to a window which is part of an inactive display, the message may be queued, to be consumed when the logical-display it is part of is activated. Messages, windows, and displays have priorities, which when combined, may cause the currently active display to be replaced by a higher priority display.

{{ **Message**

SOURCE-PROCESS:
Restriction: (TYPE is-a Process)
SOURCE-PORT:
Restriction: (TYPE is-a Port)
DESTINATION-PROCESS:
Restriction: (TYPE is-a Process)
DESTINATION-PORT:
Restriction: (TYPE is-a Port)
PRIORITY:
VALUE:
}}

Figure 8-1: Message Schema

8.3. Natural Language Interface

The goal of *accessability* can only be achieved if users are able to communicate with computers in a language that is natural to use. An obvious candidate is English. Over the last 20 years, researchers in artificial intelligence have constructed natural language understanding (NLU) systems that allow users to type English statements and questions into the computer. The state of natural language understanding research does not allow the computer to understand arbitrary sentences; they must be restricted to a particular task or domain. But, research continues to expand the capabilities of such systems.

In IMS, simple, constrained, English input is being experimented with. Two tasks have been chosen, model building and scheduling, to explore the use of natural language interfaces. Using the parsing system of Carbonell & Hayes (1981), a grammar for scheduling has been constructed and is being tested.

8.4. Explanation

Much of the failure of computer systems can be ascribed to their inability to *account* for their results. Once the output is produced, the user cannot delve into how results were produced without reading the computer program. Since many end users are not programmers, this is intolerable.

An explanation facility provides the means by which a user can investigate how results were derived, why a particular action was executed, or what the current state of the system is. The explanation facility is a natural language generating system which is able to maintain a discourse centered around analysing a particular action or state. The purpose of NLU is to translate English sentences into a representation that the machine understands. The explanation system describes the machine's reasoning processes upon request.

{{ #window

ORIGIN:
LENGTH:
WIDTH:
BORDER:
COLOR:
Comment: color of window background
FONT:
Comment: font for message displayed (may not always be applicable).
HEADER:
Restriction: (TYPE is-a header)
Comment: description of header for the window
#DISPLAY-ITEM:
Restriction: (TYPE is-a # message)
Comment: currently displayed message
TRANSFORM:
ITEM-QUEUE:
Restriction: (LIST (REPEAT (TYPE is-a # display-item)))
Comment: list of messages waiting to be displayed.
PRIORITY:
XMODE:
Restriction: (SET (OR scroll clip wrap))
YMODE:
Restriction: (SET (OR scroll clip wrap))
CURSOR-MODE:
Restriction: (SET (OR echo reset))
Comment: if echo, then typed input is echoed.
if reset, then when new item is displayed
then cursor is reset to position before
the item was dispayed.
}}

Figure 8-2: Window Schema

8.5. Discourse Modelling

The act of describing an organization can be quite complex from a natural language understanding point of view. Not only must each sentence be understood, but ideas communicated earlier are continually refered to directly and indirectly. Consider the following description:

There is an inspection area.
It seats 12 inspectors.
They have a desk, magnifying lamp and a terminal.
The room is next to the cold room

```
{{ #display

    DEVICE:
        Restriction: (OR grinnel concept100)
    #WINDOW:
        Restriction: (SET (TYPE is-a #window))
        Cardinality: (1 INF)
    STATE:
        Restriction: (OR Active Suspended Unused)
        Default: Unused
    PRIORITY:
}}
```

Figure 8-3: Display Schema

In order for the UIP to carry out a conversation, it must track the flow of ideas, bind referents, watch for and signal inconsistencies, etc. This ability is referred to as discourse modelling.

8.6. Planning

The true mark of intelligence in management systems is the ability to use the best information and apply the best methods in answering a user request. Consider the task of determining the effect of adding or changing a machine in a job shop. There may be many ways of doing this such as statistical analysis, simulation, and looking at similar situations. Deciding exactly how to measure the effect is a planning process, where a sequence of tests and actions must be constructed to arrive at an answer. This sequence is chosen from a set of competing methods. Planning requires knowledge of what tools are available, how effective they are, and the conditions under which they are to be used. Using a tool may require planning the use of other tools to provide the required conditions. For example, before doing a simulation, a model must be constructed, and initialization data be gathered.

If the user interface is to act as an intelligent aid, it must be able to extend the user's capabilities by determining the most appropriate means to implement the user's request or command. The planning mechanism examines the data and functions known to it and constructs a plan whose execution accomplishes the task. The planner is to be constructed as a production (rule-based) system in OPS5 (Forgy, 1981).

8.7. Personalization

The user interface process can be viewed as an extension of the user. But users differ in a variety of ways: task prescription and language of communication for example. The UIP must *know* the user in order to amplify or restrict the user's capabilities according to their position in the organization. Consider the machine operator. He should be able to communicate with his UIP using the vocabulary of his task, he should be able to provide status information, and request scheduling information, but

he should be restricted from finding out the plant manager's salary.

9. Integrating Distributed Systems

9.1. Introduction

Organizations are distributed. So will be the systems that manage them. Reliability and processing power needs require it. One can see this in the market place. The trend in vendor systems is towards distributed systems. In fact, one cannot read a trade magazine without stumbling across the word a dozen of times. Another term which has gained notoriety of late is *integration*. Much of the software produced to date consists of stand-alone programs that cannot interact with other programs. The goal of integration is to allow individual programs to share information and cooperate in their processing. Still one attribute eludes these systems, that is the *flexibility* with which these systems can be distributed and integrated. If new types of information is being added and/or removed, and the system architecture altered, these distributed integrated systems require extensive re-programming to *adapt* to the changing environment. In fact most of the cost in software production is in the area of maintenance, which is the guise, in many cases, for adapting the software to environmental changes.

Research is proceeding in the *flexible integration* of the multiple functions of the Intelligent Management System. As described in section 2, IMS is comprised of processes of type UIP, MIP and TIP. These processes run on processors spread throughout the plant and are connected by a communication network (figure 3-1). Any process may dynamically spawn other processes according to its problem-solving needs. For example, a scheduler's UIP may have to analyse the effect of alternative priority ratings on orders over the next year. The UIP first determines how the question can be answered. It communicates with a module librarian process (TIP) to see if there are any modules in IMS that may help in answering the question. There is a module called "simulation" that can simulate a discrete-event model over time. It also finds that there are modules that can instrument a model to gather data over time. And a module that can analyse time-series data. The UIP acquires the module definitions from the librarian and spawns as many processes as needed to execute them. Knowledge of what modules are available to solve a problem or answer a question is not programmed into a module, but is part of a module's goal description which is accessable by other modules in the system. This is in the tradition of stimulus-response frames of Hearsay-II (Hayes-Roth & Lesser, 1977).

The key points in this organization of processes are:

1. Modules have been created that provide *generic* functionality.

2. Modules are described in a machine interpretable manner.

3. Processes have problem-solving capabilities that allow them to determine what modules are appropriate to solve a problem or answer a question.

9.2. System Description

The IMS organization is being defined using ODL: Organization Design Language (Fox, 1979b). Using this language, the architecture of IMS is defined in terms of the following language constructs:

{{ **Module**
GOAL:
Restriction: (SET (TYPE # is-a Goal))
VIEW:
Restriction: (SET (TYPE # is-a Goal))
PORT:
Restriction: (SET (TYPE # is-a Goal))
}}

A **module** is the main construct of an organization. It consists of procedures, data structures and meta-descriptions of the purpose of the module. A **goal** is a module meta-description. A module may have many goals. It describes resources, procedures and data usage. It also defines the goal's importance to the module, the certainty with which it is achieved and its behavior. The instantiation of a module's goal(s) results in a module's execution. A *view* defines the goals of a module and regions of a data-module that can be seen externally by other modules and data- modules. It is a window through which other modules can view it. A *port* is the part of a module or data-module to which the channel attaches. It defines the views available to a channel, message type and direction of information flow.

A **data-module** is a set of memory locations with a defined structure, and access and modification routines. It also supplies monitoring routines to inform other modules of the appearance of specified data.

A **Channel** is a pipe or link which connects modules and data modules. It specifies what information can flow along it and its direction.

A **Message** is a specification of the format and content of messages communicated in the organization.

Last, an **Organization** is a template definition of a structure composed of a set of modules, data modules, channels, etc. Organizations have the same meta-descriptions as a module and can be viewed as primitive constructs.

The purpose of the **Module** description is to provide capability information to the rest of the system. These attributes are used by MIPS, UIPS, and TIPs to determine whether to use the module for a particular task. **Organization** and **Channel** descriptions define the architecture of the system.

{{ **Goal**
PRECONDITION: <data-model>
POSTCONDITION: <data-model>
RESOURCE-CONSUMPTION:
(Type, Source, Number, <restrictions>)*
RESOURCE-PRODUCTION:
(Type, Number, <restrictions>)*
RESOURCE-TRANSFORMATION:
(<resource>* --> <resource>*, <certainty-transform-proc>,
<description>)*
INITIATION: (<message>,<port>)*
GOAL-MODEL: <model-name>
PORTS: <port-name>*
OBJECTS: (<procedure-name>*, <data-structure-name>*)
DATA-MODELS: <data-model-name>*
ORGANIZATIONS-MEMBERSHIP: (<organization-name>, <role-name>)*
UTILITY: <procedure>
MAILBOXES: (<mailbox-name>, <procedure-name>*)*
}}

{{ **View**
GOAL:
Restriction: (SET (TYPE is-a Goal))
DATA:
Restriction: (SET (TYPE is-a Data))
PROCEDURE:
Restriction: (SET (TYPE is-a Procedure))
MAILMAN:
Restriction: (SET (TYPE is-a Mailman))
}}

{{ **Mailman:**
MESSAGE:
GOAL:
MAILBOX: }}

{{ **Port**
DIRECTION:
Restriction: (OR Input Output)
MESSAGE:
Restriction: (SET (TYPE is-a Message))
Cardinality: (1 INF)
CONNECTION-RESTRICTIONS:
Restriction: (SET (TYPE is-a connection))
VIEWS:
Restriction: (SET (TYPE is-a view))
}}

{{ **Connection**
CHANNEL:
MODULE:
MESSAGE: }}

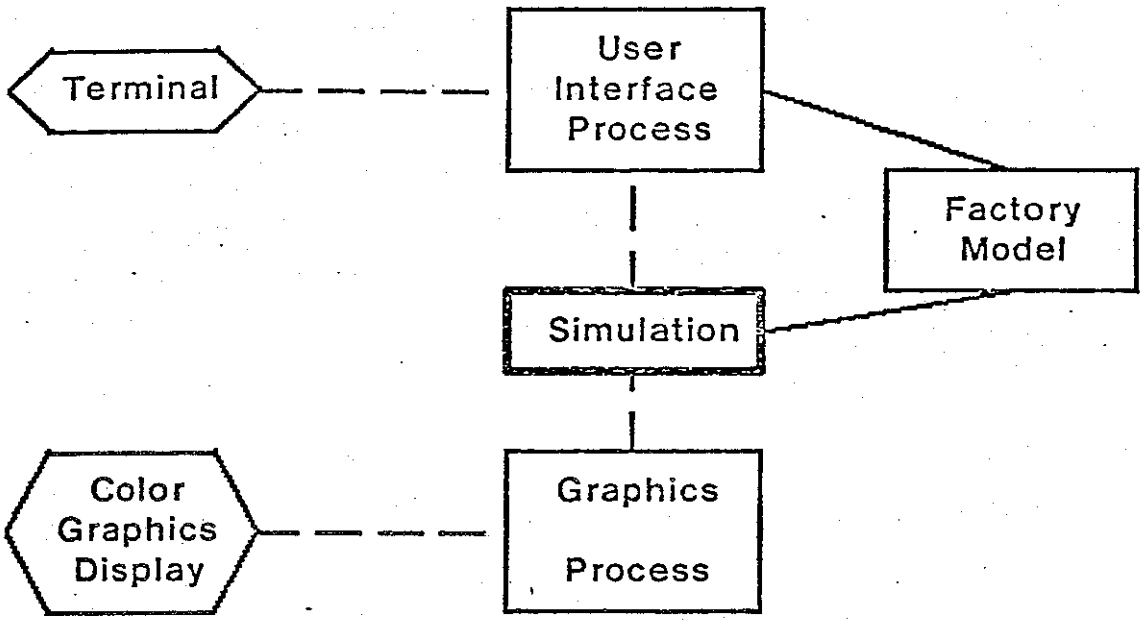
9.3. Data Accessability and Caching

When a module is instantiated as a process, a process schema (description) is created, defining who created the process and what rights it has in the current environment. A process has its own local data space which contains the information necessary to carry out it's task. When a process requires information not defined locally, it uses its process schema and the module descriptions of other processes to determine where the information may reside. Once having determined accessable processes, it sends messages requesting the information. Upon receipt of the information, it is cached in the process's local data space. This technique is used by the simulation system to acquire only the parts of the factory model it needs for a simulation (figure 9-1).

10. Conclusion

Most of the costs of producing products in complex organizations are attributed to overhead, much of which is comprised of managerial, professional, and salaried personnel. If significant productivity gains are to be made in this decade, more attention must be paid to aiding both the professional and the manager. The Intelligent Management System is a step towards this goal. By combining artificial intelligence, computer science, and management science techniques, more intelligent aids and solutions for the operation and management of complex organizations can be found.

Our focus in IMS has not been the creation of an information system to support decision making as



- Simulation initiates Graphic Display of Factory
- Executes Simulaion
- Displays Simulation Progress
- User Interface Process analyzes results and outputs answers

Figure 9-1: Simulation Model Caching

typically found in decision support research (Alter, 1979), but to explore more ill-structured problems (Simon, 1960) by means of heuristic problem-solving systems. Ten years ago the promise of artificial intelligence techniques was not clear, but advances in the last decade have resulted in systems that display surprising capabilities and in some cases perform better than experts in the field. Examples can be found in Computer Layout: R1 (McDermott, 1980); Medical Diagnosis: Mycin (Shortliffe, 1976), Internist (Pople, 1977); Geological Analysis: Prospector (Duda et al., 1978); Speech Understanding: Hearsay-II (Erman et al., 1980), Harpy (Lowerre, 1977). It is now time to re-explore the application of artificial intelligence to management problems. That is the goal of the Intelligent Management System.

11. Acknowlegements

IMS is the result of the efforts of many people in the Intelligent Systems Laboratory. They include Brad Allen, Don Cohen, Bryan Griffin, Jenny Hood, Carol Johnson, Joe Mattis, Drew Mendler, Steve Miller, Tom Morton, Ramana Reddy, Gary Strohm, Ari Vepsalainen, and Mark Wright. I also wish to thank Jaime Carbonnel, Bill Ferguson, and John McDermott for their comments on earlier drafts of this report.

12. References

Ayres R.U., (1980), "Growth Risk and Technological Choice", *Technology in Society*, Vol. 2, pp. 413-431.

Alter S.L., (1980), *Decision Support Systems: Current Practice and Continuing Challenges*, Reading, MA: Addison-Wesley Pub. Co.

Bartlett F.C., (1932), *Remembering*, Cambridge: Cambridge University Press.

Bobrow D., and T. Winograd, (1977), "KRL: Knowledge Representation Language," *Cognitive Science*. Vol 1, No. 1, 1977.

Buneman O.P., H.L. Morgan, and M.D. Zisman, (1977), "Display Facilities for DSS Support: The DAISY Approach", *Proceedings of a Conference on Decision Support Systems*, SIGBDP, Vol. 8, No. 3, 1977.

Carbonell J., and Phil Hayes, (1981), "Multi-Strategy Construction-Specific Parsing for Flexible Data Base Query and Update", *Proceedings of the Seventh International Joint Conference on Artificial Intelligence*, Vancouver B.C., Canada, August 1981.

CMU-CSD, (1979), "Proposal for a Joint Effort in Personal Scientific Computing", Computer Science Department, Carnegie-Mellon University, Pittsburgh PA.

Duda R.O., P.E. Hart, P. Barrett, J.G. Gaschnig, K. Konolige, R. Reboh, and J. Slocum, (1978), "Development of the Prospector Consultation System for Mineral Exploration: Final Report", Tech. Rep., SRI International, Menlo Park CA, Oct. 1978.

Erman L.D., F. Hayes-Roth, V.R. Lesser, and D.R. Reddy, (1980), "The Hearsay-II Speech-Understanding System: Integrating Knowledge to Resolve Uncertainty", *Computing Surveys*, Vol. 12, No. 2, June 1980, pp. 213-253.

Forgy L., (1981), "The OPS5 User's Manual", Technical Report, Computer Science Department, Carnegie-Mellon University, Pittsburgh PA.

Fox M.S., (1979a), "On Inheritance in Knowledge Representation", *Proceedings of the Sixth International Joint Conference on Artificial Intelligence*, Tokyo Japan.

Fox M.S., (1979b), "Organization Structuring: Designing Large, Complex Software", Technical Report CMU-CS-79-155, Computer Science Department, Carnegie-Mellon University, Pittsburgh, PA.

Fox M.S., (1981), "SRL: Schema Representation Language", Technical Report, Robotics Institute, Carnegie-Mellon University, Pittsburgh PA, in preparation.

Galbraith Jay, (1973), *Designing Complex Organizations*, Addison-Wesley.

Hayes-Roth F., and V. Lesser, (1977), "Focus of Attention in the HEARSAY-II Speech Understand System," *Fifth Int. Joint Conf. on Aritificial Intelligence*, Cambridge, MA, Aug. 1977.

Lenat, D., (1976), "AM: An Artificial Intelligence Approach to Discovery in Mathematics as Heuristic Search," (Ph.D. Thesis) Computer Science Dept., Stanford University.

Lowerre B., (1976), The HARPY Speech Recognition System, (Ph.D. Thesis), Tech. Rep., Computer Science Dept., Carnegie-Mellon University, Pittsburgh PA.

March J.G., and H.A. Simon, with H. Guetzkow, (1958), *Organizations*, New York: John Wiley and Sons.

McDermott J., (1980), "R1: an Expert in the Computer Systems Domain", *Proceedings of the First Annual National Conference on Artificial Intelligence*, Stanford University, Aug. 1980, pp. 269-271.

Miller S., (1981), "Expansion Problem and Batch Production Technology", Technical Report, Department of Engineering and Public Policy, Carnegie-Mellon University, Pittsburgh PA.

Minsky M., (1975), "A Framework for Representing Knowledge", In *The Psychology of Computer Vision*", P. Winston (Ed.), New York: McGraw-Hill.

Pople H., (1977), The Formation of Composite Hypotheses in Diagnostic Problem Solving: An Exercise in Synthetic Reasoning. *Proceedings of the Fifth International Joint Conference on Artificial Intelligence*, Cambridge, Aug. 1977.

Rashid R.F., (1980), "An interprocess communication facility for UNIX", Technical Report CMU-CS-

80-124, Computer Science Department, Carnegie-Mellon University, Pittsburgh PA, February 1980.

Reddy Y.V., and M.S. Fox, (1981), "Knowledge-Based Simulation", Technical Report, The Robotics Institute, Carnegie-Mellon University, Pittsburgh PA, in preparation.

Robertson G., D. McCracken, and A. Newell, (1981), "The ZOG Approach to Man-Machine Communication", *International Journal of Man-Machine Studies*, Vol. 14, pp. 461-488.

Shortliffe E.H., (1976), *Computer-Based Medical Consultations: MYCIN*, New York: American Elsevier.

Simon H.A., (1960), *The New Science of Management Decision*, New York: Harper and Row.

Williamson O.E, (1975), *Markets and Hiearchies: A Transactional and Antitrust Analysis of the Firm*, New York NY: The Free Press.

I. SRL: Schema Representation Language

SRL has its basis in schemata (Bartlett, 1932), which have come to been known as frames (Minsky, 1975), Concepts (Lenat, 1976), and Units (Bobrow & Winograd, 1978). For historical and environmental reasons, I will use the term schema to refer to the basic representational unit.

The role of a schema in SRL is both confluential and relational. A schema is the confluence of one or more conceptual entities. Consider the problem of representing the concept of a dog. A dog can be viewed as a mammal. Hence, "dog is a mammal" is a unique conceptual entity tied to the dog schema. From the mammal schema, dog may inherit attributes such as "warm blooded". A dog can also be viewed as a guard. Hence, "dog is a guard" is a second conceptual entity associated with the dog schema. From the guard schema, dog may inherit attributes disjoint from or in conflict with attributes from mammal. A dog can also perform the operation of fetching which is unique to a dog, and unrelated to "dog is a mammal" and "dog is a guard" (fetch like a dog). Hence, a third unique entity has appeared, that is, "dog is a dog". The mistake that all representations make is to mix the attributes associated with the different entities together in the dog schema. In SRL, an attempt has been made to prevent this mixing by representing each view as a schema. Hence, a schema is created for "dog is a mammal", "dog is a guard", and "dog". The first two schemata are indexed in the dog schema; attributes inherited by dog from "dog is a mammal" are stored in "dog is a mammal", and not in "dog". Only attributes unique to the concept of a dog are stored in "dog". This removes the "residency problem"; where to put an attribute inherited from more than one view.

The conceptual entity, "dog is a mammal" is represented by a schema. This schema is a *relational* description of how a dog is related to the mammal schema. It defines what attributes of mammal can be inherited and whether any modifications are to be made.

Physically, a schema is composed of a schema name and a set of slots. Each slot has a set of associated facets. Conceptually, the schema contains relations between it and other schemata. It also has attributes unique to the schema. Attributes, structure, and relations are represented by slots.

The type of attribute, structure or relation is represented by the slot name. In SRL, all slots are relations. That is, the value of the slot is related to the schema by the slot name.

{{ Platypus
IS-A: Mammal }}

In this example, a schema is enclosed in double braces. The name of the schema appear first followed by the slots and their values. There is one slot, which is an IS-A relation, relating **Platypus** to **Mammal**. Now the value of a slot is only one *facet*, other *facets* such as *type*, *restriction*, *default*, etc., when presetn will be displayed as:

{{ **Platypus**
IS-A:
Type: relation
Value: Mammal
}}

Hence, a platypus schema would look like:

{{ **Mammal**

NURSING-METHOD:
Value: Breast
Type: attribute
BIRTH-PROCESS:
Value: Live
Type: attribute
COLOR:
Type: attribute
HEAD:
Type: attribute }}

The **Mammal** schema is composed of four slots, two of which have values which are unique to the

```
{{ Platypus

    { IS-A Mammal
        BIRTH-PROCESS:
            Value: lay-egg
        COLOR:
            Value: brown }
}}
```

Mammal schema. Tied to the **Platypus** is a schema that represents the concept of "platypus is a mammal". It has an IS-A relational-schema to the **Mammal** schema. The relational-schema for "Platypus is-a Mammal" is printed in single braces, with the realtion type and target at its head (i.e., is-a mammal), and any information unique to the relation following. This allows **Platypus** to inherit slots and their values from **Mammal**. When viewing a **Platypus** as a **Mammal**, its BIRTH-PROCESS is lay-egg, and its COLOR is brown. Note that those attributes are stored in the "Platypus is a mammal" schema and not directly in **Platypus**. The rest of the chapter will describe schemata in depth.

In general, a schema will be "pretty printed" as follows:

II. System Particulars

IMS is programmed in Franz-lisp on a VAX-11/780 running the UNIX operating system. Processes communicate using the Interprocess Communication Facility (Rashid, 1980). The color display is a Grinnell system.

```
{{ <schema name>
   { <RELATION> <related schema>
        <SLOT NAME>:
             <facet>: <filler>
                       [<meta-filler-type: <meta-filler>]
                            .
                            .
                            .
             <facet>: <filler>
                  .
                  .
                  .
        <SLOT NAME>:
             .
             .
             .
   }

   <SLOT NAME>:
        .
        .
        .
   }}
```