HEWLETT PACKARD

Simulation Modeling of a Manufacturing Enterprise with Complex Material, Information and Control Flows

M. Shahid Mujtaba
Instruments & Photonics Laboratory
Manufacturing Systems & Technology Department

HPL-93-25
March, 1993

modeling, simulation, enterprise modeling, enterprise simulation, Order-to-Ship process, object-oriented simulation, manufacturing enterprise modeling, manufacturing modeling, object-oriented modeling

This paper describes practical problems encountered in modeling and simulating complex system interactions in a manufacturing enterprise. In addition to Production, the interactions of diverse activities such as Sales Forecasting, Order Processing, Production Planning, Material Requirements Planning, Material Procurement and Distribution are considered. The preliminary results of such modeling and simulation are described. A graphical model of the Order-to-Ship (OTS) process, which consisted of the activities that occurred between the receipt of orders and the shipment of products within a factory, was built using Hierarchical Process Modeling (HPM), derived from IDEF-0. Simulation model components for the OTS process were built using Object-Oriented Programming and Discrete-Event Simulation concepts. These reusable components were implemented in the class structure of the Common Lisp Object System (CLOS) within a Manufacturing Enterprise Simulator (MES), on which different Manufacturing Enterprise Models (MEMs), including the OTS model, could be executed. Simulating factory operations with the OTS model to generate system performance measures such as Order Backlog and Finished Goods Inventory provided preliminary results on why customers experience delivery delays when the manufacturing enterprise has plenty of finished goods on hand.

Published in part as "Systems with Complex Material and Information Flows" *International Conference on Object-Oriented Manufacturing Systems* (ICOOMS) Calgary, Canada, May 1992
Published in part as *ibid*, HPL-92-134
To be published in *International Journal of Computer Integrated Manufacturing* (IJCIM), 1993

Contents

1 INTRODUCTION

1.1 Definition of the Problem

On-time delivery and reducing inventory cost are two factors that are traded off in many manufacturing organizations. Customers become dissatisfied if the time from placement of an order to receipt of the shipment is too long. At the same time a global manufacturing enterprise can have significant amounts of money tied up in inventory that is in the wrong place or in the wrong product, while the long delivery time translates into large order backlogs. This dilemma was the primary problem under consideration. Understanding the reasons behind delivery delays and excessive inventory motivated this research. This project attempted to answer questions such as:

- Why do customers receive their complete orders in a matter of weeks when manufacturing cycle times have been reduced to half a day?
- Why is product availability not immediate when there is plenty of finished goods inventory on hand?
- What is the best delivery performance that can be expected given a manufacturer's current operating conditions?
- Where does attention need to be focused to reduce delivery time?

1.2 Technical Objectives and Research Challenges

An early project decision was to build a simulation model that addressed the above questions on an ongoing basis. The expectation was that better decisions would come from greater system understanding.

Computer simulation is a well-accepted engineering methodology in the design of physical systems such as computer systems, chemical processes or airplanes. It has been applied extensively to the production floor level, but at the time research plans were being formulated, none of the existing tools identified were directly applicable for building a simulation model at the enterprise level. The primary shortcoming of existing simulation tools at the time was the lack of suitable concepts that described the manufacturing enterprise at the appropriate level of abstraction to answer the questions above. This lack of applicable tools presented both a challenge and an opportunity to contribute towards a new area of research.

The research goal was to show the feasibility of applying simulation methodology at the broader level of the enterprise.

1.2.1 Scope of the Domain

The revenue for the Hewlett-Packard (HP) Company for the year ended October 31, 1992 was $16.4 billion. There are Product Development, Manufacturing and Marketing Organizations in twenty-nine cities in three countries in North America, eleven cities in seven countries in Europe, and twelve cities in ten countries in the rest of the world. In addition, there are approximately 600 HP Sales and Support Offices and Distributorships in 110 countries throughout the world (Hewlett-Packard 1992).

Trying to model the HP company was tempting, but a giant undertaking. Knowledge acquisition activities would be laborious and time consuming. Therefore, after we had generally defined the problem, talked to people in different sites and determined that the problem was applicable to different sites, we selected one representative site with the hope that its model could be expanded to others. The scope of the domain was limited to the activities in a single manufacturing site at HP from receipt of orders to shipment of those orders: the Order-to-Ship (OTS) process (Mujtaba 1992a).

The focus of this OTS model was to address the class of problems that generated the questions in section 1.1. The focus of the work was not on decision analysis research, enterprise integration, Computer Integrated Manufacturing (CIM) or better manufacturing methods, but on creating the ability to understand system performance.

The terms "we" and "researchers" in this paper refer to the author and his colleagues at Hewlett-Packard Laboratories (HP Labs) in Palo Alto, California, who were involved in the enterprise modeling and simulation activities described in this paper. The term "domain experts" refers to the people familiar with the functions and processes of the selected manufacturing site at HP. They were from a variety of different disciplines and were more knowledgeable about domain, process and business issues than about modeling and simulation issues. The term "knowledge acquisition" refers to the process of the researchers to obtain and document knowledge and information from the domain experts.

In attempting to build the OTS model, we talked to domain experts whose jobs were affected by the OTS process. Our early discussions led to the following observations:

- Because of its size and complexity, no single expert understood all the parts of the structure and behavior of the organization under study. This meant that the researchers would need to interact with multiple experts and reconcile different points of view.
- An explicit graphical model for documenting the details of system behavior was crucial to creating good communication and obtaining consensus among the domain experts and the researchers.
- The manufacturing process needed to be studied at a higher level and broader scope than the production or shop floor level.
- In addition to the flow of material that most simulation models address, its interaction with flows of information and control initiated by different parts of the organization was also important.

1.2.2 Current Practice

There is extensive literature on the simulation modeling process, e.g., Chapter 1 of Law and Kelton (1991), Chapter 1 of Pritsker (1986), Chapter 6 of McHaney (1991), and Law and McComas

(1991:21). The general consensus is that the purpose of the simulation modeling process is to clearly define a problem and to develop a model as a tool to understand and solve that problem.

Main of ALCOA (Norman et al. 1992:1004) stated that in ALCOA, the first models were "time-consuming to develop and most were applicable only to the original situation. When engineers went back to analyze the same areas, they found that the models had to be re-written."

There is an awareness that a model can be used to embody knowledge of a system rather than as a tool (e.g., Zeigler 1984). Funke (Norman et al. 1992:1004) from The Boeing Company stated that at Boeing, simulation has provided "a forum for the collection of process operating rules and assumptions in one medium as a basis to develop the model" of a process or system.

Other ongoing works on the application of models to embody knowledge at the enterprise level of manufacturing operations include TOVE (Fox 1992:176) and CIM–OSA (Jorysz and Vernadat 1990a:144, 1990b:157) . Pardasani and Chan (1992:182) describe the expansion of an infrastructure for creating simulation models based on the ISO reference model for shop floor production standards to create enterprise models.

1.3 Definition of Terms

An *entity* refers to something of interest in the real world. *Active* entities (e.g., persons) initiate actions, while *passive* entities are acted upon. A passive entity could be physical (e.g., a shipment) or informational. An informational entity could be data (e.g., a forecast) or a command (e.g., an order) requiring a response.

An *entity class* describes the general characteristics, behavior and state variables of all entities with those properties.

In this document, we define an *object* as a computer representation that has the general properties of object instances in the Object-Oriented Programming (OOP) paradigm. Since objects exist in the domain of the computer program, they may be used to represent both entities that exist and those that do not exist in the real world; examples of the latter include the simulation calendar and the event list.

An *object class* describes the characteristics, behavior and state variables of all objects with those behaviors. Individual objects have state information and characteristics that distinguish them from other objects in the class. Our objects possess class structure, inheritance, methods, instances and message-passing capabilities.

A *message* sent by one object to another causes the receiver to initiate, perform or complete some action, or provide a response to the sender, by invoking the appropriate method. The subject of the message, namely the arguments, could be a set of parametric values or a set of object instances. Where the objects represent entities in the real world, the message is the computer representation of actions, interactions or communications between those entities.

An *element* is the graphical or conceptual representation of an entity.

The *OTS Graphical Model* is the human-understandable conceptual model of the Order-to-Ship process under study. The *OTS Simulation Model* is the computer implementation of the OTS

Graphical Model. The *OTS Model* refers to the OTS Graphical Model and the OTS Simulation Model collectively.

1.4 Organization of Document

In Mujtaba (1992b:188), the author demonstrated how OOP expedited the computer implementation of the OTS Model. The purpose of this document is to discuss in greater detail the practical aspects of building the OTS Model, the preliminary results of the model and suggestions for future work. This author hopes that this discussion may benefit others who engage in a similar endeavor.

Since this document assumes that the reader is familiar with modeling and simulation and their benefits, and also with OOP, it does not describe these technologies or their general benefits in detail.

Section 2 describes the Hierarchical Process Modeling methodology and why we used it to build the OTS Graphical Model. Section 3 discusses the different types of components in the OTS process, and how these components map into HPM element types. Section 4 discusses the structure of the OTS Graphical Model, the knowledge acquisition activities undertaken to build it, and the strengths and weaknesses of the version of HPM we used. Section 5 discusses the conversion of the OTS Graphical Model into the OTS Simulation Model, and the steps required to move the model across different software and hardware platforms. Section 6 discusses how the HPM element classes map into reusable object classes and how the OTS model is one instance of a class of models called the Manufacturing Enterprise Model (MEM) that could be executed on a Manufacturing Enterprise Simulator (MES). Section 7 discusses results and recommendations for future work.

2 HPM METHODOLOGY

2.1 History and Background

Our decision to build a simulation model required finding a way to represent the domain concepts, activities and elements at an appropriate level of understanding for both the model builder and the domain expert. We found that the existing methodology for building simulation models that included elements such as queues, event generators and process nodes was a convenient graphical representation for the model builder but was inconvenient for the domain expert. Since communication with the domain expert was crucial to our project, we chose to forego convenience for the model builder in favor of clear communication.

We picked a convenient methodology, Hierarchical Process Modeling or HPM, to build the human-understandable form of the model. HPM was being developed internally at HP at the time, and one of our researchers contributed to its initial definition and development.

Marran et al. (1989:989) describes a new modeling methodology for large scale systems that "combines ideas from several well known techniques with significant innovations to obtain a powerful approach for information gathering, modeling, and ultimately, decision making. From IDEF-0 (Bravoco and Yadav 1985a:345), it took the idea of structured analysis of systems to develop hierarchical models of large organizations. From Data Flow (Ward and Mellor 1986), it took the idea of a context diagram that shows the relationship between a system and its environment. From Data Structured System Design (Hansen 1984), it adapted techniques for illustrating the hierarchical decomposition of flows of control, information, material and resources." Published descriptions on the use of this modeling methodology, subsequently named HPM, include its application to a manufacturing system (Marran et al. 1989:989) and the command and control system of a submarine (Fadali and Tacker 1990:58).

The Integrated Computer Aided Manufacturing (ICAM) program of the U. S. Air Force developed the IDEF (ICAM DEFinition) method to address particular characteristics of manufacturing. IDEF is composed of three modeling methodologies: Function model methodology (IDEF-0), information model methodology (IDEF-1), and dynamics model methodology (IDEF-2) (Bravoco and Yadav 1985a:345, 1985b:237). A large body of literature exists on IDEF, e.g., overviews of the methodology (Bravoco and Yadav 1985b:237), origin (Marca and McGowan 1988), assessment (Godwin et al. 1989:13), applications (Hughes and Maull 1985:34; Bravoco and Yadav 1985a:345, 1985c:59, 1985d:299; Le Clair 1982), and suggested enhancement (Shunk et al. 1986:12).

2.2 Notation

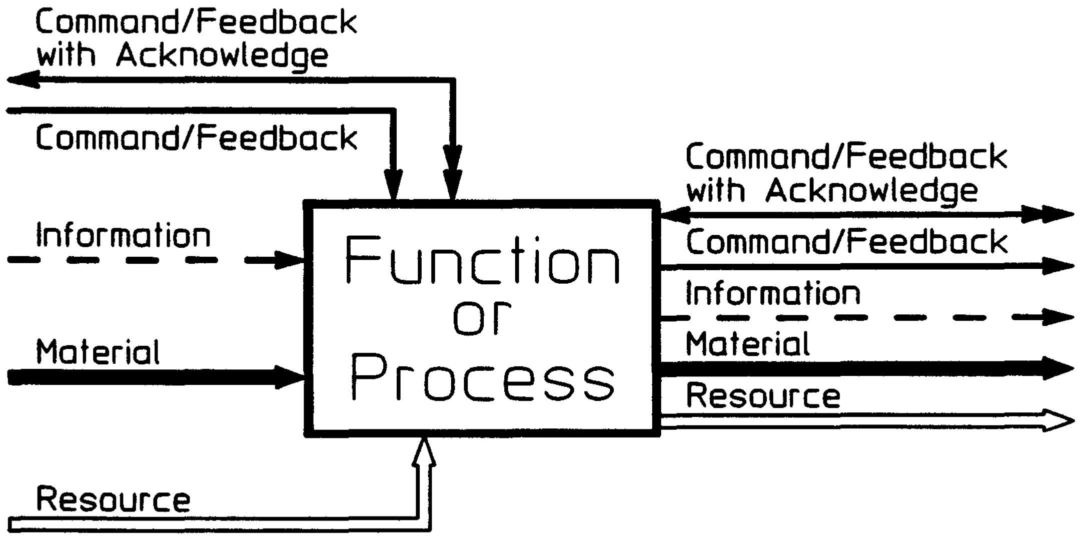

Figure 1: HPM Element Notation

Figure 1 shows the structure of the HPM element:

- The central rectangle represents the function or process under consideration.
- Heavy black lines represent flow of Material.
- Dotted blue lines represent flow of Information.
- Solid red lines represent flow of Command/Feedback. There are two kinds:

  - Without Acknowledge, represented by a single arrowhead at the commanded entity.
  - With Acknowledge, represented by single arrowhead at the commanding entity and double arrowhead at the commanded (or acknowledging) entity.
- Hollow green lines represent flow of Resources.

The reader familiar with the basic element of IDEF-0 will note the similarities with the HPM element. The combination of the color code and line type in HPM diagrams makes the type of flow obvious to the reader. When the different colors are not distinguishable, as in this document, the line type by itself is sufficient to indicate the type of flow. The above notation describes an early version of HPM that is used in this document. HPM has been undergoing development and continuing refinement, e.g., the line types or colors are not hard coded in HPM but can be customized by the user and continuous and discrete flows can be represented differently (Marran 1993). Additional details on HPM are also given in Hewlett-Packard (1988b).

3 OTS PROCESS ENTITIES

While real world entities can be described in a variety of different ways, we chose to classify them as passive physical, passive informational and active. This classification was made with a view to mapping the conceptual representations to real world entities in terms of HPM elements.

3.1 Passive Physical Entities

Passive physical entities are those that exist in the real world and are acted upon.

Products consist of components referred to as *parts*. Each *unit* of product or part could be considered a separate entity. Whether something is a product or a part is context dependent. One entity's products could be another entity's parts. We will use the term *material* to refer to either a part or a product when the distinction is not important. Material is stored in places like Finished Goods Inventory (FGI), Raw Parts Inventory (RPI) and Work In Process (WIP).

A *shipment* represents a set of product or part units that is actually moved, usually in response to an order. Material making up a shipment is considered part of in-transit FGI.

3.2 Passive Informational Entities

Informational entities refer to the content of information (e.g., a forecast), rather than to the medium (e.g., paper) on which the information is transferred.

Informational entities have different characteristics from physical entities: they are not homogeneous, their age (or vintage) is important, they can be duplicated in a way that physical objects cannot be duplicated, and they can be available in different places simultaneously. Information may be derived from previous information and has a shelf life; if the latest information is not available, the most recently available information can be used as an approximation.

A request for material is a command informational entity. A formal request is called an *Order* and has a required due date, earliest acceptable date, and latest acceptable date. An order entity can be considered something that moves between different organizational entities until the order meets the appropriate material, puts a marker on it, and then identifies the location of the material as the material moves to the order originator. A record of a set of order entities makes up the *backlog* (when the orders are received) or *part orders* (when the orders are generated).

Data informational entities describe status or information created or used by other entities. For example, *product structure* describes how a product is made up of component parts; it includes the parts list and assembly information. A *forecast* is a guess or estimate of the future action of others, and is updated periodically (e.g., an order forecast). A *plan* is a list of actions to be taken. It is derived from a forecast and knowledge of the current actual state. A *projection* is an estimate of the consequence of implementing the plan on the basis of the forecast.

3.3 Active Entities

An active entity can initiate actions and generally represents a function, a process, a person or an organization.

A *Vendor* accepts orders, adds them to its backlog and subsequently creates shipments of material. A *Customer* creates orders, sends them to some other entity and subsequently receives material requested in the orders. A *Factory* has characteristics of a vendor and a customer. It creates orders to obtain material that it transforms into products to be shipped to its customers. It keeps track of its orders as part orders. A *Distribution Center (DC)* creates orders to obtain the materials that it supplies to its customers. It is different from a factory since it does not transform the materials it receives. A *Carrier* accepts material shipments from supplying entities and moves them to accepting entities. The carrier makes no decisions about who gets which units. *Marketing* is responsible for the generation of forecasts that are used by the factory and DCs. *Research and Development (R & D)* determines the product structure and possibly the manufacturing processes used to produce the product.

3.4 Relationship Between Entities and HPM Elements

Active entities map directly to HPM elements. Passive entities map to the flows in HPM. Command informational entities map to the Command/Feedback flows in HPM. For our model, we chose to ignore flows of resources.

4 OTS GRAPHICAL MODEL

4.1 Description

The components of the HPM diagrams in this section correspond naturally to the entity classes described above. The description here is a brief summary of the HPM diagrams. The model is described in greater detail in Mujtaba (1992a).

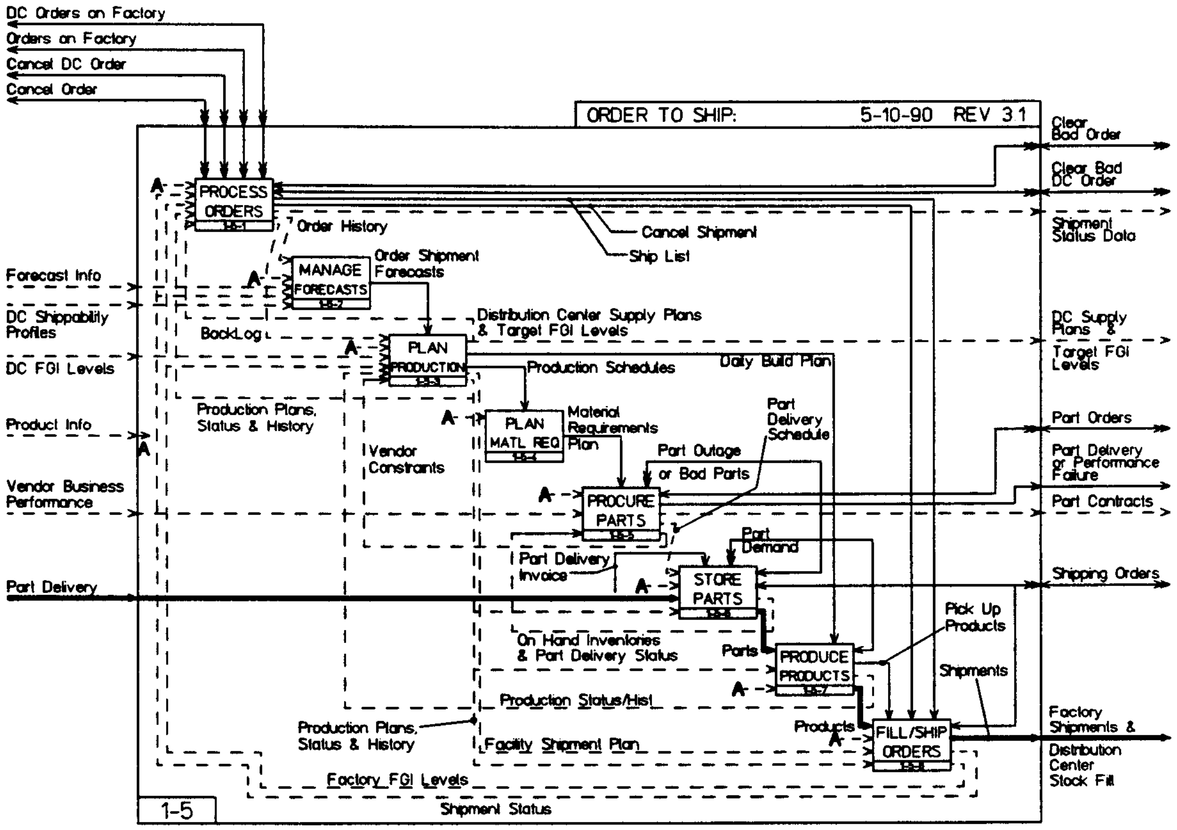

Figure 2: OTS Factory Block

Figure 2 shows the OTS block diagram of a typical factory, and is one way of decomposing the functions of manufacturing-related activities.

Figure 3 shows the Factory OTS block as a component of the SELL & DISTRIBUTE block. The actual active entities that comprise SELL PRODUCTS include the sales offices and all facilities where orders are taken verbally or through receipt of physical purchase orders. Orders received may be routed to the factory or one of the DCs, A, B or C. Part delivery is received only by the factory, whereas all DCs and the factory have outgoing shipments. The factory supplies products to the DCs through DC Stock Fills.

Figure 4 shows the context diagram or environment of the SELL & DISTRIBUTE block. The circular rather than rectangular blocks are an HPM convention to illustrate that in this diagram, the connections of the SELL & DISTRIBUTE block to the external blocks, rather than its internal structure, are of interest.

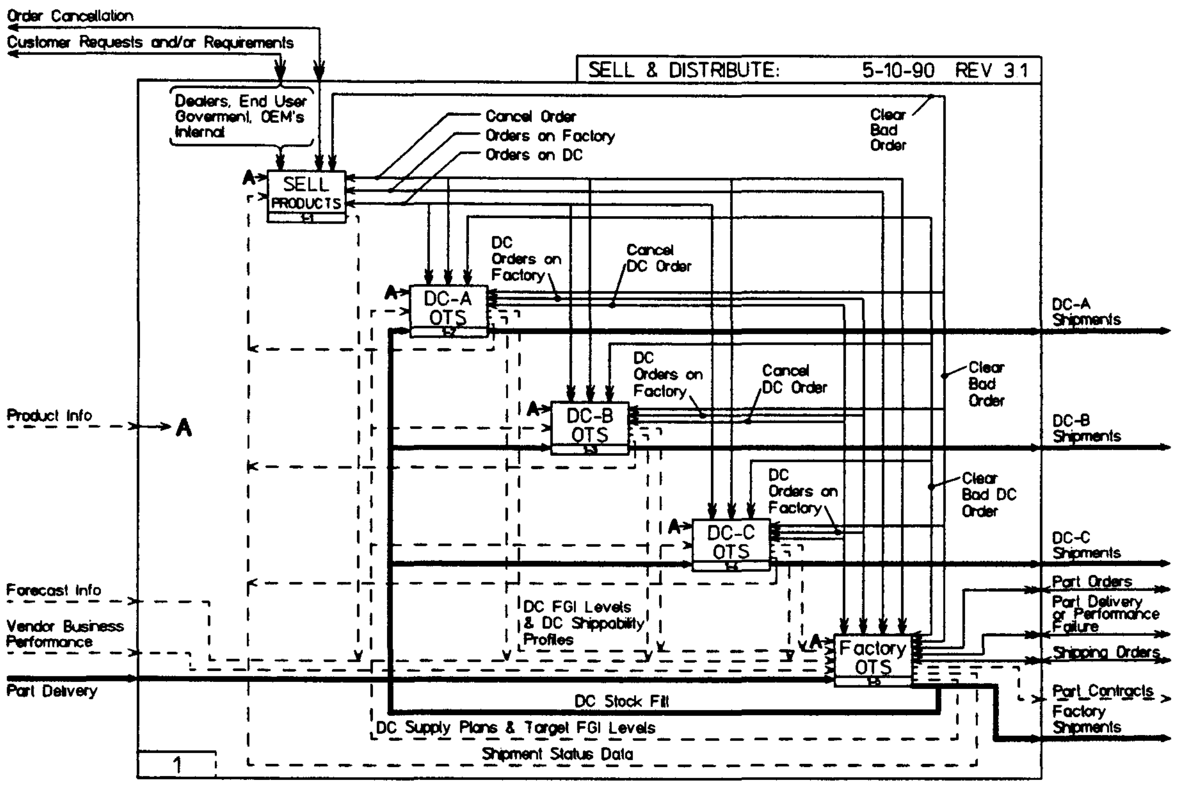

Figure 3: SELL & DISTRIBUTE Activity Block

CUSTOMERS interact with the SELL & DISTRIBUTE block through Customer Requests and/or Requirements and Order Cancellations. Forecast Info (an order forecast) is sent to SELL & DISTRIBUTE by MARKETING. Product Info is an informational entity that describes the product structure and is provided by RESEARCH & DEVELOPMENT. The outputs of SELL & DISTRIBUTE are Factory Shipments and DC Shipments to CUSTOMERS, and Part Orders (orders for parts, not partial orders) that cause parts to enter the system through Part Delivery from VENDORS. Shipping Orders represent interaction with CARRIERS and Part Contracts represent the periodic agreements made with VENDORS.

The description and structure of the model were guided by the scope of the problem we were addressing and the location of the domain experts. We studied a single factory with respect to a single product built with different options, and limited our model to reflect one product, even though the real factory produced multiple products. While it would be straightforward for the methodology to represent multiple factories or different kinds of operations at different levels of detail, at the time of knowledge acquisition we did not attempt to do so.

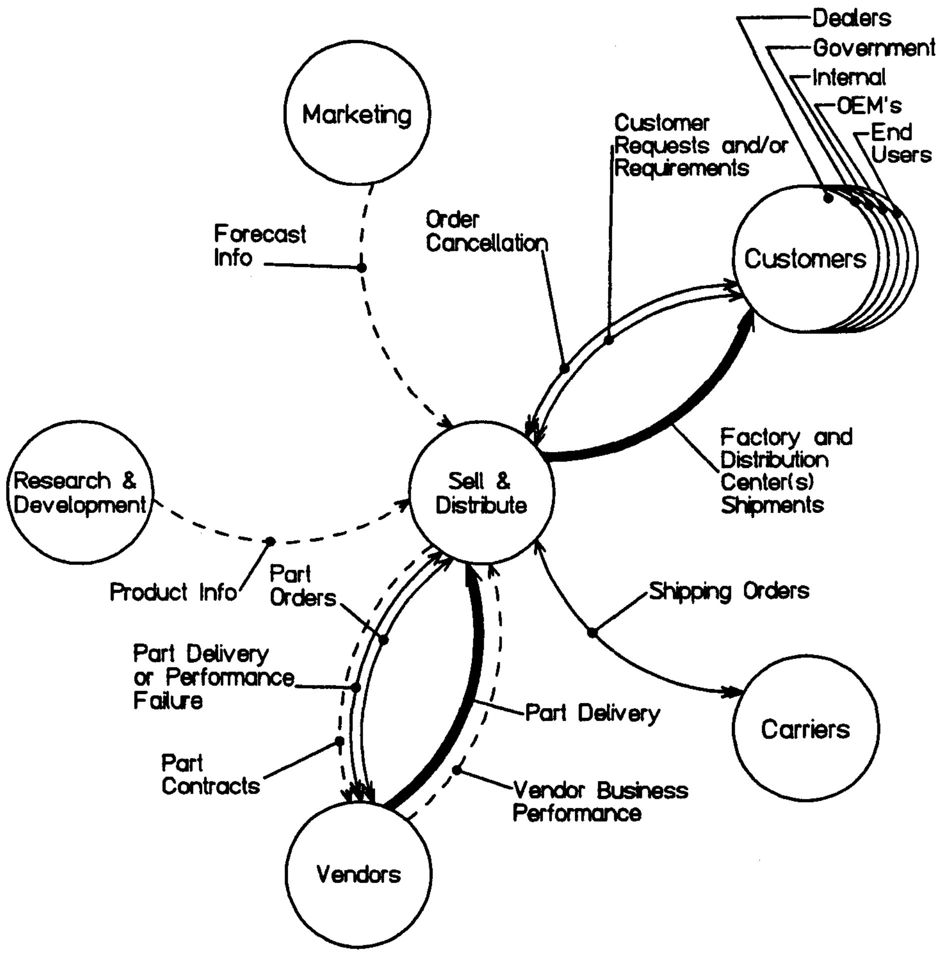

Figure 4: Context for SELL & DISTRIBUTE

4.2 Knowledge Acquisition Process

The knowledge acquisition process required the three project researchers to visit the factory (which was off-site from the office of the researchers) five times. The members of the project had formal training in various disciplines of engineering and recent training in simulation languages. One of the researchers was a charter member of the group that initially defined HPM.

The visits were spaced two to three weeks apart. On each two-day visit, we interviewed domain experts who represented the functions described in each sub-block of the OTS factory block. We took written notes and tape recorded each interview. On returning to the office, we rewrote our notes, relying on the tapes for additional details. The transcription had to be done by the interviewers, since a professional typist tended to transcribe verbatim, including extraneous filler words while leaving out important unfamiliar technical words. After writing up the notes individually, we reviewed our notes collectively to verify that they captured the essence of each interview.

In preparation for the first set of interviews, we generated a list of questions for starting the discussions. On second and subsequent interviews with each expert, we first confirmed our understanding of the previous interview by asking them to read, verify and correct our written notes. We then asked more detailed questions that arose from reviewing the notes. On the third interview, we also took the HPM diagram that was the precursor to Figure 2, on which we tried to capture the interrelationships among the functions graphically and from which Figures 2, 3 and 4 evolved.

We made two major personal observations during the course of the interviews:

- Once an interview session started, the experts provided a wealth of information, and it was all we could do to absorb even a portion of the information they provided, let alone process it. This was analogous to trying to drink from a firehose. There was no danger of running out of questions to ask the experts and the information they provided generated further questions.
- After the first interview, it was apparent that we had to consider not only the non-shop-floor manufacturing functions within the factory but also the environment in which the manufacturing entity existed, namely the sales offices and distribution centers. This realization led to the development of Figure 3, the SELL & DISTRIBUTE activity block.

The lack of a computer tool to support HPM at the time hindered the process of generating the block diagrams. All diagrams were drawn on a mechanical engineering drawing system (Hewlett-Packard 1988a) by one member of the project team, and consistency checks were done manually. We jointly defined a data dictionary to provide textual details of the blocks and flows in the diagram. We developed a simple program to check the consistency of the input, output, and block names and connections to help us verify that we did not miss or duplicate anything in the data dictionary.

4.3 Usefulness and Limitations of HPM

The methodology and notation used by HPM came in extremely useful during the knowledge acquisition process as a means of communication between people.

The HPM diagrams captured the flows of material, information and control very succinctly in one diagram, and also the relationships between the processes and functions in the organization. The mature and well-accepted technological IDEF-0 foundations of HPM were particularly conducive for describing processes, and the notation was quickly understood by the domain experts. The hierarchical nature of HPM helped to abstract different levels of detail to reduce complexity at each level. The flow of material, information and control fell naturally into the experience of the domain experts we interviewed, and the color coding in addition to the line type on the diagrams helped to distinguish each of the flows. The context diagram served to place a given block in its environment. It took about ten minutes of explanation of the HPM diagram and the notation to provide a comfortable level of understanding to domain experts previously unfamiliar with the notation. At that point, they could meaningfully discuss if the diagram made sense or if it was inconsistent with their views of the world.

We also found that each of the functions or processes did not fall directly into a discrete department of the organization. Some departments performed more than one function and some functions were performed by more than one department or by people in different departments. We were able to reconcile these differences by overlaying an organization map on the HPM diagram.

The limitations of HPM included the inability to specify temporal and behavioral information. It was not easy to capture algorithmic procedures, mathematical relationships, and timing delays and sequences on the diagram. All of this additional nondiagrammatic information had to be included in a separate supporting document. This separate document was an inconvenience only to implementing the simulation model. It was not an inconvenience to the domain experts who considered it part of the methodology.

Since no simulation or behavioral primitives were provided by HPM, the translation from the HPM notation to a simulation model was a laborious process of hand-coding.

5 OTS SIMULATION MODEL

5.1 Object-Oriented Implementation of the HPM Structure

We implemented all the entity classes of the representation and description of the OTS process as object classes. The object-oriented approach greatly facilitated not only the identification of entities and their behaviors in building the OTS Graphical Model but also their implementation as objects and the management of the objects in the system model. Minor differences between entities can be easily modeled by objects inheriting common behavior from the object class, preserving differences at the object instance level. Encapsulating behaviors within an object means that a change in internal behavior of an object can be localized without affecting the other objects in the system.

The primary object classes for active entities were Supplier, Factory, DC, Carrier and Customer. The primary object classes for passive entities were Orders, Parts, Product Structures and Current Value such as FGI, WIP and Order Backlog. Interactions between entities were modeled as messages between objects.

The model was initialized with the structural relationships of the active objects as described by the OTS Graphical Model. Passive objects such as orders and shipments were created and modified during the simulation.

5.2 Discrete-Event Simulation

According to Law and Kelton (1991), "*Discrete-event simulation* concerns the modeling of a system as it evolves over time by a representation in which the state variables change instantaneously at separate points in time. ... These points in time are the ones at which an event occurs, where an *event* is defined as an instantaneous occurrence that may change the state of the system." They mention *next-event time advance* and *fixed-increment time advance* approaches for advancing the simulation clock and state that the first approach is used by all major simulation languages and by most people coding their model in a general-purpose language.

They identify the following components in most discrete-event simulation models: System State, Simulation Clock, Event List, Statistical Counters, Initialization Routine, Timing Routine, Event Routine, Library Routines, Report Generator and Main Program. They state that although "there are now several very good and powerful simulation languages available, it is often necessary to write at least parts of complex simulations in a general-purpose language if the specific, detailed logic of complex systems is to be represented faithfully."

We found the last statement to be true. In general, discrete-event simulation supports the modeling of systems in which events occur according to some distribution of arrival, departure and processing times, with relatively simple computations occurring in between events. While we exploited discrete-event simulation concepts to manage the time behavior of the system for arrival, departure and processing times, access to a programming language to manipulate both symbols and numerical computations was imperative. For example, the production planning function required computations that could not be readily expressed in discrete-event simulation constructs but could be expressed conveniently in the form of an algorithm or a linear programming formulation.

We attempted to simulate the system using the simulation language SLAM II (Pritsker 1986) and found ourselves in the situation described above; the FORTRAN code required to represent the detailed logic of the complex system and the interfacing code to SLAM II dominated the programming effort.

We also attempted to simulate the system using the Knowledge Craft (KC) (Carnegie Group Inc. 1988) environment made up of the Carnegie Representation Language (CRL) and Simpak in conjunction with the KC OOP language, OOPak, on a Texas Instruments (TI) Explorer II computer. CRL used the concept of schemas, which provided the inheritance hierarchy, and OOPak provided the message-passing capability. Since CRL is embedded in Common Lisp (CL) (Steele 1990), we found that the code written in CL to represent the detailed logic of the complex system interfaced directly to the simulation, with no programming effort required for the interfacing.

Several factors forced us to reconsider our hardware/software platform. Our research computing environment was dominated by a network of HP workstations, and in-house system support and expertise for solving interconnectivity problems between the TI Explorer II and the network was scarce. When questions of future deployment of the simulation model within the company came up, using a non-HP platform required very strong justification.

It was clear that we needed to move to an HP hardware platform for continuing development, especially since a new generation of machines promised greater computational power. CRL, OOPak and Simpak were not available on the HP platform. We subsequently evolved to a system written in CL and Common Lisp Object System (CLOS) (Hewlett-Packard 1990) on the HP 9000 family of computers. While there were major differences between KC (CRL, OOPak and Simpak) and CLOS, the similarity in their concepts made the transition fairly smooth. We will first discuss the differences between conventional discrete-event simulation and our Object-Oriented Simulation and then discuss the issues relating to the platform migration in section 5.5.

5.3 Object-Oriented Simulation

The application of object-oriented methods to the field of simulation, particularly relating to manufacturing, has been reported extensively in the literature, e.g., Narayanan et al. 1992:59, Barros

Conventional Approach	Object-Oriented Approach
Simulation Clock: A variable giving the current value of simulated time.	Simulation Clock: A set of variables giving the current value of simulated time in seconds from some reference point, as well as in year, month, day, hour and minute. Simulation Calendar: An object that manages the Message Event List. It places messages to be sent in the future on the Message Event List. It identifies and sends messages due to be sent at the current time and takes them off the Message Event List. Dated Simulation Calendar: An object that knows about year, month, date, days of week, workdays, and weekdays and translates time to Simulation Calendar time units. Messages to be scheduled on the Simulation Calendar are sent to this object, which translates time to the appropriate units.
Event List: A list containing the next time that each type of event will occur.	Message Event List: A list containing the times when messages are to be sent to receiving objects. It is read by the Timing Routine and the Simulation Calendar, and written only by the Simulation Calendar.
Timing Routine: A subprogram that determines the next event from the Event List and advances the Simulation Clock to the time that event is to occur.	Timing Routine: A subprogram that determines the next time on the Message Event List and advances the Simulation Clock to that time.
System: A well-defined collection of entities. Entities are characterized by data values called attributes, and these attributes are part of the system state for the model.	System: A well-defined collection of objects. Objects are characterized by zero or more parameters, zero or more state variables, and zero or more methods. The object state variables are part of the system state for the model.
Event Routine: A subprogram that updates the system state when a particular type of event occurs (there is one event routine for each event type). It can create new events and interact with the Event List.	Object Method: A subprogram that responds to the message received by the Object (there is one method, which may be inherited from the method for the class, for each object message). These methods may respond to messages received from the Simulation Calendar or from other objects.

Table 1: Comparison of Conventional Discrete-Event Simulation and OO Approach Components

and Mendes 1993:53, Worhach 1992:281, Feng et al. 1992:291, Bhuskute et al. 1992:680, Pidd 1992:689, Luna 1992:694, and Shewchuk and Chang 1991:302.

Our object-oriented implementation included the components described in the previous subsection, with differences summarized in Table 1.

The first major difference between the conventional implementation and our object-oriented implementation was the concept of the Simulation Calendar object, which was the only object that could modify the Message Event List. The Timing Routine could look at the Message Event List, but could not modify it. No other object in the model or simulation system had access to the Message Event List. This was unlike the conventional approach where every event that was generated was entered into the Event List directly, and the Event Routine could directly access the Event List. The advantage of isolating the Simulation Calendar was that its implementation details were encapsulated and independent of the Object Method and the System, and could therefore be changed without affecting the System and Object Method codes.

The second major difference was that the Message Event List did not need to know or care about event types. It only knew about messages and objects. If new messages or objects were created dynamically during the course of the simulation, no modification was required to the Simulation Calendar or the Message Event List. In the conventional implementation, it was necessary to define every kind of event before the simulation started.

Certain periodic activities on a real calendar do not occur at regular intervals; e.g., events could occur on the sixth workday, second Tuesday, nineteenth day or last Friday of the month. Saturdays and Sundays are not workdays, but need to be considered when measuring elapsed time. Since calendar months are not equal in length, it is not possible to fit a fixed number of weeks into months. This creates overlapping cycles in which the overlap varies from cycle to cycle.

The management of calendar time and its implications were not handled by the objects representing the active entities of the model. Instead, we used the Dated Simulation Calendar (DSC).

Simpak used the concept of Simulation Calendar to manage an event list described in fixed units of time. By itself, it was inadequate to manage the above problem. However, we defined a DSC to handle the intricacies and subtleties of translating time expressed in calendar time to the underlying time representation in seconds and to interact with the Message Event List. The Message Event List consisted of a list of messages, objects and appropriate times at which the messages are scheduled to be sent.

All time-initiated events were generated by scheduler objects that sent messages periodically to the DSC to schedule messages to objects. There were different classes of scheduler objects for scheduling events every day, every workday, every week and same day each month, where same day could be the *n*th day, *n*th workday or *nth m*day of the month, where *m*day represents day of the week, such as Monday, Tuesday, etc.

All communication with the DSC was through messages. The Message Event List did not need to know about the contents of the messages; at the appropriate time, the Simulation Calendar sent the relevant message to the receiving object.

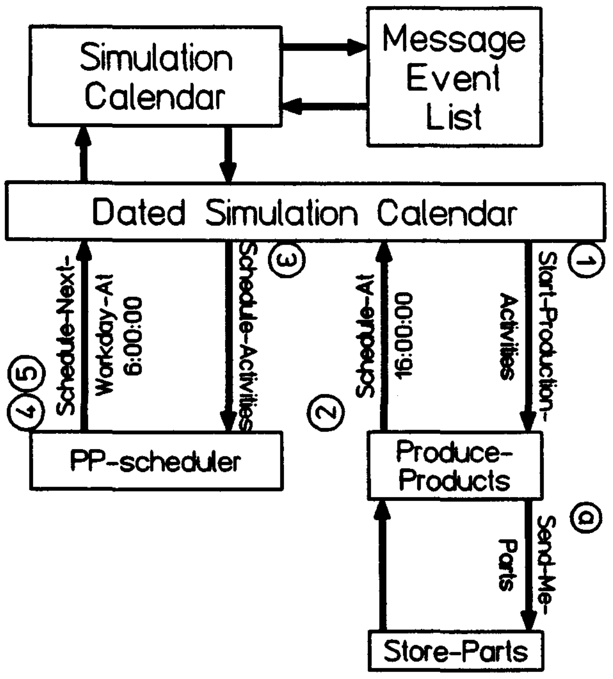

Figure 5: Message Flows Between Objects

5.4 Example

The example in Figure 5 illustrates message flows associated with scheduling events on the DSC and object interactions with one another. In the description, the left side of a colon represents an object, text in curly brackets {} represents Object Method behavior, text in parentheses represents CLOS function calls to send messages and text after semicolons represents comments for explanation. The number after the # symbol is the reference label in Figure 5.

This example shows the sequence of events that occurs every weekday morning at 6:00:00 am. PP-Scheduler is a scheduler object, and Produce-Products and Store-Parts are OTS elements shown in Figure 2.

```
;; currently it is 6:00:00 on a weekday

DSC:
    (send 'Produce-Products
      'Start-Production-Activities) ; #1

Produce-Products:
    {starts production activity such as looking at today's build}
    (send 'Store-Parts 'Send-Me-Parts
      '((part-a 3)(part-b 6))) ; #a
    ; Store-Parts sends back parts
    (send 'DSC 'Schedule-at '(16 0 0)
      Produce-Products 'End-Production-Activities)
      ; #2 – production activities last 10 hours

;; control goes back to DSC

DSC:
    (send 'PP-Scheduler 'Schedule-Activities) ; #3

PP-Scheduler:
    (send 'DSC 'Schedule-Next-Workday-At '(6 0 0) 'Produce-Products 'Start-Production-Activities)
      ; #4
    (send 'DSC 'Schedule-Next-Workday-At '(6 0 0)
      'PP-Scheduler 'Schedule-Activities) ; #5

;; all interactions at time 6:00:00 are now over, so the Time Advance Function then changes the
    Simulation Clock to the next time.
```

5.5 Migration Process from TI to HP Platform

Several issues required resolution to finish the migration across the platforms. First, CRL uses symbol names to refer to schemas, whereas CL and CLOS generally use variable names to refer to object instances. Second, CRL permits us to find all instances of an object from a given class, something that is not supported in CLOS. Both of the above issues were resolved by using a multiple inheritance capability in CLOS to define a class that during execution of its "make-instance" method generated a name and recorded the instance. Third, CLOS does not have a built-in simulation engine, so we needed to write our own.

The migration path for moving the model and simulation software was as follows:

- Model and Simulator on CL + KC on TI Explorer II Computer. This was the starting point for the migration process.
- Model and Simulator on CL + CLOS on TI Explorer II Computer. This was accomplished by replacing calls to CRL functions by calls to equivalent functions written in TI CL. Definitions of CRL schemas and instances were replaced by CLOS class definitions and instances, and messages sent to CRL schemas were replaced by messages sent to the corresponding TI CLOS objects. Functionality equivalent to the Simpak Simulation Calendar was implemented in TI CLOS. Calls to SimPak functions and methods were replaced by calls to equivalent functions and methods written in TI CL and TI CLOS.
- Model and Simulator on CL + CLOS on HP 9000 Series 300 Computer. This was accomplished by transferring the source code for the model and simulation from the TI Explorer II to the HP 9000 Series 300 computer and making minor modifications to allow for the differences in the CL and CLOS implementations of the two machines.

- Model and Simulator on CL + CLOS on HP 9000 Computer Family. This was accomplished by running the code with the appropriate version of CL and CLOS for the particular machine. At the time of writing, our model and simulator software has run on the HP 9000 Series 300, 400 and 700 computers.

6 MANUFACTURING ENTERPRISE MODEL AND SIMULATOR

6.1 Simulation System Components

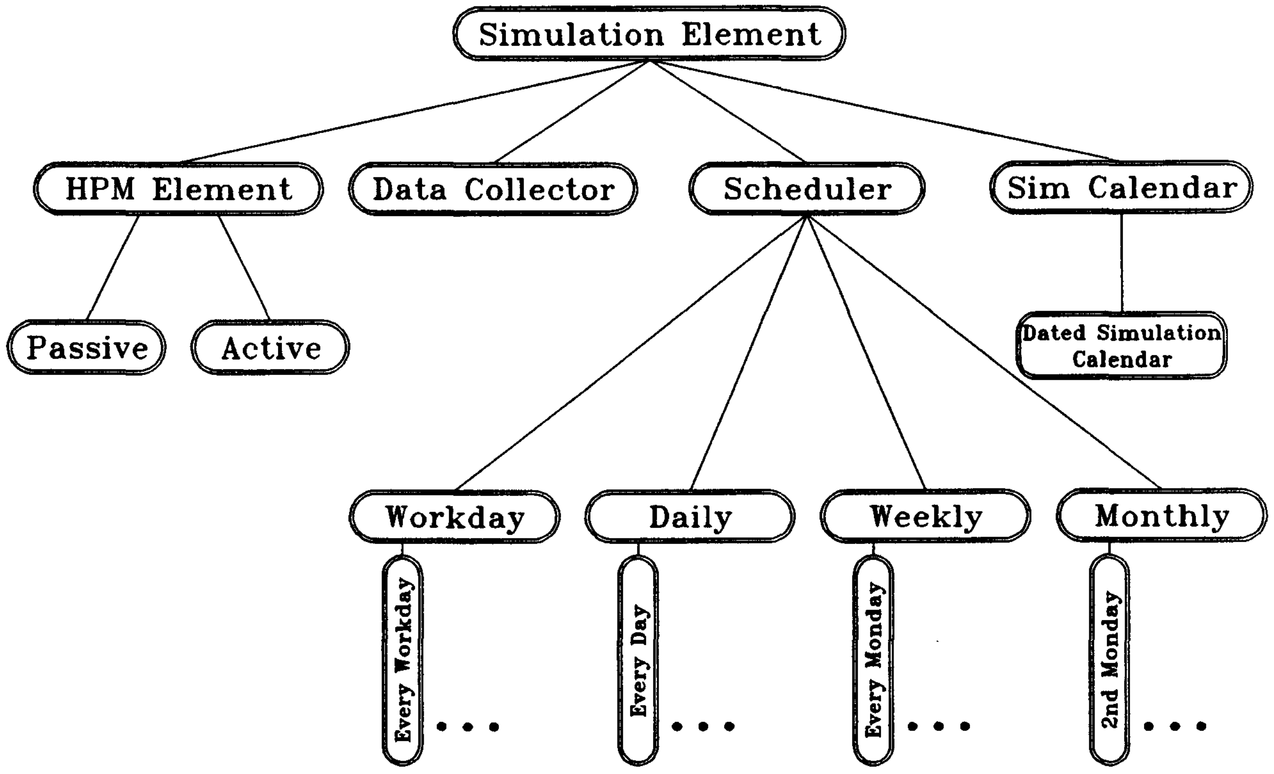

Figure 6: Classes of Simulation Elements

Figure 6 shows the four main classes of simulation elements used to build the simulation model:

- HPM element: Figure 7 gives a diagrammatic representation of the HPM element classes, which are mapped to CLOS classes to show the overall relationship between the class structures. The hierarchy is illustrative rather than complete. The pictorial symbols next to the classes represent the nature of the object class or the flows of the object class. The implementation of passive elements is straightforward and will not be discussed further. The interesting element is the generic OTS element, which we will discuss in more detail in section 6.2.

- Data Collector: By attaching a Data Collector to each physical and informational buffer and recording the values periodically, it is possible to keep the state history both of levels and of flows during a period. The history is kept not only of physical entities such as quantity of material but also of informational entities such as orders (both the number of orders and the amount of material they represent), forecasts and plans. By keeping a record of each transaction or movement of an order, it is possible to accumulate the total transactions during some specified period.
- Scheduler: The various kinds of schedulers are shown in Figure 6. The classes that were used in the simulation included workday, daily, weekly and monthly. An example was discussed in section 5.3.
- Simulation Calendar: The Dated Simulation Calendar is a calendar that is knowledgeable about calendar time, and has been discussed in greater detail in section 5.3.

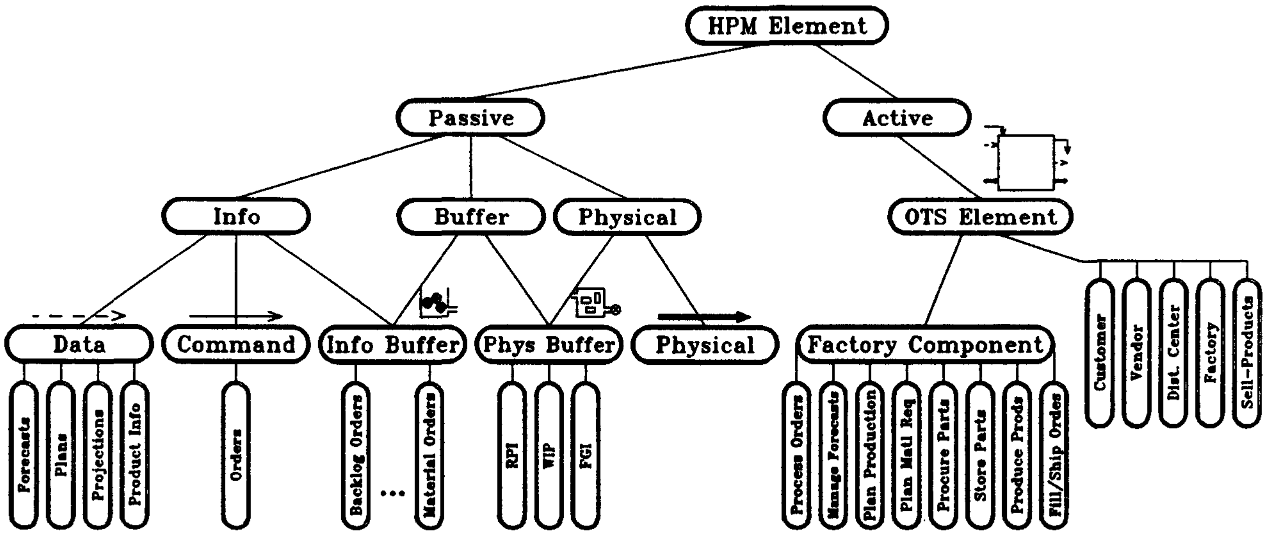

Figure 7: HPM Elements Mapped to CLOS Classes

6.2 The Generic OTS Element

Figure 8 is a graphical representation of the generic OTS element, a rich representation that captures the state information, behavior and action of an active entity. The arrows representing the flows into and out of the active entity are consistent with those of HPM. The generic OTS element is implemented as an object class in CLOS.

The generic OTS element consists of two main blocks, the Planning Block and the Operation Block. The Operation Block is further split into the Control and Material sections.

The heavy lines represent flow of material first into the Material section to RPI, then into WIP for assembly, and finally to a FGI from which finished goods are shipped out. The irregular-shaped

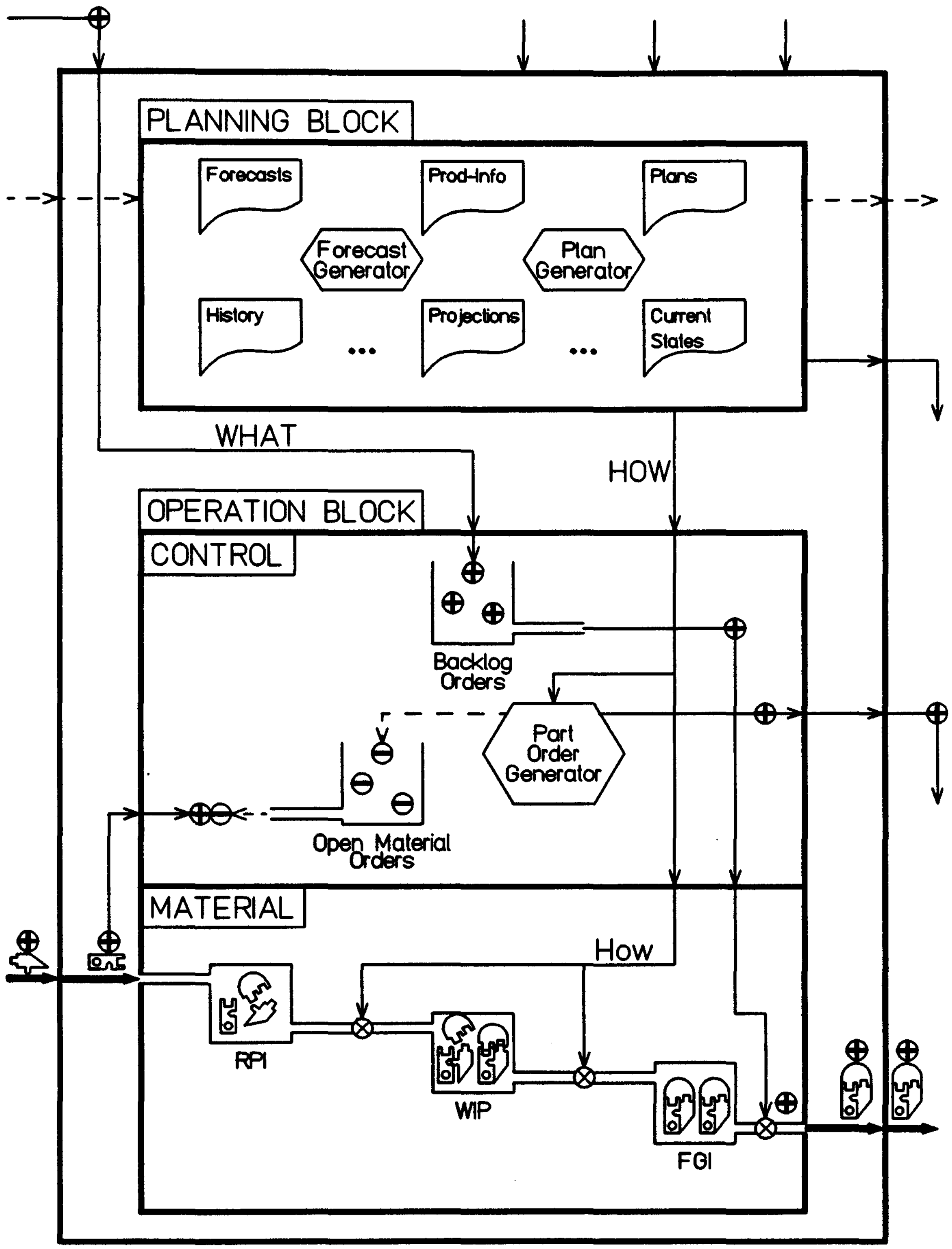

Figure 8: Generic OTS Element

objects next to the heavy lines represent the gradual transformation of different raw parts into complete assemblies in the WIP and their flow into the FGI. The flows can be controlled by valves (denoted as ⊗) at the outlets of each of the three material buffers.

Orders comprise one of the control flows. An order (represented as ⊕), as the argument of a command/feedback, comes into the OTS element, flows along the line marked "WHAT," and accumulates in Backlog Orders. Subsequently, the order moves out of Backlog Orders, controls the flow of units out of FGI, and attaches itself to those units to form a shipment. The shipment flows back to the left side of the OTS element initiating the order.

The activities in the Planning Block determine the quantities of products to build in current and subsequent periods, and therefore the amount of material flowing from RPI to WIP. In addition, they determine the quantities of different kinds of parts or material to procure in the current and subsequent periods. These quantities are sent to the Operation Block along the line marked "HOW" to the Part Order Generator, which creates a new Part Order and sends it to the appropriate supplying OTS element. At the same time, a marker (represented as ⊖) is accumulated in Open Material Orders until the physical shipment corresponding to the order is delivered. When the shipment is delivered (through the material flow on the left side of the OTS element), the order is separated from the physical material. The order is then combined with its marker and this results in closing out the Part Order (e.g., marking the Part Order ready for payment).

The other vertical solid lines coming into the top of the element represent commands that require a response but whose detailed path in not shown in the diagram. Similarly the solid line coming out of the Planning Block represents commands that are sent to other OTS elements.

Finally, the dotted lines represent information (e.g., forecasts, plans, etc.) flowing into the Planning Block from which new information, (e.g., forecasts, plans, etc.) are generated and sent out through the right side.

While the generic OTS element in Figure 8 is a generalized representation of a factory, it can also represent other active elements, e.g., a DC if its RPI and WIP were reduced to zero and units flowed directly into and out of FGI; a vendor if it had no Part Order Generator, RPI or WIP but included only Backlog Orders, FGI connected to an infinite source of material and a Scheduler that releases orders from the Backlog Orders; or production or assembly activity if it had only an Operation Block including WIP in the Material section where assembly is performed.

6.3 Translation Process

During the course of hand translating the OTS Graphical Model into the Simulation Model, several things became apparent that had not been anticipated.

Some model elements were being used over and over within the model (e.g., DCs, Vendors, FGI). These model elements had slight variations in different parts of the model, but they were broadly parallel to the HPM elements. These model elements could be configured to represent different organization structures, thereby providing a means for building models of different enterprises out of the same set of components or with slight modifications.

Non-time-delayed interactions occurred between the components represented by HPM elements (e.g., Send-Me-Parts and the parts sent in Figure 5) and Schedulers. In addition, Schedulers sent commands to HPM elements to send their state values to Data Collectors. These interactions were independent of the Simulation Calendar. On the other hand, all time-delayed interactions and activities always required interaction with the Simulation Calendar (e.g., Scheduler interaction and Start-Production-Activities in Figure 5).

6.4 Components of the MEM and MES

The distinction in the time relationship of the two kinds of interactions led to a natural split in the implementation of the simulation model. The part of the model relating to the OTS Graphical Model and its related Schedulers and Data Collectors (i.e., the non-time-delayed interactions) led to the Manufacturing Enterprise Model (MEM), which could be treated separately from the part that specified time delays and behaviors that could not be captured on the diagram and led to the Manufacturing Enterprise Simulator (MES).

The MEM is the computer implementation of a graphical model, together with the associated data collectors and schedulers. The MES is a collection of the CLOS class definitions of the simulation elements. In broad terms, the MEM describes the structural relationships and combinations of the model elements, and the MES describes the behavior and response of each of the model elements. The OTS Simulation Model is an example of a MEM.

6.4.1 Generating and Executing the Simulation Model

Figure 9 shows model creation and execution with particular reference to the OTS Model. The OTS Model Specs (short for specifications) is a computer-readable textual representation of the OTS Graphical Model which describes the structural relationships between the HPM elements, and the associated Data Collectors and Schedulers. The OTS Model Specs are given to the Model Generator (a piece of software), which creates the OTS-MEM, a instantiation of the OTS Model Specs, and the OTS-MES, an instantiation of the MES. Both these instantiations are computer manipulable, nontextual and nonhuman readable.

The OTS-MEM is a computer-manipulable form of the OTS Simulation Model, consisting of all instances of HPM elements in the OTS Graphical Model, a set of Dated Schedulers and a set of Data Collectors. For example, the part of the OTS-MEM corresponding to the OTS Graphical Model of the HPM element structure in Figures 2, 3 and 4 consists of a set of Vendors, Carriers, Customers, Marketing and R & D, and SELL & DISTRIBUTE. SELL & DISTRIBUTE is associated with Sell Products, which is associated with three DCs and one Factory. The Factory is associated with WIP, Backlog Orders and Open Material Orders, as well as Produce Products, Manage Forecasts, etc. The Produce Products is associated with a Produce Products Scheduler, as shown in Figure 5.

The OTS-MES is a computer-manipulable form that knows about the class definitions in the MES and the specific Simulation Calendar that is required by the OTS-MEM. The specific Simulation Calendar is the DSC because calendar time was important in the OTS model.

The OTS-MEM and the OTS-MES interact with each other during execution of the simulation.

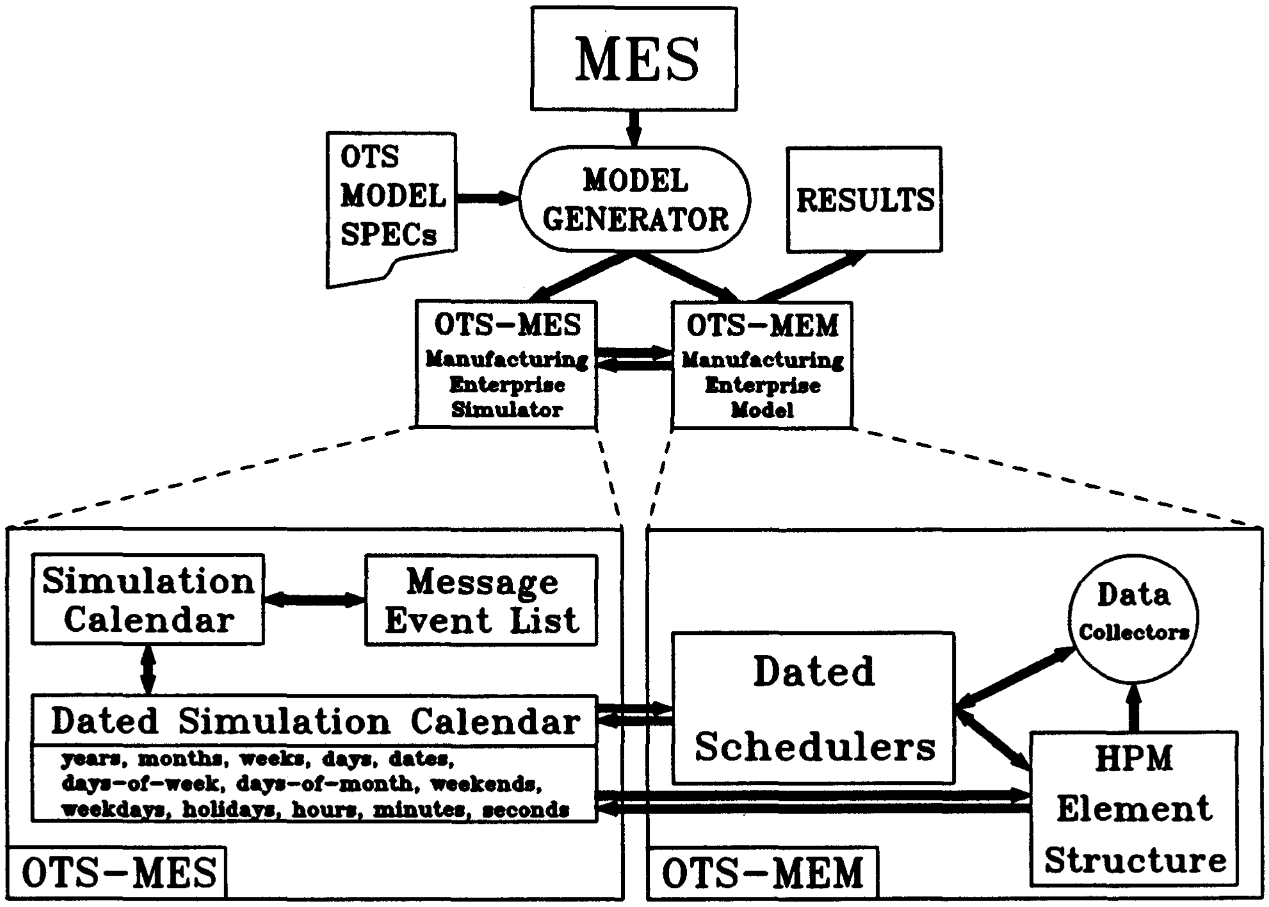
MES
OTS MODEL SPECs
MODEL GENERATOR
RESULTS
OTS-MES Manufacturing Enterprise Simulator
OTS-MEM Manufacturing Enterprise Model
Simulation Calendar
Message Event List
Dated Simulation Calendar
years, months, weeks, days, dates, days-of-week, days-of-month, weekends, weekdays, holidays, hours, minutes, seconds
OTS-MES
Dated Schedulers
Data Collectors
HPM Element Structure
OTS-MEM

Figure 9: Creation and Execution of OTS Simulation Model

All time-delayed interactions and activities occur between the MEM and MES, and the non-time-delayed interactions and activities occur among and within the three main components of the OTS-MEM. The behavior of each element of the HPM element structure, each Data Collector and each Data Scheduler is governed by its respective definition in the OTS-MES.

At the end of the simulation run, all the data captured by the Data Collectors are saved and subsequently analyzed to generate Results.

6.4.2 Creation of new MEMs and Enhancement of the MES

The MEM describes the structural relationship of the model components. Since the MEM is an instantiation of the model structure, new MEMs can be created by generating new model specifications that use components defined in the MES.

The MES describes the behavior of each model component. Occasionally it becomes necessary to modify or enhance the behavior of some component. An incremental change is made by declaring a new object subclass from an existing object class in the MES. The addition of new CLOS slots and "before," "after," "around," or new CLOS methods modifies the behavior inherited from the superclass. The interpretive nature of CL and CLOS permits the changes to be made quickly in interpreted mode. The incremental change is added permanently to the MES when the new subclass is fully debugged and found to represent a reusable component. At that time the changes are compiled and made a permanent part of the MES. The MES therefore represents a library of reusable component classes.

New behavior and component classes are incorporated in models by changing the structure of the MEM and adding definitions to the MES.

7 DISCUSSIONS

7.1 Role of OOP

The OOP paradigm was a very strong contributor in all aspects of the project. The object-oriented approach not only helped in programming the implementation but also provided a means of thinking about the problem. The CLOS OOP paradigm eased the process of going from specification to implementation of the simulation model.

The OOP paradigm let us think in terms of physical entities of the system and interactions between them (e.g., orders, FGI, WIP, production planning, Factory, Distribution Center), rather than in terms of simulation language concepts such as events, buffers, resources and activities. When system complexity can be expressed clearly without clutter from the implementation concepts, both communication and thought become more effective.

The object-oriented implementation for discrete-event simulation simplified the software structure. A Message Event List to schedule messages instead of events made it possible to add new messages and objects without modifying the logic or structure of the Simulation Calendar or Message Event List. It permitted the OTS Model to be represented as a set of independent objects sending messages among themselves and to and from the DSC. The flow of control of the simulation was determined dynamically by the Simulation Clock and the Objects.

The encapsulation characteristics of OOP enabled us to isolate individual components and enhance their capabilities without changing the rest of the system. Enhanced capabilities permitted more realistic models to be incorporated into the simulation.

The class inheritance hierarchy helped to implement new capabilities as extensions of existing capabilities. The before and after methods in CLOS helped to quickly modify behavior of the OTS element.

7.2 Answers to Questions

The preliminary results generated by the OTS Simulation Model did not provide simple, clear answers to the questions in section 1.1. We need to emphasize that while these preliminary results make intuitive sense, they have been only partially verified and validated against the real system by discussion with domain experts. While some observations may appear obvious, they provide the basis for increasing confidence in the model operation.

Forecast error is defined by the ratio of (forecast minus actual) to actual. A negative forecast error describes a forecast below actual, and a positive forecast error describes one above actual. Order forecasts provide the information for ordering sufficient material to meet future orders of products. Material is ordered to satisfy the order forecasts and make the FGI and RPI reach certain preset Target FGI (TFGI) and Target RPI (TRPI) levels if those forecasts are realized exactly. With these facts in mind, let us now revisit the original questions:

- Question: Why do customers receive their complete orders in a matter of weeks when manufacturing cycle times have been reduced to half a day? Answer: The important factor is the combination of long lead times for parts from the vendors and negative order forecast errors acting together, and not the manufacturing cycle time. (Zero lead time for parts with negative order forecast errors or long lead times for parts with nonnegative order forecast errors will not cause the customers to experience long delays in delivery.) The reason is that part orders are based on forecasts of customer orders. If insufficient material is on hand when customer orders are received (i.e., forecast errors are negative), it is not possible to build and ship the product. Furthermore, if the order is to be filled completely (i.e., no partials) before shipment, the unavailability of a single line item on the order delays the whole order.

- Question: Why is product availability not immediate when there is plenty of finished goods inventory on hand? Answer: Products with large negative forecast errors have long availabilities and zero FGI. Products with small negative, zero or positive forecast errors have immediate availability and nonzero FGI. However, when these measures are aggregated across different products and sites, the net result is long availability and high FGI.

- Question: What is the best delivery performance that can be expected given a manufacturer's current operating conditions? Answer: 100% in the best case, when forecast accuracy is positive, zero or slightly negative. The TFGI, TRPI, Production Planning and Material Ordering policies mitigate the effects of negative forecast errors on delivery performance but cause higher inventories than desired for positive forecast errors.

- Question: Where does attention need to be focused to reduce delivery time? Answer: One approach is to rank order the parts by length of vendor lead time and then concentrate on reducing the lead times for the parts with the longest lead times, to allow quicker response to forecast errors. Another is to drive forecast errors to a level where the system can compensate easily for those errors.

7.3 Applicability of OTS Model

An often-asked question is: "Has the OTS model been deployed within the company, and if so, how much impact did it have?" Since this was a research project, the prototype OTS model was built to show proof of concept, and the model owners and sponsors were the researchers in HP Labs. As described in section 1.2, the research goal was to show the feasibility of applying simulation modeling at the broader level of the enterprise to help facilitate greater understanding, rather than to solve a particular problem. The domain experts helped us to define the OTS Graphical Model. No attempts were made to identify a clear problem owner or to quantify the prospective benefits of solving a particular problem. Working with an in-house model owner gave us the rare opportunity to do research in a new technology development and application area without the time pressures that otherwise would exist.

Under the circumstances, we did not attempt to address the issues associated with model deployment, access to real time manufacturing data, usability and better techniques to analyze the results generated by the model. In the author's mind, these issues are greater obstacles to widespread deployment than the issues of precision, accuracy or level of detail of the model.

Further value will be obtained from this model when a problem owner is identified in the business domain. While the OTS model of the specific manufacturing site is not currently being used for further experimentation, its component elements incorporated within the MES, the knowledge acquisition experiences we gained and the data analysis techniques we learned to analyze the data from the simulation results are the building blocks for further research in enterprise modeling and simulation.

A second question is: "Is the OTS model usable or applicable to a different problem or a different organization?" The answer is a qualified "Yes." The OTS model is directly applicable if the characteristics of the problem or organization are similar to those of the OTS process. Clearly, the number of changes and modifications depends on the magnitude of the differences of the process from the OTS process.

The components of the OTS model, through their existence in the MES, were used to build MEMs to solve two other problems with smaller scopes in a different part of HP and produced results that were easier to interpret and of more immediate business interest. One analyzed the inventory at the end of a product life cycle for a single factory with no DCs and the other estimated the amount of inventory needed because of the length of the production and material planning cycle times for a single product. Each problem is an interesting case study outside the scope of this document. Both problems were suggested by different problem owners, each of whom had an interest in the outcome, knew where to focus for results of interest and led the implementation effort in the appropriate direction.

The author believes that further activity on the MES and enterprise modeling and simulation activities should be guided by the interests of a problem owner to provide tangible benefits for improving the decision making process. Activity with this guidance will ensure that capabilities developed contribute directly to solving problems of a business nature.

7.4 Future Technical Directions

The following is a partial list of possible enhancements to this work to be considered in light of the interests of problem owners. These are strictly the personal opinions of the author, and do not reflect the opinion, endorsement, plans or official position of the management of HP Labs or HP Company.

7.4.1 Expand Model Capabilities

The current OTS Model focuses on manufacturing. Incorporating the R&D and marketing functions into the MES in greater detail will help to show how these two major functions interact with manufacturing in an enterprise. Incorporating money and resource flows into the MES will help to address financial considerations.

When considering these additions, more levels of abstraction become necessary with additional detail. At least two kinds of abstractions are possible:

- At the conceptual level, adding levels to the hierarchical abstraction will enable improved communication and clarity of thought.
- At the implementation level, substitution of a simple abstraction as an approximation of a more complex one will reduce the computational load.

7.4.2 Integrate with Other Software Systems

By incorporating and/or communicating with other software packages and systems, we can dramatically expand the range of processes that can be modeled and the problems that can be studied. Rather than look for one package that does everything, this approach should utilize the strengths while avoiding the shortcomings of a large number of different software packages or programs designed for different purposes. This approach will enhance MES capabilities by taking advantage of previously developed models.

For example, all algorithms are currently coded directly in CL and CLOS. Incorporating Linear Programming and Discrete Time model behavior into the MES will provide expanded model capabilities at a much faster rate than coding in CL and CLOS.

Packages currently exist for doing data analysis and graphical outputs. Two such packages that have been used together with the MES are S-PLUS (StatSci 1991) and Lotus 1-2-3 (Lotus 1991). Connections to other data analysis and graphical output packages should be investigated.

7.5 Contributions

In the opinion of the author, the primary technical contribution of this work is to demonstrate the feasibility of building a prototype simulation model of a manufacturing organization from the enterprise view. Preliminary results generated by the OTS model are consistent with the intuition of the domain experts. While the focus has been on the flows of material, information and control, the greater promise of technical contribution that the work holds is the potential to understand the complex dynamic interactions between the parts of the manufacturing enterprise that initiate and are reponsible for these flows.

The immediate tangible technical contribution is the set of reusable model components that was created and captured in the MES, out of which different MEMs can be created and generated using the Model Generator. In particular, the generic OTS element is a suitable building block for capturing material, information and control flows.

The indirect contributions include addressing the broader issues of material, information and control interactions and gradual development of a set of model elements that contribute to modeling and simulating the enterprise.

8 CONCLUSION

The foregoing has been a discussion of the practical aspects of implementing the OTS Simulation Model using HPM for knowledge acquisition, bridging the gap between the OTS process and the OTS Graphical Model, and using object-oriented methods to convert the OTS Graphical Model into the OTS Simulation Model. While the OTS Simulation Model yielded a way to answer the types of questions that were posed in the introduction to this document, of greater value was the foundation it provided to answer more complicated questions and the promise of the potential of showing the impact of complex interactions among different parts of the manufacturing enterprise.

The object-oriented paradigm, in combination with modeling and discrete-event simulation concepts, helped us to express, implement and manage models of complex material, information and control flows in the domain of manufacturing. The actual implementation of the OTS Simulation Model was a step in the direction of building a more general simulation model of the enterprise.

9 ACKNOWLEDGEMENTS

The work reported in this paper was done by the team of Bob Ritter, Bob Joy and the author, all of whom are affiliated with Hewlett-Packard Laboratories. On behalf of the team, the author would like to thank the people in the operating divisions that so graciously provided us the information with which we built the model and simulation. The author would also like to thank his team members and the many reviewers for their helpful comments and assistance in the preparation of this document.

10 REFERENCES

Barros, F. J., and Mendes, M. T. 1993, Object-Oriented Flow-Shop Simulation, *Object-Oriented Simulation Conference (OOS '93)*, 53–60.

Bhuskute, H. C., et al. 1992, Design and Implementation of a Highly Reusable Modeling and Simulation Framework for Discrete Part Manufacturing Systems, *Proceedings of the Winter Simulation Conference 1992*, 680–688.

Bravoco, R. R., and Yadav, S. B. 1985a, A Methodology to Model the Functional Structure of an Organization, *Computers in Industry*, Vol. 6 No. 5, 345–361.

Bravoco, R. R., and Yadav, S. B. 1985b, Requirement Definition Architecture – An Overview, *Computers in Industry*, Vol. 6 No. 4, 237–251.

Bravoco, R. R., and Yadav, S. B. 1985c, A Methodology to Model the Information Structure of an Organization, *The Journal of Systems and Software*, Vol. 5 No. 1, 59–71.

Bravoco, R. R., and Yadav, S. B. 1985d, A Methodology to Model the Dynamic Structure of an Organization, *Information Systems*, Vol. 10 No. 3, 299–317.

Carnegie Group Inc., 1988, *Knowledge Craft*, Vol. 1 – 4 (Carnegie Group Inc., Pittsburgh, Pa.).

Fadali, M. S., and Tacker, E. C. 1990, Hierarchical Process Modeling of Submarine Command and Control Systems, *International Conference on Systems, Man and Cybernetics*, 58–60.

Feng, Y., Zao, S., and Bao, J. 1992, An Object-Oriented Knowledge-Based Manufacturing Simulation System, *International Conference on Object-Oriented Manufacturing Systems (ICOOMS)*, 291–296.

Fox, M. 1992, The TOVE Project – Towards a Common Sense Model of the Enterprise, *International Conference on Object-Oriented Manufacturing Systems (ICOOMS)*, 176–181.

Godwin, A. N., Gleeson, J. W., and Gwillian, D. 1989, An Assessment of the IDEF Notations as Descriptive Tools, *Information Systems*, Vol. 14 No. 1, 13–28.

Hansen, K. 1984, *Data Structured Program Design* (Prentice Hall, Engelwood Cliffs, NJ).

Hewlett-Packard 1988a, *Mechanical Engineering Series 10 DesignCenter*, System, User and Interfacing Manuals (Hewlett-Packard Company, Palo Alto, Ca.).

Hewlett-Packard 1988b, *Structured Methods – An Overview for Engineers and Managers* (Hewlett-Packard Corporate Engineering, Palo Alto, Ca.).

Hewlett-Packard 1990, *HP Common Lisp*, Vol. I and II and User Manual (Hewlett-Packard Company, Palo Alto, Ca.).

Hewlett-Packard 1992, *Annual Report* (Hewlett-Packard Company, Palo Alto, Ca.).

Hughes, D., and Maull, R. 1985, A Framework for Design of CIM system architecture, *Computers in Mechanical Engineering*, Vol. 4 No. 2, 34–37.

Jorysz, H. R., and Vernadat, F. B. 1990a, CIM–OSA Part 1: total enterprise modelling and function view, *International Journal of Computer Integrated Manufacturing*, Vol. 3 Nos. 3 and 4, 144–156.

Jorysz, H. R., and Vernadat, F. B. 1990b, CIM–OSA Part 2: information view, *International Journal of Computer Integrated Manufacturing*, Vol. 3 Nos. 3 and 4, 157–167.

Law, A. M., and Kelton, W. D. 1991, *Simulation Modeling and Analysis*, Second Edition (McGraw-Hill, Inc., New York, NY).

Law, A. M., and McComas, M. G. 1991, Secrets of Successful Simulation Studies, *Proceedings of the Winter Simulation Conference 1991*, 21–27.

Le Clair, S. R. 1982, *IDEF the Method, Architecture the Means to Improved Manufacturing Productivity*, Tech. Rep. MS82-902 (Society of Manufacturing Engineers, Dearborn, Mi.).

Lotus Development Corporation 1991, *Lotus 1-2-3 for UNIX Systems Version 1.1*, User's and Reference Guides (Lotus Development Corporation, Cambridge, Ma.).

Luna, J. J. 1992, Hierarchical, Modular Concepts Applied to an Object-Oriented Simulation Model Development Environment, *Proceedings of the Winter Simulation Conference 1992*, 694–699.

Marca, D. A., and McGowan, G. L. 1988, *SADT Structured Analysis and Design Technique* (McGraw-Hill, Inc., New York, NY).

Marran, L. 1993, *Personal Communication* (Hewlett-Packard Professional Services Division, Mountain View, Ca.).

Marran, L., Fadali, M. S., and Tacker, E. C. 1989, A New Modeling Methodology For Large Scale Systems, *International Conference on Systems, Man and Cybernetics*, Vol. 5 No. 2, 989–990.

McHaney, R. 1991, *Computer Simulation: A Practical Perspective* (Academic Press, Inc., San Deigo, Ca.).

Mujtaba, M. S. 1992a, *Formulation of the Order To Ship Process Simulation Model*, Tech. Rep. HPL-92-135 (HP Labs, Palo Alto, Ca.).

Mujtaba, M. S. 1992b, Systems with Complex Material and Information Flows, *International Conference on Object-Oriented Manufacturing Systems (ICOOMS)*, 188–193.

Narayanan, S., et al. 1992, Object-Oriented Simulation to Support Modeling and Control of Automated Manufacturing Systems, *Object-Oriented Simulation Conference (OOS '92)*, 59–63.

Norman, V. B., et al. 1992, Simulation Practices in Manufacturing, *Proceedings of the Winter Simulation Conference 1992*, 1004–1010.

Pardasani, A., and Chan, A. 1992, Enterprise Model: A Decision-Support Tool for Computer Integrated Manufacturing, *International Conference on Object-Oriented Manufacturing Systems (ICOOMS)*, 182–187.

Pidd, M. 1992, Object Orientation and Three Phase Simulation, *Proceedings of the Winter Simulation Conference 1992*, 689–693.

Pritsker, A. A. B. 1986, *Introduction to Simulation and SLAM II*, 3rd Edition (Systems Publishing Corp., West Lafayette, Ind.).

Shewchuk, J. P., and Chang, T. 1991, An Approach to Object-Oriented Discrete-Event Simulation of Manufacturing Systems, *Proceedings of the Winter Simulation Conference 1991*, 302–311.

Shunk, D, Sullivan, B., and Cahill, J. 1986, Making the most of IDEF modeling – The Triple-Diagonal Concept. *CIM Review*, Vol. 3 No. 1, 12–17.

StatSci, Inc., 1991, *S-PLUS: a new philosophy of data analysis*, User's Vol. 1 and 2, and Reference Manuals (Statistical Sciences, Inc., Seattle, Wa.).

Steele, G. L. 1990, *Common Lisp: The Language*, Second Edition (Digital Press, Inc., Bedford, Ma.).

Ward, P. T., and Mellor, S. J. 1986, *Structured Development for Real-Time Systems*, Vol. I–III (Yourdon Press, Prentice-Hall, Engelwood Cliffs, NJ).

Worhach, P. 1992, Object Oriented Simulation for Equipment Level Design and Analysis in Semiconductor Manufacturing, *International Conference on Object-Oriented Manufacturing Systems (ICOOMS)*, 281–285.

Zeigler, B. P. 1984. *Multifacetted Modelling and Discrete Event Simulation* (Academic Press Inc., London).