THE DESIGN OF A SUPPORT DEPARTMENT SIMULATOR FOR USE IN THE DESIGN OF MANUFACTURING SYSTEMS

ANDREW THOMAS JACKSON

Doctor of Philosophy

THE UNIVERISTY OF ASTON IN BIRMINGHAM

November 1996

The University of Aston in Birmingham

The Design of a Support Department Simulator for Use in the Design of Manufacturing Systems

Andrew Thomas Jackson **Doctor of Philosophy 1996**

Summary

Manufacturing firms are driven by competitive pressures to continually improve the effectiveness and efficiency of their organisations. For this reason, manufacturing engineers often implement changes to existing processes, or design new production facilities, with the expectation of making further gains in manufacturing system performance. This thesis relates to how the likely outcome of this type of decision should be predicted prior to its implementation.

The thesis argues that since manufacturing systems must also interact with many other parts of an organisation, the expected performance improvements can often be significantly hampered by constraints that arise elsewhere in the business. As a result, decision-makers should attempt to predict just how well a proposed design will perform when these other factors, or 'support departments', are taken into consideration. However, the thesis also demonstrates that, in practice, where quantitative analysis is used to evaluate design decisions, the analysis model invariably ignores the potential impact of support functions on a system's overall performance. A more comprehensive modelling approach is therefore required.

A study of how various business functions interact establishes that to properly represent the kind of delays that give rise to support department constraints, a model should actually portray the dynamic and stochastic behaviour of entities in both the manufacturing and non-manufacturing aspects of a business. This implies that computer simulation be used to model design decisions but current simulation software does not provide a sufficient range of functionality to enable the behaviour of all of these entities to be represented in this way. The main objective of the research has therefore been the development of a new simulator that will overcome limitations of existing software and so enable decision-makers to conduct a more holistic evaluation of design decisions.

It is argued that the application of object-oriented techniques offers a potentially better way of fulfilling both the functional and ease-of-use issues relating to development of the new simulator. An object-oriented analysis and design of the system, called WBS/Office, are therefore presented that extends to modelling a firm's administrative and other support activities in the context of the manufacturing system design process. A particularly novel feature of the design is the ability for decision-makers to model how a firm's specific information and document processing requirements might hamper shop-floor performance. The simulator is primarily intended for modelling make-to-order batch manufacturing systems and the thesis presents example models created using a working version of WBS/Office that demonstrate the feasibility of using the system to analyse manufacturing system designs in this way.

Keywords: Manufacturing System Design, Simulation, Administration, Object-Oriented, Office Modelling

Acknowledgements

I have a number of people to thank, without whom I would never have been able to complete my research.

Firstly, I am grateful to colleagues at Aston University, and to Nick Boughton, in particular, for carefully reading and commenting on earlier drafts of the thesis. I must also thank Peter Ball and Jeff Barton for their patient assistance in resolving software problems. Sincere thanks must also go to my supervisor, Doug Love, for offering me the chance to become involved in this research. It has been an enjoyable and very rewarding experience.

Acknowledgement is also due to Lucas Engineering and Systems and the Department of Mechanical and Electrical Engineering at Aston University for supporting the research.

Finally, I must thank the many friends and family who have provided invaluable encouragement and moral support during the research and final preparation of the thesis.

Table of Contents

List of Figures

List of Tables

1. Introduction

The work described in this thesis relates to the development of a computer simulation tool that will help manufacturing system designers to evaluate how administrative and other activities might constrain the performance of a production system. This chapter will first comment on some important aspects of the terminology used within the thesis and will then introduce the wider context within which the research has been carried out. In addition, the general format of the thesis and of subsequent chapters will be outlined.

1.1 The Thesis

The thesis examines the provision of a computer simulation tool to assist in the design and analysis of production systems. Manufacturing firms are driven by competitive pressures to continually improve the effectiveness and efficiency of their organisations (Hill, 1993; Bennett & Forrester, 1993). For this reason, manufacturing engineers often implement changes to existing processes, or design new production facilities, with the expectation of making further gains in manufacturing system performance. However, since these systems must also interact with many other parts of an organisation, the expected performance improvements can often be hampered by constraints that arise elsewhere in the business. As a result, engineers need to be able to predict just how well a proposed design will perform when these other factors are taken into consideration.

While simulation tools are already used by designers to evaluate alternative manufacturing system configurations, this type of analysis invariably ignores the potential impact of support functions on a system's overall performance. Furthermore, current simulation software does not provide a sufficient range of functionality to be able to assess these effects properly. A working object-oriented simulator has therefore been constructed that will overcome this limitation of existing software in order to improve the manufacturing system design process.

1.2 A Note on Terminology

The vocabulary used to describe different aspects of a manufacturing organisation is

likely to vary significantly, depending on a particular author's objective and perspective on this type of business. When discussing manufacturing issues, it is therefore important to understand the context within which an author refers to different aspects of an organisation.

Since this thesis is presented from a manufacturing system design perspective, the terms *manufacturing function* and *shop-floor* will be used inter-changeably, referring explicitly to those elements of a business most directly involved in the production of goods. These elements are likely to be physically located within a firm's manufacturing facility and will include, for example, parts, machines, people, material-handling devices and so on. In addition, the terms *support function*, *support department* and *non-manufacturing function* will also be used inter-changeably but will refer to all those other elements not physically located in the manufacturing facility and therefore not directly involved in the production of goods. For example, these elements might include functions such as stores, purchasing, finance, sales order processing and so on. More specific examples will be introduced later.

Importantly, however, the terminology is not intended to imply that the relative significance of manufacturing or 'direct' functions is perceived differently to that of 'indirect' functions. It is simply a convenient way of grouping together those elements of business that the author will deal with in a similar way.

1.3 Background To Thesis

The research described in the thesis has been conducted within the context of a broader research agenda proposed by Love et al. (1992). The proposal has formed the basis of work carried out by the Integrated Design and Manufacture Research Group within the Aston Business School. The group's objective is to develop a software tool that satisfies the requirements for a Whole Business Simulator (WBS). For a fuller account of WBS the reader is referred to Barton et al. (1992) and Love & Barton (1993); the following discussion refers only to those aspects of the system that relate directly to this thesis.

In their paper, Love et al. (1992) demonstrate that product design and manufacturing system design decisions will have a significant impact on the financial performance of

any business. For this reason, WBS is proposed as a novel modelling tool that will predict how the business is likely to perform financially if a particular engineering decision or design were to be implemented. The tool will allow decision-makers to test the likely consequences of alternative proposals without disturbing the real business. The most appropriate design or operational change can then be implemented in the real business with a better understanding of the likely financial performance that the company might achieve as a consequence.

WBS overcomes at least two deficiencies of other modelling tools. Firstly, the scope of models created using other tools tends to be limited to manufacturing functions, with indirect functions usually being ignored. Secondly, where other tools do extend to modelling non-manufacturing functions, many aspects of the business tend to be represented in a very abstract or simplistic way. The consequences of evaluating engineering decisions using these restricted modelling tools will be discussed in more detail in a later chapter. WBS will, however, include both the manufacturing and non-manufacturing elements of a business within the same simulation model:

> *"The scope of WBS would extend to a least design, process planning, production control, manufacturing operations, sales, purchasing and accounts."*
>
> Love et al. (1992)

Some aspects of WBS have already been developed. Ball (1994), for example, has developed the Advanced Factory Simulator. This application enables the physical aspects of a manufacturing system to be modelled. Boughton (1995) has developed WBS/Control which enables alternative manufacturing planning and control systems to be modelled. WBS/Control can be combined with the Advanced Factory Simulator to produce a single model of a manufacturing system that includes its planning and control system. This allows a more comprehensive prediction of the effects of engineering decisions on manufacturing functions than would otherwise be possible if the physical manufacturing system alone were modelled.

Nevertheless, the Advanced Factory Simulator and WBS/Control are still both limited to modelling only manufacturing functions, support functions are ignored. The original WBS concept, proposed by Love et al. (1992), suggests that support functions

might simply be represented using real company software systems normally associated with that type of activity. For example, an accounts department might be represented in WBS by an actual financial accounting system, or a production engineering department represented by an actual computer-aided-process-planning (CAPP) system. However, demonstrating the feasibility of a necessary and more comprehensive approach to modelling non-manufacturing or support functions will form the subject of this thesis.

1.4 Structure of the Thesis

It will be demonstrated that such is the nature and frequency of interactions between a manufacturing system and its supporting functions, that simply representing each support activity using a corresponding software package is too simplistic. Predicting the wider business implications of manufacturing system design decisions requires a more comprehensive understanding of the delays and constraints that will be imposed on shop-floor facility by support functions. These functions will also be shown to account for a significant proportion of total business costs. The concept of using a simulation approach to model support functions will therefore be introduced; that is, modelling support departments in the same way as manufacturing functions are presently modelled. Establishing this novel but necessary application of simulation represents the first part of the thesis.

The problem then becomes the provision of appropriate simulation elements to model the support functions within a business and to integrate those elements with models of the manufacturing functions. It will be demonstrated that none of the many tools currently available provide the necessary range of functionality to cater for this application of simulation. Existing tools either lack the scope to model support departments at all, or do so in such a simplistic way that the interactions between different elements of the business cannot be modelled properly. The purpose of the second part of this thesis is therefore to present the design of a new simulation tool, WBS/Office. The implementation of this new design will then be shown to demonstrate the feasibility of simulating support departments in the context of the manufacturing system design process. A more detailed account of subsequent chapters follows.

Chapter 2 will begin the first part of the thesis by examining the role of support activities within a manufacturing business. The discussion will commence by developing the brief comments made earlier in this chapter to provide a more definitive statement of what is meant by the term 'support function'. The influence that these functions have on the performance of a manufacturing system, and ultimately on overall business performance, will then be demonstrated. This happens as a result of the many interactions that are required to operate and control a typical production system.

Chapter 3 considers the implications that these effects *should* have on the manufacturing system design process, by examining what factors related to the design process might cause support departments to become overloaded. Chapter 4 then investigates how effectively *actual* manufacturing system design methodologies are geared towards incorporating these factors into the decision-making process. It will be shown that, in practice, these formal methods make little or no attempt to assess how changes in support department workload might affect the outcome of design decisions. The lack of a suitable modelling tool for carrying out this kind of quantitative analysis will be shown to be a contributor to this shortcoming.

Chapters 5 and 6 therefore assess a variety of modelling techniques, with a view to establishing a more appropriate method for evaluating manufacturing system design decisions. Chapter 5 identifies specific modelling requirements, from both a functional and a practical point of view and then judges a range of techniques, such as spreadsheet analysis, queuing network theory and process modelling against this set of criteria. All of the techniques fail in some way to provide a comprehensive, and at the same time, easy to use method of analysing manufacturing system designs properly. Chapter 6 then establishes simulation as a more appropriate modelling technique but again, a review of commercial simulators concludes that current systems cannot be easily configured to model this class of problem. In other words, the need for a new type of simulator is identified.

Chapter 7 then looks at the most appropriate way of developing a new simulator, and concludes that the principles of object-orientation are best suited to creating this type of application. This chapter also briefly explains the different stages within a typical object-oriented development process, namely object-oriented analysis, design and

programming. These form the basis for a structured development of WBS/Office.

The object-oriented analysis of support functions will then be presented in two parts. Chapter 8 introduces the first part of the analysis which involves a general overview of different types of support activity as well as more specific analysis of administrative-type work. Chapter 9 completes the analysis phase by addressing issues such as the representation of people, the working environment, and other non-administrative aspects of support department activities.

The actual object-oriented design of WBS/Office then builds on the output of the analysis phase, and incorporates the object model that resulted from this phase into an existing library of simulation classes, namely the Advanced Factory Simulator and WBS/Control. Chapter 10 provides an overview of the design and how it fits into this existing library as well as discussing more detailed design issues.

Chapter 11 describes the factors involved in implementing this design to provide a usable simulator. This includes a discussion of the system's user-interface, which represents an extremely important part of any simulator. This chapter also presents some example models whereby WBS/Office is used to simulate the dynamic and variable behaviour of entities in both the shop-floor and support functions within a business, and to also mimic the kinds of interactions that would typically take place between these entities in a real organisation. In other words, the models demonstrate the feasibility of modelling manufacturing system design decisions this way and so WBS/Office thereby fulfils the objective of providing a more appropriate simulation tool for use in this particular design process.

Finally, chapter 12 will present the main conclusions from the research project and identify opportunities for further work. This will include possibilities for extending the simulator's modelling functionality even further, most specifically to meet the objectives of the Whole Business Simulator, as well as opportunities to apply WBS/Office to model other types of business problem. The need to incorporate WBS/Office into a new manufacturing system design methodology will also be discussed.

2. The Role of Interactions Between Support Functions and a Manufacturing System

This chapter examines the role of support functions in a manufacturing organisation. More specifically, the discussion will emphasise the influence that these functions have on the performance of a manufacturing system and ultimately on the performance of an entire business. To begin the discussion, a more definitive statement will be provided of what is implied by the term 'support function'.

2.1 What is Implied by the Term 'Support Function'?

The vocabulary used to describe and categorise different elements of a manufacturing business will vary, depending on an individual author's particular objective and perspective on an organisation. Since the term *support function* will be used throughout this thesis, it seems appropriate at this point to clarify which particular elements of a manufacturing business are implied when using this terminology.

Cost accountants have developed ways of categorising the different elements of manufacturing organisations; for example, the terms 'direct' and 'indirect' cost are often used (Drury, 1992). While cost-accounting is not the concern of this thesis, this cost-type classification serves as a useful starting point for differentiating between the manufacturing and support aspects of a business.

Within the context of cost-accounting, *direct costs* refer to expenditure on materials and labour that can be specifically attributed to a particular product. For example, *direct material cost* refers to expenditure on material that eventually becomes part of finished goods. Similarly, *direct labour cost* refers to the expense of employing people to physically transform direct material into finished goods.

All other expenses incurred by a business, in standard cost terms at least, are categorised as indirect or *overhead costs*. Within this category, cost accountants normally make two further distinctions; *manufacturing overhead* and *non-manufacturing overhead* (Drury, 1992). Precisely how this split is implemented will vary but the manufacturing overhead category typically applies to the cost of any activity that relates to the production of goods in general rather than to any specific

item. For example, material handling or maintenance costs will normally be treated as manufacturing overhead since these activities contribute to a firm's manufacturing operations but not to specific products. Non-manufacturing overhead, on the other hand, typically refers to the cost of activities that do not relate directly to production, such as administrative tasks or research and development. Non-manufacturing activity will normally be performed in office areas rather than on the shop-floor. Table 2-1 illustrates how a standard costing approach might classify different aspects of a manufacturing business in terms of those that contribute to manufacturing overhead and non-manufacturing overhead costs respectively.

Manufacturing Overhead	Non-Manufacturing Overhead
Goods Inwards & Stores	Financial Administration
Maintenance Engineering	Human Resource Management
Material Handling	Information Technology
Packing & Despatch	Manufacturing Engineering
Production Planning and Control	Marketing and Sales
Production Supervision	Product Design & Development
Quality	Purchasing and Logistics
	Training

Table 2-1 Contributors to Overhead Costs in a Manufacturing Business

It should be noted that Table 2-1 is not intended to represent actual costs, it only illustrates the activities or departments for which costs are incurred. For example, *Goods Inwards & Stores* identifies a function that will incur costs in terms of wages, consumable items, depreciation on equipment and so on. This illustration is also intended only as a rough guide since descriptions of functional areas will vary from business to business.

From a cost-accounting perspective, this classification is used as the basis for allocating different overhead expenses to the cost of specific products (Drury, 1992). However, the same classification also neatly groups together different aspects of an

organisation from a manufacturing engineering perspective. For example, it will be demonstrated later in the thesis that a manufacturing system design exercise will primarily address aspects such as the number of machines and machine operators; these factors all relate to the actual transformation of materials into finished goods. In contrast, the same exercise will generally consider all other aspects of the firm to be indirect or support activities since they do not contribute directly to the transformation of materials into products.

It follows that while the manufacturing system design process may address some aspects of support activity, such as material handling, quality and plant maintenance, attention is mainly focused on factors that contribute to the more 'direct' costs of production, namely materials, machines and labour. Consequently, the following definition of a support function is appropriate in the context of the manufacturing system design process:

> *The term 'support function' refers to any department or similar collection of resources that performs activities which enable a firm to continue manufacturing and selling products to customers but which does not involve the actual physical transformation of material or sub-components into finished goods.*

In other words activities that, from a standard costing point of view, contribute to manufacturing overhead and non-manufacturing overhead costs, can also be categorised as support functions from the view point of the manufacturing system design process. As a result, when the term *support function* (or *support department* or *non-manufacturing function*) is used within this thesis it refers to any function that corresponds to those described in Table 2-1. Again, however, it must be emphasised that the groupings presented in Table 2-1 are only representative of functions typically found in a manufacturing organisation and are intended to serve only as a point of reference for the discussions that follow.

2.2 Relationships Between Support Functions and a Manufacturing System

In terms of supporting the activities of a manufacturing system, a firm's indirect functions constantly interact with each other and the manufacturing system by passing

information and materials between each other. The relationships between a manufacturing system and its supporting functions are characterised by these interactions and will be now be discussed in more detail.

2.2.1 An Illustration of the Kinds of Interaction that Can Occur

There are notably few published studies of the kinds of interaction that typically occur between a manufacturing system and its support functions. In one such example, however, Tobias (1991) does attempt to illustrate some general flows of goods and information that might take place but does not suggest how these interactions have been identified or provide any references to endorse his illustration. Nevertheless, his work is still particularly relevant to this thesis since it is intended to represent the types of interaction that might take place from a manufacturing system design perspective.

Tobias's observations are also relevant since they closely match the conclusions of a small study conducted as part of the research described in this thesis (Boughton & Jackson, 1994). The aim of this study was to provide an indication of the range and number of different types of interaction that might typically occur in the course of operating a manufacturing business. Conclusions were drawn from a combination of experience and interviews with manufacturing system design consultants from Lucas Engineering & Systems and representatives of a UK-based manufacturing site. The results of the investigation are illustrated in Figure 2-1, with Table 2-2 providing a key to the various information and material flows depicted in the diagram.

The conventional boundary of a *manufacturing system* is clearly defined in Figure 2-1. The remaining functional areas, with the exception of *suppliers* and *customers*, are intended to represent different types of support function. While neither suppliers nor customers are considered to be physically part of a manufacturing organisation, the significance of their relationships with the manufacturing system and some of the support functions justified their inclusion in the illustration. It should also be noted that this particular representation of support functions, information flows and material flows is intended only as a general guide rather than as a definitive statement of the types of interaction that will always occur. Indeed, it was even evident during the course of this small study that both the types of support function and the kinds of interaction that one would expect to take place can vary.

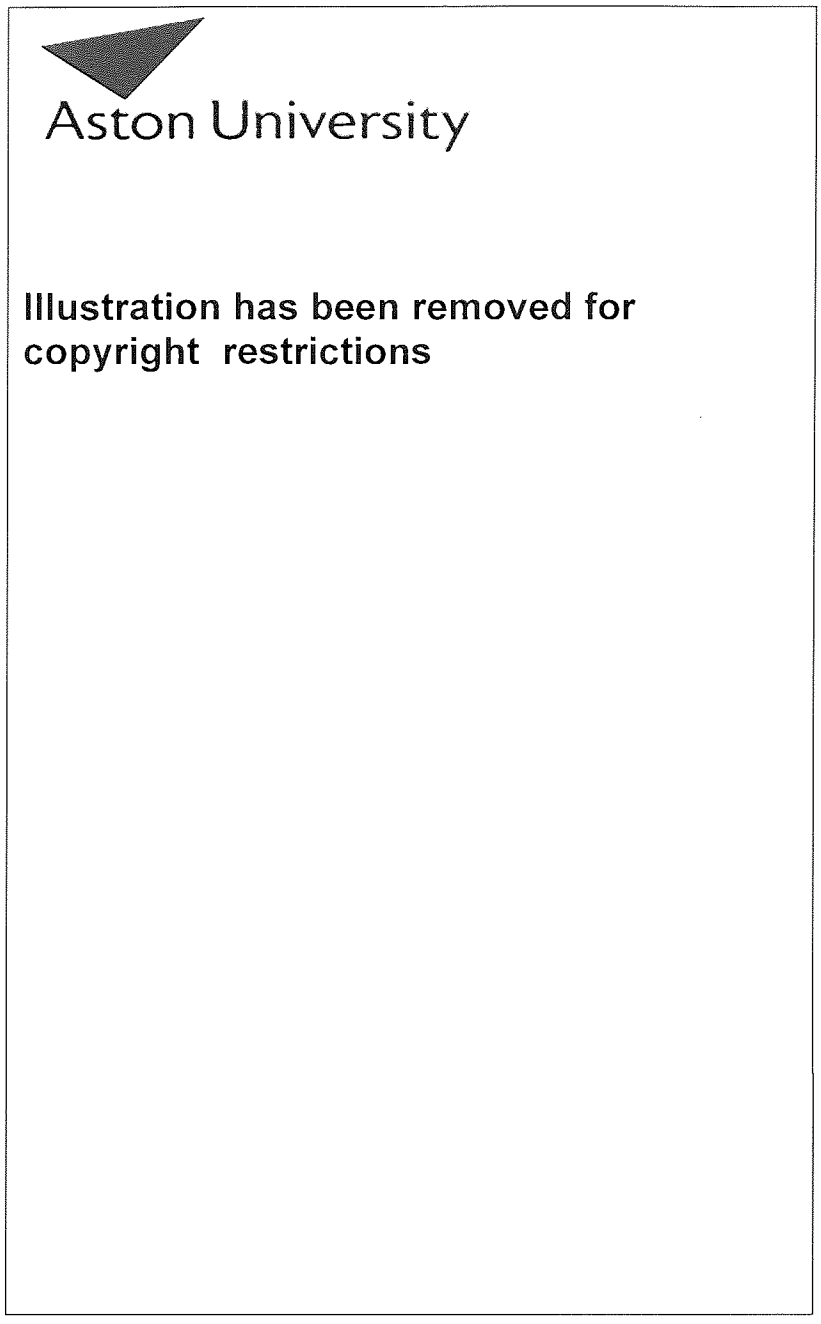
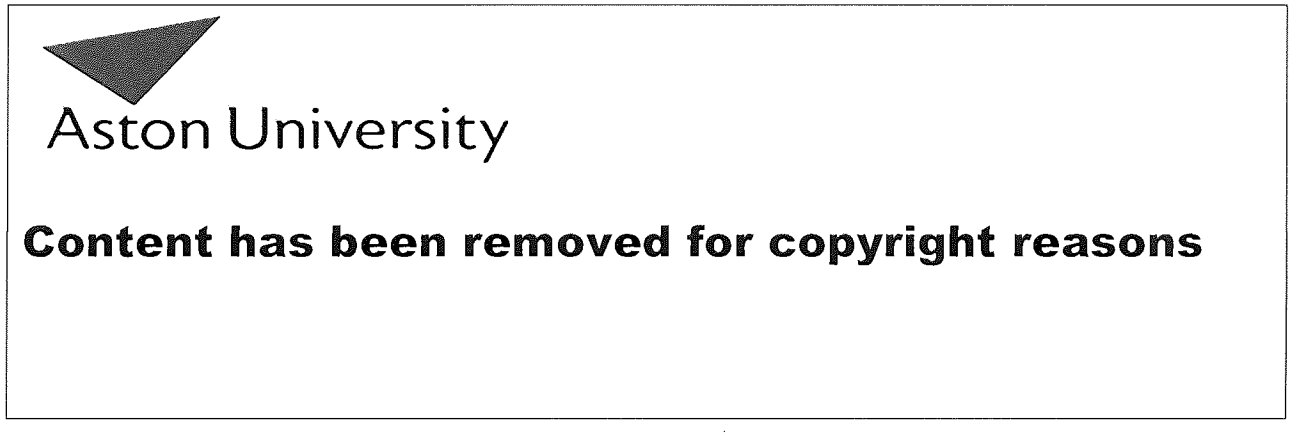


Figure 2-1 Typical Interactions Between a Manufacturing System and Support Functions (after Boughton & Jackson, 1994)

Table 2-2 Key for Figure 2-1



For clarity, only information flows or material flows *between* functions have been shown in Figure 2-1. Invariably material will also be moved *within* a manufacturing system and other kinds of interaction will also take place between individuals within the same support function. Nevertheless, the illustration emphasises the elaborate system of inter-dependencies and interactions that are typically involved in running a manufacturing organisation, and in particular, the high degree of contact between support functions and a manufacturing system.

Importantly, the notion of information or documents flowing through an organisation is not intended to imply only physical pieces of paper. While documents are presented in Figure 2-1 as physical 'carriers' of information, in practice, some interactions will involve electronic transfers of information rather than physical transfers of documents

by internal or external mailing systems. For example, in some cases information may be transferred internally via electronic mail or a work-flow system. In other cases information may be transferred to external suppliers or customers by fax or electronic-data-interchange (EDI). In addition, less formal exchanges of information, such as impromptu conversations between personnel from different functional areas, are excluded from the diagram. In practice, people either from different departments or from within the same department will confer with each other and thereby pass on information. Nevertheless, given the amount of information that will need to be processed and transferred between departments to operate a manufacturing business, it can be assumed that any crucial operational information relating to the manufacturing system will always be documented rather than being passed only by word-of-mouth.

2.2.2 The Implication of Interactions Between Different Functional Areas

Overall, the results of the study carried out by Boughton & Jackson (1994) emphasise the inherent complexity of a manufacturing organisation, given the variety of interactions that can take place. In practice, any interaction that occurs between two functions represents a potential delay, both in terms of the time taken to process the material or information and also to then transfer what has been processed to another destination. Figure 2-1 also shows that interactions invariably involve two-way flows of material or information. As a result, recipients will often introduce further delays while they prepare or process more information before returning documents back to their originators. This implies that substantial delays are likely as functions interact to exchange information or production materials, regardless of whether documents exist electronically or as physical pieces of paper.

Boughton & Jackson's study also emphasises that support functions often interact directly with each other as well as the manufacturing system. Nevertheless, many of these exchanges of information between support functions are still very relevant to the manufacturing system since they will invariably form part of a wider business process that will include the actual production of finished goods. For example, the diagram depicts a *stores and material handling* function sending requests for consumable items to the *procurement* function. The consumables are delivered from *suppliers* to the *stores and material handling* function via the *goods inwards* department. The

manufacturing system is obviously affected by this sequence of events since it will be the ultimate user of the procured items. As a result, the shop floor will still eventually be affected by any accumulation of delays that occurs even between support functions.

Altogether, Boughton & Jackson's study identified almost 60 different types of interaction. That implies around 60 different potential sources of delay that will ultimately affect a manufacturing system. The next two sections will present specific examples of how this system of interactions and delays affects the performance of a manufacturing system and potentially affects overall business performance.

2.3 The Impact of Support Functions on Manufacturing System Performance

The particular sequence in which interactions and delays will occur very much depends on the nature of the organisation. Andersen (1994), for example, presents a basic model of the different stages that contribute to the overall lead time for delivery of a product in a make-to-order business. An adapted version of this model is shown in Figure 2-2. This simple example represents a hypothetical case involving just a single customer order but will be used to demonstrate the principles of how support functions can constrain the performance of a manufacturing system.

Figure 2-2 Model of Lead Time in a Make-To-Order Business (after Anderson, 1994)

The diagram shows *customer lead time* to be made up of *order processing*, *supplier*, *manufacturing* and *delivery* lead times. The *manufacturing* component represents the manufacturing system actually transforming raw materials and components into

finished products. The other components of customer lead time involve tasks performed by support functions. These tasks will involve interactions such as those illustrated earlier. For example, *order processing* might involve interactions between the sales, production planning and procurement functions. The outcome of these information flows might be a production schedule or works order being sent to the shop-floor, and purchase orders being sent to suppliers for raw materials and tooling.

In this hypothetical case it will also be assumed that customer lead time is based on a confirmed date for delivery of the finished goods and cannot be changed, otherwise custom will be lost. Consequently, should any aspect of the business exceed its allotted lead time, the impact will be felt by downstream activities since they will be forced to reduce their own lead times to ensure that the customer's delivery deadline is still achieved. In this case, for instance, the manufacturing system is very much dependant on support functions such as production planning and procurement for the timely provision of works orders, raw materials and tooling to enable them to produce the finished goods within their allotted time. If either of these support functions fail to complete their respective tasks on time then the shop-floor will be forced to reduce its manufacturing lead time. In effect, the support department will prevent the manufacturing system from operating as planned.

In practice, manufacturing lead time might be reduced by temporarily running overtime on critical production processes. Alternatively, and as a longer term solution, a firm may decide to maintain inventories of common raw materials, sub-components and consumable items of tooling. However, if the firm operates as a job-shop there may only be a very limited number of common items that could be held as stock. As a result the firm may simply have to manage with longer term overtime arrangements or consider investing in additional plant or personnel for critical processes. This would increase the output of these processes and potentially improve the response time of the manufacturing system although it may simply contribute to a more complex and unpredictable system.

Any of these actions will at the very least increase the direct and indirect costs of operating the manufacturing system. There is also likely to be a degree of overlap between the start and end points of each stage in the overall make-to-order process

which is likely to make the activity even more difficult to control properly. Therefore, even in this simple case support functions can be shown to affect the performance of a manufacturing system in terms of its operating costs.

In practice businesses will be handling many orders simultaneously that relate to a range of products and different customer delivery deadlines. The manufacturing system is much more likely to be delayed where support functions have to simultaneously process many different customer orders. Since manufacturing process times will also vary from product to product, the potential pressures placed on a manufacturing system to maintain delivery dates are likely to be even greater than the hypothetical case suggests. For example, jobs assigned to key manufacturing resources may suddenly have to be rescheduled to allow other jobs to be completed. These other jobs may not have been started earlier because the order had not yet been released to the shop-floor or raw materials and tooling were not yet available. Attempting to reschedule any one particular job may set off a cascade effect whereby a series of other jobs also need to be rescheduled. Management of the manufacturing system will get progressively more complicated as it becomes harder to predict when particular jobs will start and finish.

As suggested in the hypothetical case, a firm faced with these circumstances may be forced to implement overtime or even sub-contract some jobs to ensure delivery deadlines are still honoured. Again, the costs of these actions are an unnecessary burden imposed on the shop-floor as a consequence of tardy support department performance. Spencer & Wathen (1994) describe the actual experiences of a make-to-order furniture manufacturer where behaviour similar to that just described did occur. In this case the order entry process became a constraint on the manufacturing system following modifications that were made to the shop-floor. The authors describe an 'end-of-month' syndrome which typically occurred as a consequence of orders being released late to the shop-floor by the production planning function. Overtime had to be introduced during the last week of each accounting period to ensure that shipping targets were met. Furthermore, shop-floor workers were often laid-off during the first week of the new accounting period because there was no inventory left to complete other shipments.

In another example (Collins, 1992), a make-to-order firm supplying industrial fastenings found itself in the position where some of its support functions were constraining the manufacturing system's ability to meet new customer delivery schedules. Unfortunately, the author does not discuss exactly how the shop-floor was affected but the impact on business performance must have been significant because the firm was forced to redesign the offending support departments.

This type of behaviour is not restricted to make-to-order firms; similar effects will also occur in other manufacturing environments. Traditional make-to-stock firms hold inventories of raw materials, work-in-progress and finished goods to ensure customers can be supplied immediately despite unforeseen delays in the manufacturing process. Further delays imposed on a manufacturing system by support functions may encourage firms to maintain larger inventories. For example, levels of raw material safety-stocks may be deliberately increased to cater for possible procurement delays. Similarly, firms may also decide to maintain buffer stocks of intermediate and finished goods to cushion other production delays imposed by support functions.

In addition, manufacturing performance is often measured in terms of resource utilisation. Consequently, where a manufacturing process is required to wait for a support function to complete a task, shop-floor managers may be tempted to keep their operators and machines busy by producing other items ahead of schedule. This will also cause levels of work-in-progress and finished goods to increase. While corrective action, such as increasing inventory levels, may apparently produce favourable performance figures in the short term, the longer term business implications are likely to be more detrimental.

2.4 The Business Significance of Support Functions

The examples described above illustrate firms that elected to change operating policies in their manufacturing systems to overcome constraints imposed by support functions and so maintain an acceptable level of performance. Changes included decisions to increase inventory levels even though the cost implications of holding unnecessary stock are well documented (see, for example, Bernard (1989), Toelle & Tersine (1989)). In addition, maintaining larger inventories and producing goods ahead of

schedule is only likely to introduce additional problems for the business. For example, the process of tracking and monitoring larger inventories will be more complex and may, ironically, place additional burdens on already overloaded support functions. Throughput times for all products are likely to increase and fewer goods are actually likely to be available on time for release to the customer (Goldratt, 1989). Given that responsiveness to customer demands will continue to become an increasingly important order-winning criteria (Chase & Garvin, 1989; Doll & Vonderembse, 1991), the failure of a manufacturing system to meet the delivery expectations of customers will invariably affect overall business performance.

An alternative to changing the operating policies for a manufacturing system would be to improve the performance of support functions, by increasing the number of personnel in crucial areas. In fact, if the expense of employing additional resources in support areas were small relative to the costs of operating a manufacturing system, then support functions could even be over-resourced to ensure that they are never likely to become overloaded. However, overhead costs are becoming an increasingly larger proportion of total costs incurred by manufacturing organisations, a fact that has been widely publicised.

Some of the most convincing evidence comes from surveys of UK manufacturing firms which, amongst other things, have sought to discover how the different components of business costs are changing. Murphy & Braund (1990), for example, analysed over 250 replies from questionnaires sent to CIMA (Chartered Institute of Management Accountants) members working in a "broad cross-section of industry where new technology is used". Another survey, published in 1988 by CAM-I (Computer Aided Manufacturing International), is referred to by Hill (1993) in a discussion of cost structures in manufacturing organisations. The CAM-I survey was conducted by interviewing representatives of around 40 UK companies. Table 2-3 shows the results from these two surveys, with the different elements of cost shown as percentages of the total.

The main observation from these results is that cost structures have changed over recent years to the point where overhead now accounts for approximately one third of total costs. This figure matches the findings of another, slightly more recent, survey

that is also referred to by Hill: a study of UK manufacturing firms carried out by Price Waterhouse during 1990/91 claims overheads represent between 25 and 50 per cent of total costs. Other authors such as Sutton (1991), Tatikonda & Tatikonda (1991) and Brimson (1993) also acknowledge that overhead costs are become increasingly more significant in relation to other costs, although the do not provide any empirical evidence of how cost structures have changed.

	CIMA Members (Murphy & Braund, 1990)		CAM-I (Hill, 1993)	
	1984	1989	1960	1986
Direct Materials	47%	50%	56%	53%
Direct Labour	18%	18%	22%	15%
Overhead	36%	34%	22%	32%

Table 2-3 The Elements of Cost in UK Manufacturing Firms

Overall, this evidence suggests that support functions now contribute to a significant proportion of total business costs. It follows that attempting to counteract poor support department response times by employing additional staff in these areas is likely to be expensive and so affect the financial performance of the business. A more effective solution would be to consider how a manufacturing system might be affected by these kinds of activities during the actual design of the system and then, if necessary, tailor the system to operate within any support department constraints. It follows that a pre-requisite to implementing this type of approach would be the availability of appropriate tools and techniques for analysing manufacturing system designs in this way.

2.5 Summary

This chapter has illustrated how support functions can limit the potential performance of a manufacturing system. This happens as a result of the many interactions with support functions that are required to operate and control a manufacturing system; every interaction invariably represents a potential delay which ultimately affects a manufacturing system's ability to respond to customer demands. In fact, examples of companies have been described where the performance of support functions has

eventually placed constraints on shop-floor performance.

While changing policies or manning levels in either the manufacturing system or support areas might overcome these constraints in the short-term, the longer term effects on business performance are likely to be more detrimental. A more effective solution would be to consider how a manufacturing system might be affected by these kinds of activities during the design process. The next chapter therefore examines what factors related to the design process are likely to cause support departments to become constraints on manufacturing system performance.

3. Support Department Performance and the Manufacturing System Design Process

The previous chapter demonstrated that delays incurred by support functions can have a significant impact on the performance of a manufacturing system and ultimately on the performance of an entire business. This chapter will identify the implications that these effects have in terms of the manufacturing system design process by examining what design factors cause support departments to become constraints.

3.1 Understanding Why Support Functions Become Overloaded

Since any manufacturing system will only have a limited number of operators and machines available at any one time, managing the loading of these resources is a fundamental requirement for managing a manufacturing business. Support functions are no different to a manufacturing system in that they too will only have a limited number of resources available. Consequently, there will always be a limit to the volume of work that a support function will be able to process in any given period.

In practice, however, the capacity of individual support functions tends not to be compared with the volumes of work that they will be expected to process to anywhere near the same extent that loading of a manufacturing system is controlled. For this reason, not only would it be relatively easy for a support department to be overloaded but the impact that this would have on the performance of other aspects of the business is more likely to go unnoticed. This is evident from views expressed by several different authors who emphasise the lack of attention that firms pay to monitoring the workload placed on support functions.

Altogether the different views expressed span a significant period of time but are all still very relevant to this discussion since they are very similar in their assessment of how support department loading changes. Dixon (1953), for example, described how new tasks can all too often be allocated to support functions with little or no analysis of how the loading of resources in these areas will be affected. He described the effects of this behaviour as 'creep' and suggests that:

> *"[Support] activities tend in some cases to edge into the operations of a business firm where no special thought has been given to their*

short-run or long-run effects upon costs and efficiency"

The author also argued that creep will eventually affect not just support functions, but the whole business:

> *"[Creep] is not a short run problem. It consists of the more or less gradual, unrecognised, cluttering up of business activity, accompanied by a parallel deterioration of company efficiency, a building up of fixed costs, and the undermining of profit potential"*

Dixon blamed the costing methods used by firms at that time to predict the financial implications of business decisions as the main reason for the occurrence of creep. He argued that the treatment of many indirect expenses as fixed costs led to the assumption that changes within the business will have no effect on the performance of support functions.

The notion of 'creep' is still relevant today since similar concerns have been put forward more recently by proponents of activity-based costing. For example, Cooper & Kaplan (1988a) also challenge the traditional 'fixed' versus 'variable' cost classification and argue that it often obscures the true complexity of support department behaviour. They suggest that support functions are usually constrained to increasing their resources only when budgeted to do so, rather than when demand for the service which they provide changes. As a result, a support department will often have to operate for a period of time without the necessary resources to process its current workload. Consequently, the performance of that department will suffer and inevitably other aspects of the business that depend on the service provided by that support function will also be affected.

Miller & Vollman (1985) also emphasise the lack of attention paid to support department loading. They have coined the term 'hidden factory' to describes the mass of off-line transactions that will take place in a manufacturing business; in other words, the document and information flows described in the previous chapter. This term seems very appropriate since it suggests a similarity between the concept of work in manufacturing and support functions, yet emphasises the different attitudes to managing these aspects of the business. In other words, it suggests that while

processing documents is analogous to processing raw materials, considerably less attention is paid to monitoring the loading of support departments than is paid to the loading of the factory-floor. The significance of changes in support activities also tends to go unnoticed.

Miller & Vollman argue that some firms do manage support department loading by simply over-resourcing these areas to ensure that over-loading never happens. In other words, some businesses purposely deploy more resources in support functions than is actually necessary to ensure that they will be able to cope with additional workloads in the future. It is suggested that this approach originates from a time when the cost of support department resources was relatively low in comparison to expenditure on shop-floor resources.

However, it was demonstrated in the previous chapter that the cost structure of manufacturing organisations has now changed, in that support functions now account for a much higher proportion of the total costs incurred by a manufacturing business than ever before. Consequently, over-resourcing support departments is now a very costly way of ensuring that these functions can cope with additional workloads in the future. In addition, the concept of 'creep' put forward by Dixon emphasises that changing circumstances elsewhere in a firm invariably change the loading on support departments. This implies that while a support function may be over-resourced given its current workload, if that workload is not monitored properly then it may still eventually reach a level sometime in the future where the function could be overloaded. In other words, there is no guarantee that simply by over-resourcing support functions they will not become a constraint on other aspects of the business at some point in the future.

In summary, changing the way a manufacturing business is organised or operated can have potential knock-on effects in terms of altering the demands placed on support functions. Even if the amount of indirect work is not altered, the current workload on indirect areas may be enough to prevent the intended impact of changes from being fully realised. Furthermore, since the capacity of individual support functions tends not to be compared with the volumes of work that they will be expected to process, the onus must be on decision-makers to anticipate how support departments will be

affected by their changes. Clearly this conclusion has implications for the manufacturing system design process. The next section will attempt to demonstrate how decisions relating to the design of manufacturing systems can be responsible for this overloading.

3.2 The Impact of Design Decisions on the Performance of Support Functions

A firm may decide to design or re-design a manufacturing system for a number of reasons; new product introductions, variations in the product mix, technological developments in manufacturing processes, or as an ongoing programme of improvements aimed at enhancing the firm's competitive position (Dales & Johnson [*sic.*], 1986). Nevertheless, on whatever basis a manufacturing system design or re-design exercise is instigated, decisions taken as part of this process can potentially affect support functions in number of ways.

This is evident from the results of the study cited earlier (Boughton & Jackson, 1994) that provided an indication of the range and number of different types of interaction that typically occur in a manufacturing business. In particular, the study emphasised that these interactions invariably involve two-way flows of documents or material. This means that the activities of a manufacturing system must be responsible for initiating flows in at least one direction and, therefore, will be partly responsible for the overall volume of interactions that occur. Consequently, decisions taken during the manufacturing system design system process must be partly responsible for the workload placed on support functions. What follows is a more detailed discussion of the design factors that might typically affect support department loading.

3.2.1 The Significance of Process Choice

A fundamental aspect any manufacturing system design is the notion of 'process choice' (Hill, 1993). This concept is also sometimes referred to as 'process technology' (Hayes & Wheelwright, 1984) and relates to a collection of five generally accepted styles of manufacturing system. These are briefly described below:

- *Project-based* implies production of large custom or tailor-made products in very low volumes, often just single items. Manufacturing tasks vary from project to project, therefore standard procedures or

routings seldom apply. Examples include shipbuilding and civil engineering projects.

- *Job-shop* usually refers to firms that manufacture low volumes of a very high variety of products, most of which require a different sequence of processing steps. Jobs-shops produce more and usually smaller goods than project-based firms but the degree of repetition will still be relatively low. Examples include specialist toolmakers or printing firms.

- *Batch-manufacturing* firms typically produce a wide range of a relatively similar line of products. Most jobs will be repeated periodically. Examples include the manufacture of automotive components and metal castings.

- *Flow-line or mass-production* implies high volume production of a relatively small range of products. Typically workstations are arranged in sequence and the product passes through a series of steps at a controlled rate. Examples include motor vehicle manufacture and the production of household electrical goods.

- *Continuous-processing* is similar to mass-production but goods are not made from discrete parts. An even lower variety and higher volume of goods will be manufactured. Examples include chemical manufacturing.

From these definitions it is clear that process choice closely relates to both the range and volumes of products that a firms intends to manufacture. This will determine many of the physical aspects of a manufacturing system, such as the level of automation and the placing of machines in functional, cellular or dedicated flow-line arrangements. In addition, however, evidence suggests that the choice of process technology also has significant implications in term of the demands placed on support functions.

Hill (1993), for example, uses the term 'infrastructure' to describe the collection of systems and procedures needed to support the physical activities of a manufacturing system and argues that process choice will affect fundamental aspects of this

infrastructure. He suggests, for instance, that in order to provide adequate support to manufacturing, the complexity of a manufacturing facility will need to be reflected in the system of indirect functions.

Hayes & Wheelwright (1984) adopt a similar view but in addition to process choice they also identify that the size of the organisation will affect the type of supporting infrastructure required. Since both Hill (1993) and Hayes & Wheelwright (1984) present their case from manufacturing strategy perspective, they provide little in the way of specific examples of how process choice will affect support functions. Nevertheless, the arguments proposed in each case imply that the process of designing a manufacturing system design should also take into account many non-manufacturing aspects of the business.

Ward et al. (1992) go somewhat further and attempt to identify empirical relationships between process technology and aspects of a firm's support functions, or 'infrastructure'. They analyse US Department of Labor survey data on employment by occupation, from a sample of more than 110 thousand US manufacturing establishments. From this data they establish mean numbers of non-production workers per 100 production workers for different choices of process technology. These relationships are depicted inTable 3-1.

The results suggest that the levels of some types of support staff, such *as material movers* and *plant maintenance workers*, appear to be generally more significant in mass production and continuous processing firms than in other types of business. However, this may simply reflect the higher levels of automation and therefore fewer numbers of direct operators in these industries rather than any actual variation in the levels of this type of support. There does not appear to be any similar trend in the relative levels of staff employed in other support areas such as clerks, administrative specialists and engineers. Nevertheless, the study emphasises that staffing levels in some areas will vary considerably depending on the choice of process technology, presumably because of the different demands that process choices place on of support functions. In fact, the lack of any obvious trend in many cases actually suggests that predicting the manning levels in other areas is likely to be more complicated.

Occupation	Description	Job-Shop	Batch Prodn.	Mass Prodn.	Continuous Processing
Clerical Staff	Clerical and administrative support workers, including clerical supervisors.	45	34	50	51
General and Functional Managers	Production, financial, personnel, purchasing, marketing, administrative, scientific and other managers.	18	16	15	21
Material Movers	Material moving equipment operators, hand material movers, machine feeders and other labourers.	16	30	42	72
First Line Supervisors	First line supervisors in production, construction and maintenance.	14	12	13	26
Engineers	Engineers and engineering technicians.	11	15	8	13
Plant Maintenance Workers	Plant mechanics, installers and repairers.	10	15	19	56
Administrative Specialists	Accountants, buyers, personnel specialists, computer specialists and other professionals not classified elsewhere.	8	11	10	15
Inspectors	Inspectors, testers and related workers.	6	9	13	7
Service Workers	Catering, cleaning and other service workers and supervisors.	3	5	4	9
Scientists	Natural and mathematical scientists and technicians.	0	4	1	10
Sales Workers	Sales and related workers.	No statistically significantly relationship found.			
Material Control Workers	Material control, dispatching, scheduling and distribution workers.	No statistically significantly relationship found.			

Table 3-1 Employment of Indirect Workers per 100 Production Workers by Process Technology (adapted from Ward et al., 1992)

Interestingly, the data did not produce statistically significant relationships between manufacturing planning and control workers and process technology. This is surprising given that this function is arguably more closely related to a manufacturing system than other aspects of the organisation, and one might therefore expect a close correlation between process choice and control system. Nevertheless, this conclusion does not imply that the numbers of production control staff are insignificant, it simply establishes that the volume of work in this area of business is typically dependant on factors other than process technology. Indeed, the effects of different manufacturing planning and control systems on support departments will be discussed in more detail later in the thesis.

It should also be noted that the analysis conducted by Ward et al. (1992) does not suggest how effectively or efficiently support functions will be able to serve a manufacturing system by maintaining these staffing levels. It can only be assumed that these numbers do provide a general indicator of the staffing need to ensure an acceptable level of service. Consequently, these figures also provide further evidence of the significance of support functions. For example, the study shows that around 50 clerical staff are required per 100 production workers.

Overall, this empirical evidence demonstrates that decisions relating to process technology can affect the level of support required by a manufacturing system. Consequently, it can be implied from this aspect of the manufacturing system design process alone that decision-makers should attempt to anticipate any changes in the number and frequency of interactions that a process technology decision is likely to demand of support functions. In addition, decision-makers should attempt to anticipate how effectively the resources that will be available in these areas will be able to respond to this new level of demand.

3.2.2 Other Decisions Relating to the Manufacturing System Design Process

While different styles of manufacturing system clearly place different demands on support functions, other aspects of the design process can also affect the loading on indirect areas. For example, a manufacturing system redesign exercise might involve a make-or-buy analysis. The outcome of this type of decision will determine the number

of different items to be purchased. In turn, this will affect the volume of interactions between a purchasing department, suppliers and a goods-inwards function, and will therefore affect the loading on these aspects of the business. Other decisions such as those relating to operating policies like batch sizes will also affect the number of interactions between a manufacturing system and production control functions.

However, since the loading on support functions is generally not monitored to the same degree that shop-floor capacity is controlled, it is quite feasible that a manufacturing system design decision will increase a support department's workload to the point where it cannot process all the work on time. Furthermore, the impact of this overload is unlikely to be realised until the support function becomes an obvious constraint on the performance of the manufacturing system.

Santosus (1993) describes the actual experiences of a window manufacturer where behaviour similar to that just described did occur. The organisation had mass-produced standard products for many years and the company's support departments were able to cope with the levels of off-line administration that this entailed. Problems arose when the firm decided to redesign its manufacturing system to also produce goods for make-to-order and custom markets. Various support functions simply could not cope with the additional workload that this change implied. For example, the author describes how difficulties in order scheduling, inventory management, product-labelling and shipment verification eventually reached such a level that relationships with customers deteriorated. The firm eventually managed to resolve the situation by redesigning the offending support departments. Customer relationships may not have suffered at all if the firm had been able to predict the effects that changes to their manufacturing system would also have had on their support departments.

In other cases, manufacturing engineers have found that they have improved the productivity of a manufacturing system to the point where the constraining factor moves off the production line into clerical support areas. Underwood (1994), for instance, presents the case of a computer manufacturer that implemented computer-integrated-manufacturing and Kanban control on the shop-floor. The performance of the manufacturing system and the business did improve as a result of these actions but the potential performance gains were not fully achieved because the more responsive

manufacturing system was now constrained by the off-line order processing function.

In another example, Billesbach & Schniederjans (1989) describe the application of just-in-time techniques to administrative functions in manufacturing organisations. The authors justify the approach on the basis that inefficiencies in support functions can offset productivity gains in manufacturing, and suggest that effort should also be applied to improving the performance of these functional areas. They present two case examples of manufacturing firms where their approach did help to overcome support department constraints.

3.3 Summary

This chapter has established that implementing changes in a manufacturing business can have potential knock-on effects in terms of altering the demands placed on support functions, particularly since the capacity of support functions tends not to be compared with the volumes of work that they will be expected to process. It has also been established that the extent to which support department delays affect shop-floor performance can be influenced significantly by decisions taken during the manufacturing system design process. Given this evidence, one might expect that decision-makers would endeavour to understand how their manufacturing system designs will affect support areas. The next chapter will therefore investigate just how effectively manufacturing system design methodologies are geared toward incorporating these factors into the decision-making process.

4. The Limitations of Current Approaches to Manufacturing System Design

Preceding chapters have identified the significance of support functions in terms of their ability to restrict the performance of a manufacturing system. Given the significance of these problems one would expect manufacturing system designers to consider how a new or re-designed manufacturing facility is likely to be affected by support areas. This chapter will examine the extent to which decisions-makers do try to anticipate the potential impact of support functions during the process of designing a manufacturing system.

The discussion will review both the manufacturing system design process in general as well as other more specific, but closely related, decision processes; namely the selection or design of production planning and control systems, and make-versus-buy analysis. Decisions taken in these situations are notoriously difficult, not only because of the many factors involved, but also because the implications of making the wrong decision are potentially so severe. Consequently, a range of methodologies and techniques have been developed specifically to guide engineers toward making the right decisions in these different areas. These serve as useful points of reference from which to evaluate the quality of decision-making in these areas.

4.1 The Concept of a Manufacturing System

The approach to changing the way production is carried varies enormously between companies. Consequently, there are many different approaches to the design or redesign of a manufacturing system (Fritz et al., 1993). This happens both as a result of the different markets within which firms must operate and because of the different perceptions of what a manufacturing system actually involves. Parnaby (1979) suggests that there is no single concept of a manufacturing system, but does identify certain factors that should be taken into account, regardless of the type of industry that a company is operating within. These factors are shown below:

- A manufacturing system should be an integrated whole, composed of different subsystems, each of which interacts with the whole system.
- The manufacturing system should be created by selecting appropriate subsystems to physically or chemically process raw materials, linking

these together and controlling the resulting system and its interaction with its environment.

- To be able to operate successfully, a manufacturing system also requires an administration and information flow system.

All of these characteristics emphasise that a manufacturing system involves considerably more than a collection of machines, transport devices and operators on a factory floor. A manufacturing system must also include a supporting infrastructure to manage all planning and control aspects of the system on a day-to-day basis. In other words, Parnaby acknowledges that the design of a manufacturing system must also take into account many non-manufacturing elements of the business.

On this basis, Parnaby suggests that the design of a manufacturing system should adopt a *systems engineering* approach (Parnaby, 1986), whereby technical, organisational and human factors are all considered. Specific aspects of the manufacturing system design approach proposed by Parnaby will be introduced in more detail in the next section. As a more general point, however, he asserts that the process of designing a manufacturing system must form part of a wider business strategy that will also incorporate marketing, product engineering and the design of information systems.

Other commentators on manufacturing strategy adopt a similar view of manufacturing system design. For example, the notion of an *infrastructure* (Hill, 1993, Hayes & Wheelwright, 1984) that supports manufacturing activities was introduced in the previous chapter. Given that, from a strategic point of view, a manufacturing system should be an integrated system of material flow and information processing functions, one would expect that formal methods relating to the design of manufacturing systems would incorporate both aspects. The next section will examine the extent to which formal manufacturing system design methodologies do consider these aspects and, in particular, the role of support functions in the overall design solution.

4.2 The Application of Formal Methods to the Design of Manufacturing Systems

Fritz et al. (1993) present the results of a survey which attempted to establish why so many manufacturing organisations encountered difficulties when designing production

systems. In particular, the survey looked at the extent to which UK firms used structured approaches to manufacturing system design. The response suggests that only 30 per cent of firms do use a structured approach. Furthermore, of this 30 per cent, the majority (around 66 per cent) claimed their methodology had been developed in-house. The remainder of firms adopting a formal approach used methods provided by consultants or techniques that were described in published literature.

These figures are similar to results obtained from another survey of UK firms that was conducted by Devereux et al. (1994). In this case the response suggested that some 50 per cent of firms used a formal approach to the design of manufacturing systems. Consistent with the other survey, the majority of these firms (around 60 per cent) adopted an in-house methodology. Again, the remainder of firms adopting a structured approach used methods that originated from consultants or the public domain.

Given the relatively low uptake of published methodologies, any assessment as to how firms approach manufacturing system design that is based on an evaluation of these approaches is unlikely to be representative of current practice. However, it can only be assumed that published methodologies represent current 'best' practice, and are likely to have been tested in a much broader range of organisations than techniques developed in-house. For this reason, two structured approaches to planning production systems will be examined; the Lucas Engineering & Systems Manufacturing System Design Methodology and the Design Rules for Analysing Manufacturing Activities (DRAMA II). Both of these methodologies are based on experiences and analyses of UK manufacturing firms and literature relating to each technique is readily available in the public domain.

4.2.1 The Lucas Engineering & Systems Manufacturing System Design Methodology

Lucas Engineering & Systems have developed an approach that has been extensively applied to the design and redesign of manufacturing systems. The technique is generally referred to as the 'five-step' design methodology (Parnaby, 1986; Lucas Engineering & Systems, 1989). The five different stages are illustrated in Figure 4-1. The illustration shows how the Lucas methodology adopts a 'systems engineering' approach by incorporating the different subsystems of a business into the design of a

manufacturing system. The methodology also attempts to satisfy the strategic requirements of a manufacturing system described earlier; process choice and infrastructure.

Figure 4-1 The Lucas Five-Step Manufacturing System Design Methodology (after Parnaby, 1986 and Lucas Engineering & Systems, 1989)

Stage 1, for example, identifies the choice of process technology by evaluating the volume and variety of parts to be manufactured. Stages 2 and 3 develop the concept of process choice further and produce a detailed design of the shop-floor aspects of the system. The likely performance of the shop-floor facility will also be tested at this stage. For example, during the dynamic-design phase simulation techniques may be used to evaluate the shop-floor aspects of the design. In other words, a simulation model of the proposed shop-floor configuration will be used to assess how the real system would behave under conditions that differed from the average conditions assumed during the steady-state design phase. Such factors might include the

breakdown of machines, fluctuations in the demand for different products and variations in the performance of operators. Simulation would also provide a better insight into how the elements of the real shop-floor system would interact.

Stages 4 and 5 of the design process relate to the infrastructure needed to support the shop-floor system and which therefore represent a very important aspect of the broader manufacturing system. These stages may include a redesign of existing paper-flow or information-flow systems to reflect proposed changes to the shop-floor. In addition, decisions may be taken to replace manual paper-flow systems with automated information-flow systems in an attempt to make these aspects of the business more efficient.

The scope of the Lucas methodology is therefore such that it does at least attempt to cover all aspects of a manufacturing system. Indeed, the methodology has resulted in some successful system designs or redesigns (Wood, 1986; Kellock, 1992; Turner, 1994). However, the approach does place significantly more emphasis on the shop-floor functions and, in particular, on the physical configuration of people, machines and other equipment directly involved in manufacture. The emphasis is therefore on material rather than information flows.

In addition, the design of the physical aspects of the manufacturing system will have effectively been finalised by the time the processes needed to support the shop-floor facility are considered. For example, the dynamic-design phase normally includes a rigorous evaluation of how the different elements of the shop-floor interact. Indeed, this stage of the process is intended as a final test of that aspect of the system, in terms of its robustness to cope with variations in the behaviour of different shop-floor elements. However, the methodology makes no provision for assessing the interaction between manufacturing functions and support functions in the latter part of the design process. As a result, designers will not take the opportunity to test the robustness of the existing or proposed support functions to cope with demands placed on them by the shop-floor or other aspects of the business.

Consequently, the manufacturing system that results from this design process may not perform effectively as an 'integrated whole', even though the earlier discussion on

manufacturing strategy emphasised this to be essential for overall business success. For example, in a series of studies Lewis (1994) describes cases where the Lucas methodology has resulted in the implementation of manufacturing systems that have failed to achieve the expected improvements in manufacturing or business performance. These cases include an example of where failure to properly consider support functions was one of the factors that caused a manufacturing system implementation to fail; Lewis describes the experiences of one particular company where the effect of a new manufacturing system implementation was to increase the workload on the goods inwards function. This resulted in parts being delivered late to the shop-floor, hindering its ability to adhere to production schedules. Problems of this kind could have been avoided if the methodology had made provision for testing how support functions would be affected before implementing the new facility.

As a more general point, the time taken to design and test the shop-floor aspects of a manufacturing system will affect the time that can be afforded to the design or evaluation of support functions. For this reason, any technique for testing the interaction between manufacturing and support functions should be relatively quick and easy to use. This is an important aspect of the solution that will be presented later in the thesis for overcoming the limitations of the manufacturing system design process that have just been described.

4.2.2 DRAMA II : Decisions Rules for Analysing Manufacturing Activities

DRAMA II is an alternative methodology proposed by Bennett & Forrester (1993) that relates to both the design and implementation of manufacturing systems. It is intended to be a generic approach, applicable to the design of all types of manufacturing system, although it originates from an earlier technique specifically developed to support the design of electronics assembly systems. The design methodology comprises ten components that are intended to be carried out sequentially, as follows:

- *Market and Environment* involves analysis of a firm's external environment to determine corporate objectives and marketing strategy.

- *Manufacturing Strategy* will attempt to set more local manufacturing targets in line with the overall business strategy.

- *Organisation* includes the definition of the organisational structure and lines of communication.

- *Justification* will attempt to prepare the case for a new or modified manufacturing system, based on the financial or other more qualitative gains that would be achieved as a result.

- *Project Management* addresses the planning and control of the manufacturing system design and implementation. It takes into account time and monetary constraints.

- *Physical System Design* relates to the physical arrangement of facilities on the shop-floor.

- *Control and Integration* involves the selection or design of a manufacturing planning and control system that best matches the design of the physical aspects of the manufacturing system.

- *Work Design* addresses the human aspects of the proposed system, for instance, the flexibility of operators and their responsibility for quality.

- *Implementation* is concerned with translating the proposed design into a real working manufacturing system.

- *Evaluation* concentrates on an ongoing evaluation of the new or modified system's operational performance.

Each of these components is associated with a set of guiding principles to assist the manufacturing system designer. This guidance is presented in the form of *key parameters* and *design option guides*. Key parameters comprise a collection of different objectives for each of the ten design stages, one of which should be identified as the primary objective for each stage. For example, key parameters relating to the *physical system design* phase include, amongst other things, output maximisation, inventory minimisation and volume or variety flexibility. The decision-maker will need to identify which of these parameters is most important in relation to that aspect of their manufacturing system design.

Design Option Guides (DOGs) then present a selection of alternative solutions to each design problem and indicate how each option is likely to affect the key objective of each design stage. For instance, one of the DOGs relating to *physical system design* presents the following alternative solutions to the problem of layout configuration : modular, flow-line, functional or a group technology arrangement. For each of these alternatives, the guide suggests which of the key parameters that a particular configuration would be most likely to achieve. In this case, for example, a flow-line arrangement would maximise output, minimise inventory but would not provide volume or variety flexibility.

By using the combination of key parameters and design option guides, DRAMA II does provide a comprehensive range of guiding principles to assist the manufacturing system designer. In addition, the scope of assistance extends to include strategic and operational aspects of the decision-making process, and also post-implementation issues. However, the guidance only goes as far as to recommend general solutions to particular design problems. As a result, the evaluation of design proposals is essentially presented as a qualitative assessment of cause-and-effect relationships between alternative design proposals. The methodology makes no provision for measuring or evaluating the effects of alternative solutions in quantitative terms.

This limitation is particularly interesting, given the apparent interactive nature of the design process. The methodology suggests that no aspect of the proposed manufacturing system is actually finalised until all ten design components have been addressed. In other words, it is expected that some aspects of earlier design solutions may have to be redefined as a result of problems that might not be discovered until much later in some of the other design phases. Since the methodology makes no mention of evaluating, in quantitative terms, how the different aspects of the system will perform together, it is difficult to see how the designer can ensure that the individual components of their system will perform effectively as an integrated whole.

For this reason, it is questionable if the most effective changes can really be made to earlier aspects of the design. From the viewpoint of this thesis in particular, the methodology does not consider how shop-floor aspects of the design will affect the loading on support functions. For example, the trade-offs between high volume and

high variety manufacture will have a significant affect on the workload placed on non-manufacturing areas but the guidance offered in DRAMA II would not be able to quantify these effects.

To summarise, the Lucas 'Five-Step' Approach and DRAMA II are, in many ways, complimentary methodologies. The former emphasises more detailed design and evaluation of certain aspects of a proposed manufacturing system. The latter approach presents an even broader view of the overall design process but less provision is made for detailed design issues. Most significantly, however, neither approach attempts any real evaluation of how non-manufacturing functions will affect the success of a new or redesigned manufacturing system. Therefore, on the basis that these two methodologies are representative of current best practice in manufacturing system design, there is a need for such techniques to extend the testing of manufacturing system designs to include support departments. It also suggests that a suitable tool is required to perform this kind of analysis.

4.3 The Design or Selection of a Production Planning and Control System

The concept of a production planning and control system has already been mentioned briefly in the context of manufacturing system design but will now be examined in more detail. While the design of a production system should also include the specification of a production planning and control system, this aspect of a manufacturing facility represents a major area of decision-making in its own right. As in the review of manufacturing system design methodologies, the discussion that follows will relate to what factors influence the decision-making process and to what extent the effect on support functions is taken into account when designing or selecting control systems.

4.3.1 Features of Production Planning and Control Systems

As the term 'production planning and control' suggests, this aspect of a manufacturing organisation is generally taken to imply a set of policies and procedures that are concerned with the day-to-day running of a production system. The term is often used inter-changeably with other terms such as 'manufacturing control', 'material control', or simply 'production control'. Burbidge (1979) defines production control as follows:

> *"Production control is the function of management which plans, directs and controls the material supply and processing activities of an enterprise, so that specified products are produced by specified methods to meet an approved sales programme; these activities being carried out in such a manner that labour, plant and capital available are used to the best advantage."*

This definition emphasises the scope of the production planning and control task; that is, the management of different shop-floor constraints such as production equipment, direct labour and materials. The process is essentially a support activity, many aspects of which are carried out by support functions. Ironically, however, given that the task aims to effectively manage a limited set of resources, little consideration is given to how the resources in the production planning and control function itself actually perform. This is evident from a closer examination of different types of control system and the methodologies relating to their design and implementation.

There are a number of well established production control techniques including Material Requirements Planning (MRP), Manufacturing Resource Planning (MRPII), Kanban and Optimised Production Technology (OPT). A full discussion of the relative merits of each of these and other approaches is beyond the scope of this thesis but details can be found in most production management texts (see, for example, Wild, 1989; Slack et al., 1995).

Manufacturing organisations will generally select a standard version of one of these established control methods or design a variation on one of these themes that provides a better match with their particular manufacturing environment. The type of control system that is eventually implemented will depend on a variety of factors. These will include the nature of the market within which a firm operates and also the choice of process technology adopted by a manufacturing system. In practice, manufacturing firms can often employ more than one production control technique or a hybrid of established techniques to manage different aspects of their operations (Turner & Saunders, 1994).

The type or types of control system employed by a firm will determine the volume of

data that will need to be processed to properly support the shop-floor. The timeliness and accuracy of data needed to successfully operate a control system will also vary. Larsen & Alting (1993), for example, differentiate between the information and data processing requirements of some well established production control techniques. A summary is shown inTable 4-1.

Production Control Philosophy	Information and Data Processing Requirements
Material Requirements Planning	Detailed
Kanban (i.e. Just-in-Time)	Few
Optimised Production Technology	Very Detailed

Table 4-1 Information Requirements for Different Production Control Philosophies (adapted from Larsen & Alting, 1993)

By implication, the volume of data to be processed will affect the workload placed on support functions, influencing the extent to which both information technology and support staff are employed in these areas. For example, Dale & Russell (1983) describe how the implementation of a new production control system in one particular company enabled the number of production control personnel to be reduced. Had a more data intensive control system been adopted then this number is likely to have increased, or else the control system would have been unable to properly support production.

On a similar theme, Hunter (1993) advocates that many firms should thoroughly investigate ways of eliminating production control transactions in order to alleviate the workload on support functions. He suggests, for example, that firms should reduce the number of operations that are tracked per order by making changes to the bill-of-materials structure in an MRP system to include more 'phantom' parts. While these actions may reduce the burden on support functions, Hunter's approach is really just overcoming the kind of constraints that should be investigated prior to implementing any control system.

In summary, given that control systems are so closely related to shop-floor performance and that they also depend on adequate levels of support staff to operate

them, one would expect that any decision to adopt a particular system will have considered its affect on existing support functions. In particular, one would expect that a decision-maker in this situation would have attempted to estimate the change in workload that a control system will impose on these functional areas. The next section will examine to what extent methods for designing and selecting control systems do consider these effects.

4.3.2 Formal Methods for Designing or Selecting Control Systems

There are notably fewer well established methodologies that relate to the design or selection of a production control system than exist for manufacturing system design. However, those that are available in the public domain can be assumed to be typical of current practice since they are likely to have been developed from studies of control systems in a variety of manufacturing organisations.

Kochhar et al. (1992) present one such methodology in the form of a structured framework for selecting manufacturing control systems. This takes the form of three different knowledge-bases to aid the choice of an appropriate system and to assess issues governing the implementation of that system. One of these knowledge-bases does consider the volume of transactions to be processed (see Kochhar & McGarrie, 1991) but only in terms of the extent to which computerisation will be needed to process all the data required. In other words, the framework makes no provision for assessing if the existing personnel will be able to cope with the volumes of work generated by a new control system. This may lead to operational problems given that a manufacturing system is so dependant on its control system for timely production of information and provision of material and labour for manufacture.

Maull et al. (1990) present an alternative methodology that integrates the design and implementation of a production control system with a firm's manufacturing strategy. The nature of this methodology is such that it is directed toward a computerised solution to the manufacturing control problem but does also propose to give consideration to other aspects of the business:

> *"The overall performance of the [control] system may be enhanced by changes to the infrastructure that support the software"*
>
> Maull & Childe (1993)

Indeed, this approach does emphasise infrastructure in terms of 'policies, procedures and practices' as being a fundamental aspect of any successful control system solution. Despite this claim, the methodology provides very little in the way of guidance or techniques for analysing how effectively personnel in non-manufacturing functions would be able to carry out any new policies, procedures or practices.

Another methodology, presented by Banerjee et al. (1994) focuses on both strategic and more detailed design aspects of production control systems. The approach can be viewed in terms of four stages that cover both the analysis and design of a control system. These four stages are depicted inFigure 4-2.

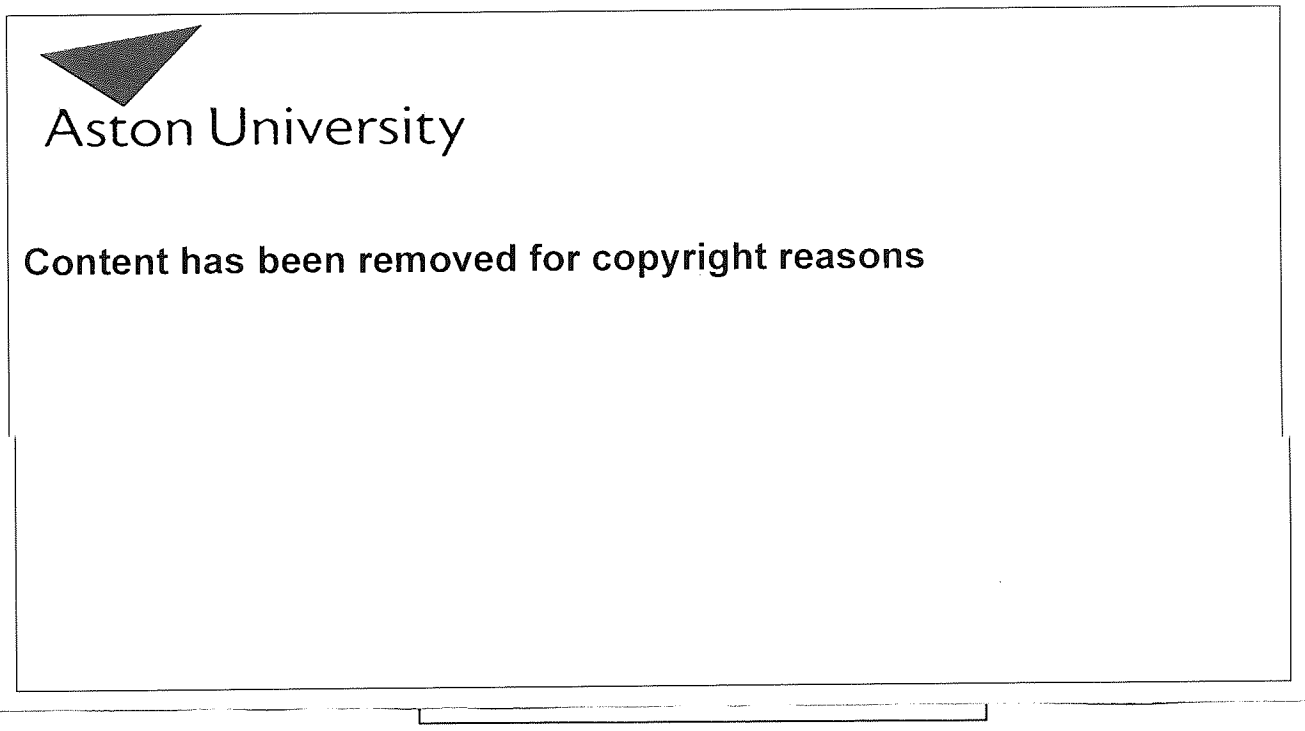


Figure 4-2 Production Control System Design Methodology (Banerjee et al., 1994)

The concept design stage is of most relevance to this discussion since it attempts to establish the information flows that will form part of the system and also the frequency in which they will occur. However, despite these aspects of the proposed system, the methodology makes no provision for actually quantifying these features of the design in terms of the workload that will be placed on production control functions within a firm. Consequently, decision-makers adopting this approach will not take the opportunity to test how those functional areas will be able to operate a proposed manufacturing control system.

In summary, it is clear that the concept of a 'control system' in manufacturing organisations relates to systems for effectively managing manufacturing resources but does not extend to managing the loading on non-manufacturing functions. In addition, a sample of methodologies, that have been developed to formalise the design or selection of production control systems, have been examined. This sample suggests that while some consideration is given to the infrastructure that will support a control system, little attention is paid to the detail of whether or not existing support functions will be able to manage the changes in workload that a new control system would introduce. Again, it also suggests that an appropriate method or tool is needed for carrying out this type of quantitative analysis.

4.4 Make-or-Buy Analysis

Decisions as to whether parts should be made or bought are some of the most problematic faced by manufacturing companies. These decisions can arise in a number of different circumstances that are relevant to the research described in this thesis. For example, they may occur as part of a strategic re-design of a manufacturing system or as a result of an ongoing programme of continuous improvement to manufacturing operations.

4.4.1 Conventional Approaches to Make-or-Buy Decisions

At an operational level, many businesses approach make or buy decisions from a purely cost-driven point of view, selecting which components to out-source by comparing the internal cost of manufacture with supplier's prices. Manufacturing capacity may be another factor. For example, sub-contracting may be deemed necessary to fulfil excess demands for items normally manufactured in-house, where the volumes concerned are not sufficient to justify additional capital investment.

Invariably, analysis based on cost or capacity issues will involve evaluating the financial gains or losses that would be incurred by making or buying a particular item. For example, most financial and operations management texts describe techniques for evaluating make-or-buy decisions in purely cost terms (see for example, Drury (1992); Slack et al. (1995)). To simplify the analysis, only marginal costs are considered. In other words, only those costs that are expected to change as a result of the decision will

be included in the analysis. Typically, when evaluating different options in this way, support department costs are assumed to remain fixed and therefore the effect of a make-or-buy decision on non-manufacturing functions tends to be ignored.

However, the nature of a make-or-buy analysis is such that the outcome of a decision will certainly affect the workload of some support functions. For example, retaining production in-house will increase the burden on activities such as production control and stores management needed to oversee the manufacture of these items. Alternatively, buying an item will place additional burdens on all of the functions involved in the procurement process such as purchasing, expediting, goods inwards and accounts payable. It was explained in a previous chapter that changes in the volume of transactions or other forms of loading will affect support functions in one of two ways. They will either be forced to invest in additional resources to maintain a satisfactory level of service to the shop-floor or, the level of support they are able to offer will deteriorate. The methods advocated in many texts for evaluating make-or-buy decisions are therefore inappropriate because they ignore some very significant effects that an outcome might have on support functions.

Elaborate costing models have been employed in an attempt to overcome these limitations. The activity-based costing approach (Cooper & Kaplan, 1988b) is one such example, but there are also many others (see, for instance, Yoon & Naadimuthu (1994); Darlington (1995)). However, these models all focus purely on costs and are, in fact, unable to predict how the service level provided by support functions will affect the performance of a manufacturing system.

This general criticism of costing methods concurs with work carried out by Dale & Cunningham (1983) specifically in relation to make-or-buy decisions. Using case studies, they confirm that decisions-makers are not satisfied with simplified costing approaches for make-or-buy analysis, mainly due to a lack of confidence that these methods can generate the true cost for each alternative. In particular, they suggest that decision-makers do recognise that overhead costs will be affected but realise that their costing systems are inappropriate. To overcome these limitations, Dale & Cunningham can only suggest that other qualitative factors are taken into consideration and that these decisions should be resolved jointly by members of all the affected

functions. They do not suggest how the impact of a make-or-buy decision on indirect areas might be quantified, either in terms of cost or in the level of support that these functions will still be able to maintain.

Bergen (1977) and Buchowitz (1991) introduce decision frameworks for make-or-buy analysis. Their purpose is to ensure that the same questions are answered in the same sequence by all functions involved in a make-or-buy exercise. Bergen, for example, suggests that such a framework will encourage actions to be taken that are more likely to benefit the whole organisation rather than just optimise one particular aspect of the business. This kind of approach may at least encourage the impact of decisions on overhead functions to be considered but does not provide any mechanism for actually quantifying, in financial or other performance terms, what these effects might be.

4.4.2 Make-or-Buy Analysis in the Context of Manufacturing Strategy

The significance of make-or-buy analysis is also well recognised in the context of manufacturing strategy formulation. This introduces new concepts in addition to the popular 'lowest cost' option described earlier. Venkatesan (1992), for instance, links the concept of strategic sourcing to make-or-buy. Buying-in is advocated where a supplier's experience of a critical manufacturing process and hence quality of goods they produce may be superior to one's own. In the same way, it is suggested that a company should restrict the range of goods actually produced in-house to those that are sources of competitive advantage.

While a full discussion of strategic-sourcing is beyond the scope of this thesis, any proposed action should still be evaluated in terms of how resources involved in the material acquisition process will be affected in the light of a new sourcing strategy. Venkatesan makes no mention of this aspect of make-or-buy analysis.

In contrast, Probert et al. (1993) comment on the lack of an overall methodology for the make-or-buy decision. For this reason the authors present their own four-stage process that incorporates both the strategic and financial aspects described so far. The methodology is represented inFigure 4-3.

Aston University

Content has been removed for copyright reasons

Figure 4-3 Make-or-Buy Methodology (after Probert et al., 1993)

This methodology does at least emphasise the need to weigh up the best combination of options on a quantitative basis. A spreadsheet-based or similar type of cost model is suggested as an appropriate tool for carrying out this evaluation. In terms of support functions, the methodology also emphasises that overhead costs, in particular material acquisition costs, should be carefully evaluated.

This represents a significant improvement on some of the more factory-focused approaches described earlier. However, given that the service level provided by support functions to the manufacturing system is often a constraining factor, the evaluation process should also consider that aspect as well as overhead costs. In addition, the ability of a spreadsheet-based model to properly predict the loading on support functions is questionable; this point will be illustrated later. Overall, the four-stage methodology proposed by Probert et al. would provide a comprehensive analysis of a make-or-buy situation if a more rigorous method for evaluating the impact on support functions were employed.

4.5 Summary

This chapter has looked in detail as some typical decisions that will be taken in the course of designing a manufacturing system. Methodologies aimed specifically at these types of decision have been reviewed. The purpose has not been to lay criticism at any one methodology in particular, those that have been examined merely serve as an indicator of current best practice in these different areas. A common theme running through the different techniques that have been reviewed is that they emphasise the importance of making any decision in the context of how it will affect the whole business rather than just a single functional area. It is therefore surprising that the same methodologies make little or no attempt to rigorously test how the changes in workload that a particular action might impose on support functions will affect the success of these decisions.

Arguably, since the effects on manufacturing areas tend to be analysed in some detail, the lack of a suitable tool or technique for quantifying the effects on support functions is one reason why these aspects of the business tend to be ignored. The application of this tool or technique would ultimately need to form part of a new or revised set of methodologies relating to different types of manufacturing system-related decision. The next chapter will therefore investigate the kind of evaluation technique or modelling tool that decision-makers would need to properly anticipate how manufacturing systems will perform given the potential constraints that support activities might impose.

5. Modelling and Analysis of Manufacturing Systems

Preceding chapters have identified limitations in the way manufacturing systems are designed, and particularly in the way these designs are tested prior to implementation given the impact that support departments can have on a system's ultimate performance. This chapter therefore examines a range of modelling techniques for analysing manufacturing systems with a view to establishing a more appropriate way of evaluating design decisions.

5.1 What is Modelling?

Models of various types are used to investigate solutions to many kinds of business problem (Szymankiewicz et al., 1988). A model will be created to represent those aspects of the business that are seen as relevant or likely to be affected by a proposed course of action. The model can then be used by decision-makers to experiment with new ideas and solutions more economically, quickly and safely than would be the case if those same aspects of the real business had to be disturbed (Pidd, 1992a). The process of creating a model also helps decision-makers to understand the system being studied (Seila, 1995; Elder, 1995b) and provides insight into aspects of the business that might otherwise be ignored or taken for granted. Consequently, modelling should form an important part of any decision-making process.

Models can take a variety of forms even within the manufacturing system design process. A physical or scale model of a manufacturing facility will enable any physical constraints that a proposed change might introduce to be identified. For example, a scale model would enable decision-makers to investigate how easily operators and material handlers would be able to move between a new arrangement of machines on the shop-floor.

In contrast, decision-makers need to adopt an altogether different kind of modelling approach in order to investigate the behaviour of a manufacturing system in terms of the interactions between its different elements and how the performance of the whole system might change over time. Decision-makers tend to employ mathematical or logical models rather than scale models for this type of application (Pidd, 1992a). These types of model consist of mathematical equations or logical relationships to

represent the behaviour of objects in the real world. Since this thesis is concerned with the relationships between manufacturing and support functions, the discussion will relate mainly to mathematical and logical modelling techniques rather than other types of models.

Another important and closely related factor is the scope and level of detail to be included in a model, given that it is intended only to be a simplified representation of a real system. Both scope and level of detail will depend on the type of problems that a model will be used to study but a model must also be realistic, and therefore any assumptions or approximations need to be carefully considered. In view of this, several concepts are often used to classify modelling approaches:

- *Static vs. Dynamic Models* : This concept relates to how closely a model describes the time varying behaviour of a real system. Coyle (1977) suggests that all commercial and social organisations show dynamic behaviour. In other words, the variables by which the condition of a system is measured (such as inventory, sales or profits) fluctuate as time passes. Dynamic models attempt to capture this time varying behaviour but static models approximate these characteristics (Shannon, 1975).

- *Deterministic vs. Stochastic Models* : This concept relates to how sources of uncertainty that may exist in real systems are captured in models of those systems. A deterministic system is one whose behaviour is entirely predictable; in other words, the timing of every event can be predicted with absolute certainty. In a stochastic system, however, a statement can only be made about how likely certain events are to occur or how long they will take to complete (Pidd, 1992a). Uncertainty is closely related to the dynamic behaviour of a system since variability contributes to this type of behaviour.

- *Continuous vs. Discrete Models* : This refers to how time is represented in dynamic models and therefore how the elements in a system change state (Seila, 1995). Each change of state in a discrete system, such as a machine changing from *running* to *idle* or a document changing from *completed* to *signed-off*, is clearly identifiable. In continuous systems

state changes are more gradual; for example, the temperature of a furnace will change continuously from a low to a higher temperature. Continuous models are sometimes used to approximate discrete systems but the nature of any real system and the type of problem being addressed will determine how realistically it can be modelled by either of these approaches.

The next section will examine how these modelling concepts relate to the type of model that should be used to analysis manufacturing system design decisions.

5.2 The Requirements of a Modelling Tool for Evaluating Manufacturing System Design Decisions

The thesis has so far established that the interactions between support functions and a manufacturing system should be modelled to identify any constraints that support departments might impose on the shop-floor. Some important characteristics of these different functional areas and the interactions between them will now be examined from a modelling point of view. Practical considerations for a modelling tool to be used by manufacturing system engineers will also be discussed.

5.2.1 Features of the Environment to be Modelled

By implication, the majority of delays incurred by manufacturing and support functions interacting with each other will be due to information flows, such as a production control department issuing works orders to the shop-floor, or the shop-floor sending raw material requisitions to stores. Some types of interactions may also involve a combination of material and information flows. For example, delays will be incurred as material moves from a supplier, through goods inwards to a machine-shop, and there may also be other administrative delays incurred in completing any paperwork involved in this process.

Since decision-makers will ultimately be interested in how these delays are going to affect manufacturing system performance they need a model that will be able to represent the complex sets of interactions within and between different functional areas that contribute to these delays. For example, within support departments documents

may pass through several administrative areas before reaching the shop-floor. During this time a document is likely to go through a series of transitions such as being processed, duplicated, filed or moved to other departments within a firm. At each stage, administrative personnel will be dependant on upstream processing activities having completed their work on a document and then transferring it to the next processing stage. The limited availability of supervisors or managers for signing-off or authorising some types of document may also affect its progress through the organisation. Support department personnel might also be interrupted from the task of processing a document by meetings or having to interact with another department or company to obtain a piece of information.

This all suggests that the behaviour of the different elements making-up support functions is likely to be complex and the time for a document to pass through the system will vary, due to events or interruptions that occur randomly. In modelling terms, these factors emphasis that support departments are characterised by dynamic and stochastic behaviour. A modelling tool should therefore be able to represent dynamic and stochastic aspects of document processing activities in order to properly evaluate their impact on manufacturing system performance.

The operation of a manufacturing system itself is also complex and characterised by dynamic behaviour and uncertainty (Coyle, 1977; Parnaby, 1986). For example, sources of uncertainty will include production losses due to machine breakdowns, the inter-arrival time of parts and variability in processing times (Law et al., 1994). Variability in the performance of operators will be another factor. For this reason, a modelling tool should also be able to model the dynamic and stochastic behaviour of shop-floor entities, and indeed, model how this might be influenced by equivalent behaviour in support functions. In other words, these different parts of a business should be represented in the same model.

Another feature of manufacturing and non-manufacturing environments is that they can both be described in terms of discrete elements, such as people, parts, documents, machines, desks, computers and so on. In modelling terms, therefore, not only is a manufacturing business characterised by complex and dynamic behaviour, but distinct or discrete changes in the states of the different elements that make up the organisation

can also be identified. For example, people in a manufacturing system will process discrete batches of parts while their counterparts in an administration system might process discrete batches of documents. Interactions between these different functional areas will also involve distinct exchanges of information or material; for example, a production-control clerk, by issuing a works order to the shop-floor, will be instructing an operator to begin manufacturing a batch of the particular product requested on that document.

These different characteristics suggest that to be able to properly represent delays a model should portray the dynamic and stochastic behaviour of the discrete elements that make up a manufacturing business. While these factors represent important functional requirements, the next section will look at some other practical considerations.

5.2.2 Practical Considerations for a Modelling Tool

As well as providing a way of evaluating and comparing alternative design options, modelling is also a very useful method of communicating new ideas to those people that will be affected when a new manufacturing facility is eventually implemented (Gogg & Mott, 1992; Seila, 1995). For this reason, the individuals most involved in making the decision and those who will operate the new system should be involved in the model development process. This introduces two practical considerations; both the time that these people will have available to build a model and also their understanding of different modelling techniques are likely to be limited. Consequently, if a modelling tool is to be used in practice to evaluate manufacturing system design alternatives then it should allow people with only a limited knowledge of modelling techniques to quickly build models (Brewer, 1995). The extent to which these people can be distanced from the mathematics or logic of a model contributes significantly to the ease and speed of the model development process.

This suggests that much of the logic that describes the behaviour of a manufacturing business ought to be pre-defined in the modelling tool itself, allowing a model-builder to simply configure these elements together to create a model of their particular organisation. However, such a tool should also be flexible enough to model the

features of both manufacturing and support activities within a range of firms. In addition, the model development process should be carried out, as far as possible, using terminology and concepts that the model-builder understands rather than using specialist modelling jargon. This implies that models should be configured from entities and systems typically found in a real manufacturing organisation such as operators, machines, documents, procedures and so on.

In summary, both a manufacturing system and its supporting functions are characterised by dynamic and variable behaviour. A modelling tool that captures these features is needed to properly evaluate the performance of a new manufacturing system. This also implies that the manufacturing and non-manufacturing elements of a business should be modelled as an integrated whole rather than in isolation. For practical reasons model building should be relatively quick to complete and carried out by people involved in the decision-making process rather than by modelling experts. This means that either the modelling approach should be easy to understand or the approach packaged within an easy-to-use modelling tool in such a way that model-builders are presented with concepts that are familiar to them.

5.3 A Review of Techniques for Modelling Manufacturing Systems

Having now established some functional and practical requirements of a modelling tool for assessing manufacturing system design decisions, this section will look at some of the different techniques that are available and determine which is the most appropriate for investigating this type of problem. The discussion will relate to modelling approaches rather than specific commercial modelling tools.

5.3.1 Opinion Models and Rules-of-Thumb

Gogg & Mott (1992) coin the term *opinion model* to describe approaches that are based more on 'rules-of-thumb' than on any formal analysis. In other words, this type of approach relates to situations where decision-makers decide which course of action to take based solely on their own personal experience, or on the practices of other apparently similar firms, rather than attempting to actually quantify the consequences of alternative actions. By their very nature, opinion model solutions derive from qualitative evaluations of engineering or business problems. For that reason, opinion

modelling is not a modelling technique in the sense of a mathematical or logic-based approach, but it still warrants discussion here since solutions to some aspects of a manufacturing system design are formulated in this way.

For example, a methodology for selecting production planning and control systems using knowledge-bases was presented in the previous chapter (Kochhar et al., 1992). This can be described as an opinion modelling tool since it can be used by decision-makers to provide general solutions to the production control aspects of a manufacturing system. Thomas & Davies (1996) present a similar type of model that effectively imposes a supporting infrastructure onto a manufacturing system design. In practice, however, the dynamic behaviour and inherent variability within different functional areas contributes to making each particular manufacturing system unique in some way. For this reason, generic solutions derived from opinion models can be misleading even when restricted, for example, to production control system problems (Boughton, 1995). In other words, the successful production control solution for one manufacturing company may not necessarily be the most appropriate system for an apparently similar organisation (Ptak, 1991) and so general solutions to this type of problem are often inappropriate.

Specifically, the rule-of-thumb approach lacks any quantitative assessment of alternative design solutions. This means that decision-makers cannot even anticipate how shop-floor performance might be constrained by average support department workloads. Lewis (1994), for example, presents the case of a firm that developed some general rules-of-thumb for a major make-or-buy exercise but which ignored the impact on support department areas. The company adopted a policy whereby any item having either less than 20 minutes processing time, or that could not be produced in any of its existing manufacturing cells, should be out-sourced. The decision-makers assumed that this approach would reduce shop-floor complexity but they did not attempt to quantify how effective their policy would actually be in achieving this objective. In addition, they did not investigate how the complexity of support activities, such as procurement, might be affected. As a result, arrears and inventories of bought-in items increased to the point where shop-floor performance actually deteriorated.

For the individuals concerned this bad experience should mean that 'better' rules-of-thumb can be applied in future projects. Indeed, opinion models and rules-of-thumb can provide an informal and relatively quick means of producing generic solutions to some manufacturing system design problems. However, since no quantitative evaluation is performed these types of model cannot provide any insight into how delays in either manufacturing or support areas might affect business performance. In other words, a more sophisticated and quantitative modelling approach is required.

5.3.2 Static Mathematical or Spreadsheet Models

A static mathematical model uses linear equations and average values for process times, costs and so on to describe the behaviour of a real world system. This type of modelling is probably the most basic of the approaches that do attempt to quantify the consequences of decisions. Spreadsheets are widely used for evaluating decisions in this way and are particularly popular because they allow end-users to create models without the help of modelling experts (Floyd et al., 1995).

In terms of the manufacturing system design process, spreadsheets models are often used to assess capacity or cost-related issues. For example, a survey of UK businesses conducted by Berry & McLintock (1991) identifies the widespread use of spreadsheet tools for building financial models, including models for evaluating manufacturing decisions. The study also suggests, however, that spreadsheet models are typically deterministic and quite basic in their representation of the relationships between the different aspects of a business:

> *"Typically the understanding of cost behaviour in a model is based on the traditional fixed variable split, where variability implies a simple proportional relationship."*

More recently, Clarke & Tobias (1995) reviewed the application of spreadsheets for corporate modelling which is relevant here since, by implication, a 'corporate' approach should consolidate both manufacturing and non-manufacturing aspects of a business into the same model. Speed and ease of model creation are again identified as the main advantages of this approach but the authors claim that since spreadsheets inhibit 'richness' in modelling the behaviour of organisations, they are not appropriate for evaluating the effects of operational or engineering decisions:

"Corporate models are mostly used for strategic financial analysis and are becoming, in general, less concerned with physical logistics and the detail associated with the lower corporate structural levels".

Again, the criticism derives from the fact that spreadsheets and other static mathematical modelling techniques use average rather than variable values to represent the behaviour of elements in a real system. It can actually be demonstrated from first principles that assuming constant values for process times that actually vary can distort even short-term predictions of the performance of an individual person or machine (Law & Kelton, 1991). Consequently, since an administrative or manufacturing system is made up of many different but inter-dependant people and machines, the predicted performance of such a system is certainly likely to be distorted if variability is not included in a model of that system.

Another limitation of spreadsheet models is that while they are relatively easy to use, they contain only written descriptions of the relationships between the elements of a real system. Berry & McLintock (1991) suggest that this type of presentation does not convey any assumptions made about a real system as clearly as a graphical representation and, consequently, the authors raise serious concerns about how easily the validity of spreadsheet models can be tested. This aspect of spreadsheet models also implies that they are unlikely to be suitable for communicating proposed changes to affected personnel within a firm.

In can be concluded that spreadsheets or other static mathematical modelling approaches provide a relatively quick and easy way for decision-makers to build models. However, these types of model lack the necessary richness in terms of being able to predict how dynamic behaviour and uncertainty will affect the various sources of delay that can constrain manufacturing performance.

5.3.3 Analytical Modelling and Queuing Theory

While spreadsheets use linear equations to describe static behaviour, an alternative mathematical approach involving probability equations is sometimes used to model the time-varying or dynamic behaviour of a system. The term *analytical modelling* is often used to describe this type of approach and models based on *queuing theory* are

often used to analyse real systems in this way (Suri & Diehl, 1987). Queuing theory uses the concepts of *customer*, *server* and *queue* to describe the system being modelled (Hall, 1991):

- A *customer* represents something that is waiting to be served and might include, for instance, an operator waiting to use a machine or a document waiting to be processed.

- A *server* provides the service; for example, an accounts clerk that processes invoices or a machine that manufactures sub-components both represent different types of server found within manufacturing firms.

- A *queue* is a group of customers waiting to be served; for example, this might include a batch of purchase requisitions waiting to be signed-off.

Models are created by identifying customers and servers within a real system and then describing relationships between them; probability distributions are used to represent arrival times of customers to the queue and also the time taken for each customer to be served. However, many simplifying assumptions embedded in basic queuing theory mean that, in practice, it is of little use in predicting the performance of even simple systems (Byrd, 1978).

Limitations include the assumption that the arrival of each customer is not related to the arrival of previous customers (Hall, 1991). This restricts queuing theory's ability to accurately model systems where several customers progress through a number of processing stages, such as a typical scenario where *Customer2* only progresses through *Server1* to arrive at *Server2* after *Customer1* has done the same. In this example, the arrival time of *Customer2* at *Server2* should be dependant on the earlier arrival time of *Customer1* at *Server2* but would not be represented in a queuing theory model.

Clearly, this kind of inter-dependency between the arrivals of customers is prevalent in both manufacturing and administrative systems. For example, works orders might be represented as 'customers' in a queuing model of an administrative system and each have to progress through a series of 'servers' in the form of an MRP system, a production planner, a cell-supervisor and finally a machine operator. This limitation

means that basic queuing theory is inappropriate for evaluating manufacturing system design decisions.

Queuing-network theory overcomes many of the limitations of basic queuing theory and can model a collection of servers acting together to serve customers (Hall, 1991). In addition, while the mathematical concepts involved in queuing-network theory might not be familiar to people involved in the manufacturing system design process, software packages are now available for creating these types of model but using terminology that model builders will more readily understand (Snowdon & Ammons, 1988; Suri et al., 1995). As a result, this type of analytical modelling has become popular in recent years as a means of testing the dynamic and stochastic behaviour of manufacturing systems (Heuttner & Steudel, 1992; Papadopoulos et al., 1993; Askin & Standridge, 1993).

Ultimately, however, certain characteristics of manufacturing organisations are not easily represented by mathematical equations and so even queuing-network models still incorporate some major unrealistic assumptions (Chaharbaghi, 1990; Jackman & Johnson, 1993). These assumptions include, for example, that :

- the system being modelled is operating in steady-state
- service times vary according to exponential distributions

The assumption of steady-state operation ignores the effects of transient behaviour. It was argued earlier in this chapter that tasks in administrative functions might, for instance, involve personnel in one department having to interact with personnel in another department, or even in a different organisation. Interruptions of this kind will affect the time taken to complete administrative tasks. Similarly, manufacturing activities will be interrupted by machine breakdowns or the rescheduling of jobs. These different examples illustrate sources of transience in both manufacturing and non-manufacturing aspects of an organisation and demonstrate why steady-state performance is difficult to achieve or maintain for any length of time in a real system (Jackman & Johnson, 1993). In addition, these types of interruption are likely to have a complex effect on service times for both manufacturing and support activities, to the extent that modelling these times using only statistical distributions, rather than incorporating entity-interactions into the model, is unlikely to capture the true

behaviour of a real system. Consequently, these assumptions represent another limitation of this type of modelling approach.

Both queuing theory and queuing-network theory also assume that the participants of a real system can be modelled as a customer, server or queue. However, given the scope of model that it has been proposed necessary for evaluating manufacturing systems, some entities may need to be modelled as both a customer and a server. For example, clerks in a buying department might queue to use a computer terminal or to get purchase orders signed-off by their supervisor. In this scenario, the clerks represent 'customers' waiting to be served. However, the same model might also have to represent purchase requisitions progressing through different processing stages, including the buying department. In this situation, buying clerks would represent 'servers', providing a service to purchase requisitions.

Overall, while analytical models can represent some aspects of the variability and dynamic behaviour found in manufacturing organisations, they also make some basic assumptions that limit their ability to predict how support functions might constrain the performance of a new or re-designed manufacturing system.

5.3.4 Process Modelling

Process modelling offers decision-makers a systematic way of representing the structure of a firm's manufacturing activities (Busby & Williams, 1993). Creating a process model generally involves documenting or describing the various interactions and flows of material or information that occur between different aspects of an organisation. While several methods are available for describing a business in terms of a process model, the discussion that follows will concentrate on two methods that might typically be used to represent an administrative or a manufacturing system; RADs (Role Activity Diagrams) and the IDEF methodology (ICAM Definition).

5.3.4.1 Role Activity Diagrams

Creating models using Role Activity Diagrams (RAD) involves conceptualising the different elements of an organisation into roles, activities, goals (a state to be reached) and interactions (Huckvale & Ould, 1994). Figure 5-1 shows a simple engineering

design process that has been modelled using RADs and provides an overview of the main features of this approach.

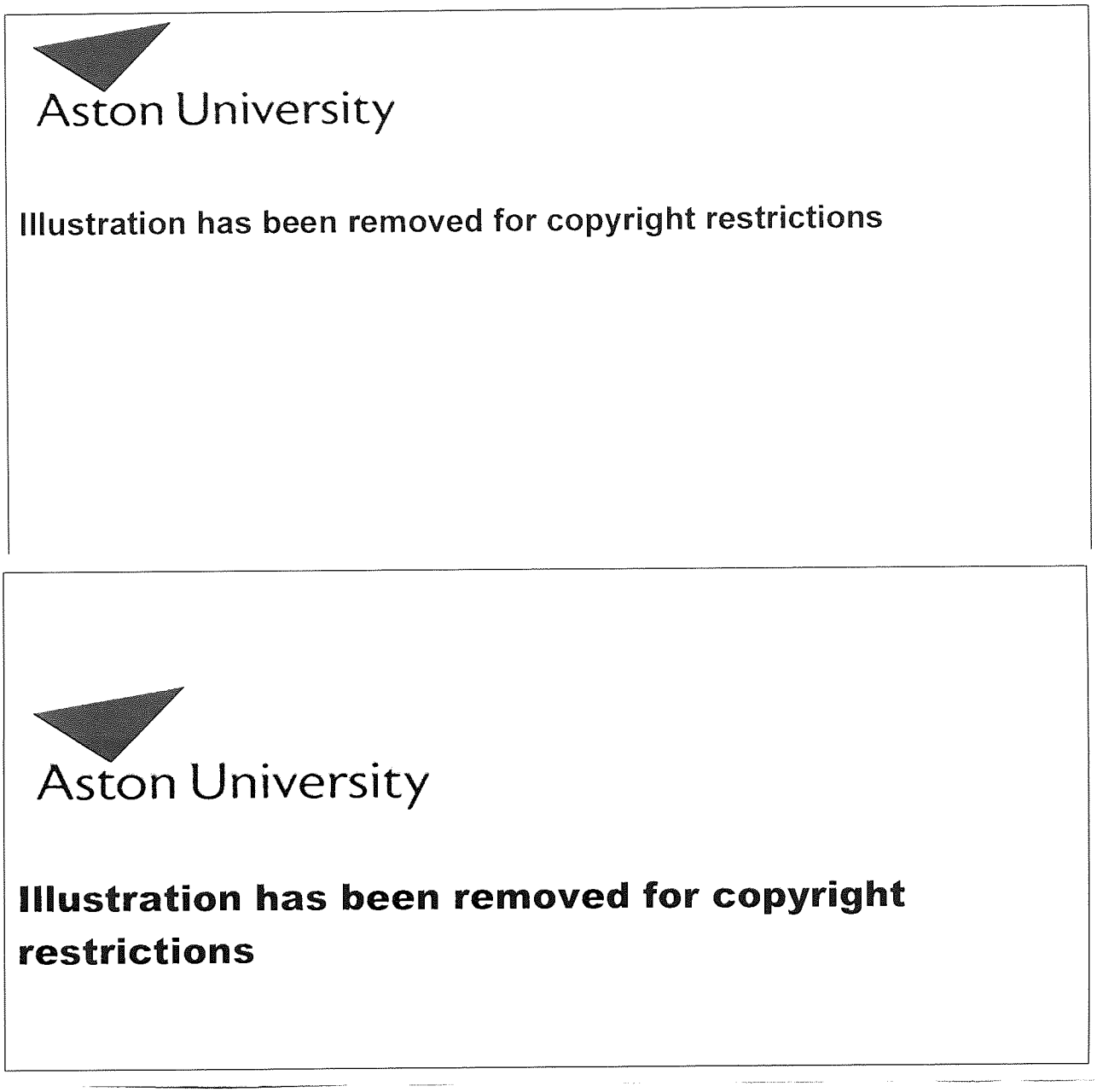


Figure 5-1 An Engineering Design Process : Modelled Using Role Activity Diagrams (after Ould, 1995)

This example is relevant here since in a jobbing-shop, for instance, the completion of this support activity will determine how long a manufacturing system has to actually manufacture goods. The diagram shows a sequence of activities that are performed by an individual or group of designers and project managers, and also shows some of the interactions that take place between these different sets of participants. The model is effectively a template for multiple instances of the design process that may occur

simultaneously. For example, separate design projects may be carried out in parallel for Customer A and Customer B, each one following the same sequence of events as depicted in the model.

While this type of model can provide a very graphical representation of the interactions that take place between a manufacturing system and its support functions, it provides very little in the way of quantitative data to help a decision-maker understand how existing personnel would cope if the number of instances of any process increased. For example, the model could not predict how the current design team would cope if an additional design project had to be started before the existing jobs for Customer A and Customer B could be finished. Indeed, the RAD notation allows the concept of a role (in this case a *designer* or *project manager*) to represent either a single person or a group of individuals which means that models convey no indication of the level of resourcing available to carry out each process. As a result, a RAD model could not even be used to model changes in the static loading of support functions that resulted from manufacturing system design decisions.

Huckvale & Ould (1994) promote the technique as being appropriate primarily for modelling business process designs as part of a re-engineering exercise but it is also relevant in the context of the manufacturing system design process. They emphasise that models allow decision-makers to clearly describe and communicate the 'as-is' and 'to-be' business process designs for an organisation but Huckvale & Ould offer very little in the way of advice to decision-makers as to how they might then use the model to analyse alternative design proposals, except:

> *"Some simple metrics of the efficiency of the process may be derived by counting the numbers of actions and interactions. A process (or role) with fewer interactions or a lower ratio of interactions is likely to be better."*

This statement, merely represents a common-sense strategy for improving a set of related activities within a firm rather than, as the authors suggest, a method for performing quantitative analysis. In a slightly more recent publication, Ould (1995) does acknowledge that a quantitative model requires data that is strictly outside the sort of model provided by RADs. As a result, he suggests using the approach of *system*

dynamics to perform quantitative analysis on a RAD process model.

System dynamics is a simulation approach that represents a system as a series of 'flows' of material between 'stocks'. It is a continuous modelling approach, typically applied to predict the stochastic and dynamic behaviour of discrete systems, such as organisations or even whole nations. It will be demonstrated in the next chapter that systems dynamics makes assumptions about the behaviour of systems that makes it inappropriate for evaluating manufacturing systems or support functions. Nonetheless, the suggestion that system dynamics be used to evaluate these models emphasises that RADs are perceived only as a technique for representing the interactions between administrative and manufacturing systems in purely qualitative terms. As such, Role Activity Diagrams might help decisions-makers understand, and communicate to others, how manufacturing system design decisions might affect support functions but would be unsuited to actually evaluating where support departments might constrain or limit shop-floor performance.

5.3.4.2 The IDEF Methodology

IDEF (ICAM Definition) was originally developed by the US Air Force as part of their Integrated Computer Aided Manufacture Programme to support its aircraft manufacturing activities, but has now become a collection of structured graphical tools for describing the architecture of more general manufacturing systems (Bennett & Forrester, 1993). Table 5-1 summarises some of the different IDEF techniques that are available.

Method	Name
IDEF0	Functional Modelling
IDEF1	Information Requirements Modelling
IDEF1X	Database Design
IDEF2	Simulation or Dynamic Modelling
IDEF3	Process Description Capture

Table 5-1 IDEF Techniques (adapted from Plaia & Carrie, 1995)

While this complete set of modelling techniques is intended to allow decision-makers to describe and analyse different aspects of organisations, the majority of documented

attempts to apply these methods within manufacturing companies have been limited to IDEF0 (Colquhoun et al., 1993; Wu, 1994; Booth, 1995). IDEF0 is used to create functional models of an organisation. In other words, these types of model consist of a network or hierarchy of diagrams representing both the manufacturing and support activities of a firm, in order to illustrate how each functional area can affect others in the business.

Figure 5-2, for example, uses the IDEF0 notation to model a simple make-to-order process that includes interactions between shop-floor and support functions. The notation uses the concepts of *inputs*, *outputs*, *controls* and *mechanisms* to describe the relationships between different activities. In this case, the model illustrates the inter-dependency between three different functions or activities, namely *production scheduling*, *manufacture* and *delivery*. For instance, the *output* of the *production scheduling* activity *controls* the *manufacture* of finished goods. In turn, finished goods represent an *input* to the *delivery* aspect of the process.

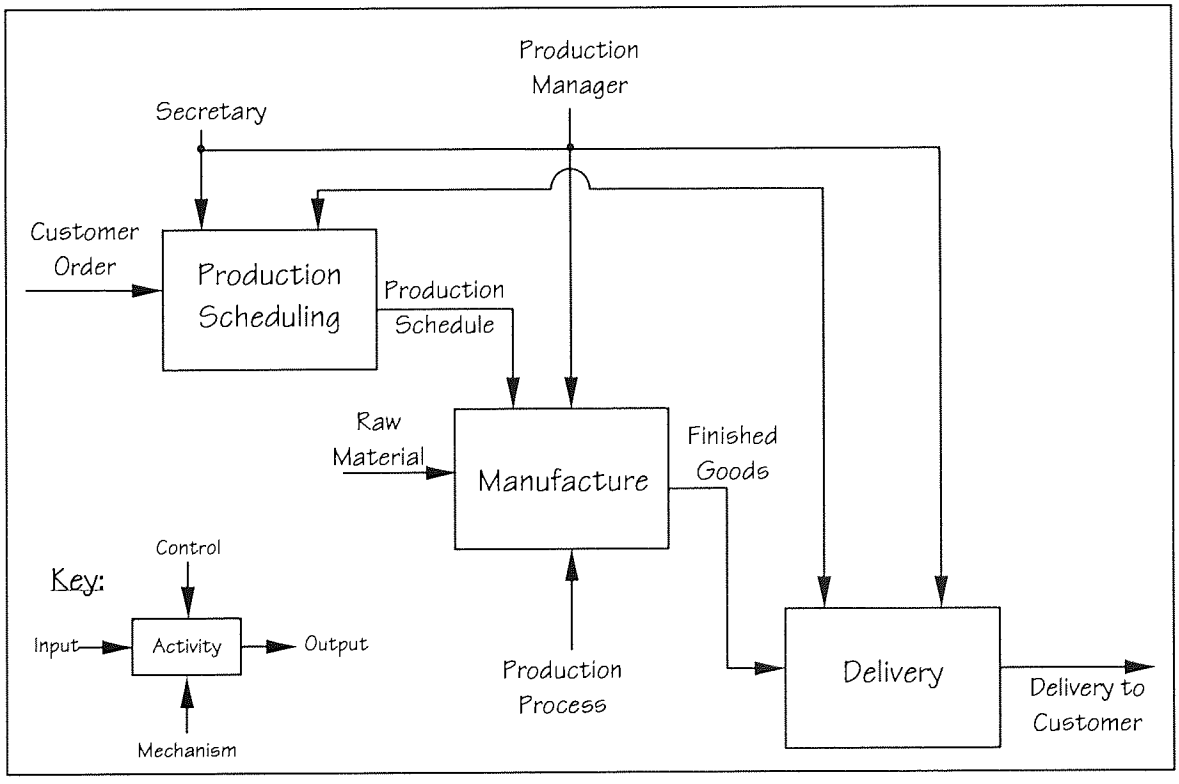


Figure 5-2 IDEF0 Model of a Make-to-Order Process
(adapted from Plaia & Carrie, 1995)

While the notation of an IDEF0 model is somewhat different to the Role Activity Diagram notation described earlier, both methods essentially produce the same type of model; that is, a wholly qualitative and graphical representation of information and

material flows within an organisation. Like a Role Activity Diagram, an IDEF0 model is also just a template for multiple instances of the make-to-order process that may take place in parallel, but the lack of any quantitative data again implies that the model cannot be used for analysis or experimentation. In other words, since the models make no reference to numbers of resources such as people or machines that will perform these activities, decision-makers cannot predict how effectively or quickly instances of a make-to-order process will be performed.

In addition to the lack of quantitative data, Busby & Williams (1993) also suggest that IDEF0 models represent processes in a deterministic way. In Figure 5-2, for example, a decision-maker would find it difficult to represent how the *manufacture* stage would be affected by machine breakdowns. Similarly, while the methodology would support each activity being modelled at lower levels of detail to those shown, the notation could not easily describe a set of different routings for parts flowing through a manufacturing system.

While IDEF0 can describe the activities performed by different functional areas, an IDEF1 or IDEF1X model will describe an organisation's information requirements (Bravoco & Yadav, 1985; Wang et al., 1993). This could form the basis for specifying the information systems needed to support a manufacturing system. Again, however, these types of model provide only a graphical and qualitative representation of this particular aspect of a manufacturing organisation. This means that decision-makers could not use these information models to evaluate, for instance, how manual or automated information systems might affect the performance of shop-floor activities.

In contrast, IDEF2 is often presented as being able to analyse the dynamic behaviour of a real system (Wu, 1994; Plaia & Carrie, 1995). For example, Bravoco & Yadav (1985) suggest that IDEF2 models allows decision-makers to describe the time-varying behaviour of manufacturing systems in such a way that the descriptions can be analysed to generate measures of a system's performance. However, Bravoco & Yadav also imply that the analysis stage must actually be performed using an additional third-party modelling tool, typically a simulator. A similar conclusion can also be drawn from the description of IDEF2 offered by Wang et al. (1993). In other words, an IDEF2 model really only provides a description of those characteristics of a

system which contribute to its dynamic behaviour, such as specific logic or timing aspects that can be associated with different activities. This is not the same as a model for evaluating dynamic behaviour, although an IDEF2 representation may provide the necessary data to build such a model using another modelling technique.

The IDEF3 methodology compliments the IDEF2 approach by providing additional information to describe a system's dynamic behaviour:

> *"[IDEF3] captures precedence and causality relations between situation and events in a form that is natural to domain experts."*
>
> Plaia & Carrie (1993)

Again, it is implied that this technique provides a way of documenting how an organisation works in order to build another type of model for actually performing quantitative analysis. Simulation is the modelling technique typically used for performing this type of analysis and will be discussed in more detail in the next chapter. However, the accepted role of simulation emphasises that neither the IDEF2 nor IDEF3 modelling approach are themselves enough to provide manufacturing system engineers with a means of evaluating alternative system designs.

Overall, process modelling offers a structured way for decision-makers to describe how the various functions within a manufacturing organisation interact and to represent the relationships between the different activities which these functions perform. In the context of the manufacturing system design process, this helps decision-makers to understand how a new manufacturing facility will interact with support functions. In addition, the graphical nature of process models means that they can be used for communicating new ideas to those people that will be affected by changes. However, process models provide only descriptions of those characteristics that contribute to dynamic behaviour. In other words, process models are not actually able to assess the effects that a particular proposal might have on other functional areas. It can therefore be concluded that process models themselves cannot be used to analyse a manufacturing system.

5.4 Summary

This chapter has reviewed a variety of modelling techniques with respect to their ability to properly analyse manufacturing system design decisions. The review has compared each technique in terms of its ability to model important characteristics of both the manufacturing system itself and also support functions, namely dynamic behaviour and uncertainty. Any suitable technique must also be practical for manufacturing system engineers to use in terms of the level of understanding and also time required to create models.

Techniques based on rules-of-thumb, spreadsheet analysis, queuing theory and process modelling have been assessed but each fails in some way to represent important features of either shop-floor or indirect activities that could affect the outcome of a manufacturing system design exercise. Consequently, another type of modelling approach is required for investigating this class of problem. The next chapter will therefore look at the advantages that simulation offers over the methods described in this chapter.

6. The Limitations of Current Simulation Tools

Each of the modelling approaches described so far fails in some way to provide an easy-to-use model that can assess the impact of delays between a manufacturing system and its support functions. Simulation is another type of modelling approach and has been extensively used to predict the consequences of a range of business decisions (Pidd, 1992a; Szymankiewicz et al., 1988). This chapter will demonstrate the potential advantages of adopting a discrete-event simulation approach for analysing the combined behaviour of a firm's manufacturing and support activities but will also identify limitations of the various simulation tools that are currently available for modelling this type of problem.

6.1 Adopting a Discrete-Event Simulation Approach

Simulation is one of the most frequently used systems analysis techniques since it is more suited to studying complex and dynamic systems than any other method (Seila, 1995). The term 'simulation' is actually often used in quite a general sense to describe any technique that allows experimentation to be performed using models of a real system. In some disciplines, for example, the use of spreadsheets or queuing theory models is described as simulation (Mould, 1990). However, in the context of the manufacturing system design process the term is generally restricted to describing a more specific set of computer-based modelling techniques, ranging from *continuous* to *discrete-event* simulation (Carrie, 1988).

The concept of continuous and discrete systems was introduced in the previous chapter; the terms refer to how time affects the dynamic behaviour of a real system and therefore how the elements in that system will appear to change state. Continuous simulation and discrete-event simulation systems attempt to model the characteristics of continuous and discrete systems respectively. A discrete-event model represents the behaviour of the different entities in the real system using procedural logic statements. For example, discrete-event logic might describe the conditions or events that cause a person to progress from a state of being *idle*, to *walking*, to *operating* a machine and eventually becoming *idle* again. By attributing times and sources of uncertainty to each of these events a simulator can model the dynamic behaviour of individual

entities in a system and thereby also model the overall dynamic behaviour of a large system such as a manufacturing organisation.

In contrast, a continuous simulation model will only approximate this type of behaviour using mathematical equations. *System dynamics* (Forrester, 1958), for example, is an established continuous simulation technique (Pidd, 1992a). This particular approach uses differential equations to describe a system in terms of information flows, material flows and a network of feedback loops. System dynamics has been applied to model different aspects of manufacturing system performance (Love, 1980; Tucci et al, 1991). In practice, however, a manufacturing business is a discrete system in the sense that it is made up of distinct entities such as people, departments, documents, batches and so on that interact with each other but can also exist and change state independently of each other. For this reason, a discrete-event simulation model will be a more natural representation of a manufacturing business than a continuous model since it will more closely represent the same discrete elements that make up the real system.

In addition, the procedural logic that makes up a discrete-event model will more closely describe how entities in a real manufacturing business interact. This aspect of discrete-event simulation is particularly relevant since manufacturing system engineers need to assess how interactions between a shop-floor facility and its support functions will affect manufacturing system performance. Furthermore, while all models make assumptions about how a real system behaves, the entities in a discrete-event model correspond to objects found in the real system. For this reason discrete-event simulation is potentially easier for model-builders with limited simulation experience to understand.

The benefits of employing discrete-event simulation techniques for evaluating manufacturing system performance are already well established (van der Walde, 1991; Robinson, 1993) although these models are generally used to investigate only shop-floor aspects of these systems such as material flows, inventory levels and resource utilisation (Blundell, 1994). In contrast, the application of discrete-event simulation to evaluate the performance of support functions in manufacturing organisations is less well established. Davies (1994), for example, comments generally on the surprisingly

few examples of where the technique has been used to model office processes given that other methods, even dynamic mathematical models, cannot represent the complexities of inter-dependency and external delays that are characteristic of office processes. Davies successfully applies discrete-event simulation to evaluate administrative processes in a financial services firm. In a similar case study, Ketcham (1991) also uses the technique to model office procedures in a financial services company. Ketcham justifies the use of simulation on the basis that the impact of administrative bottlenecks can only be assessed properly using a technique that models at this level of detail.

These different examples demonstrate that discrete-event simulation is generally considered to be the most appropriate technique for modelling both manufacturing and office activities. Ironically, however, there do not appear to be any reported attempts to combine manufacturing and support activities within the same discrete-event model. The remainder of this chapter will therefore evaluate some different types of discrete-event simulation tool as well as some specific simulation applications in terms of their suitability for building such a model for use in the manufacturing system design process.

6.2 The Case for Using Data-Driven Simulators

Simulation models can be constructed in a variety of different ways. Models can be created using anything from general-purpose programming languages to visually-interactive modelling systems (Pidd, 1992b). Choosing how a model will be created is essentially a trade-off between ease-of-use and flexibility.

Ease-of-use in this context relates to how quickly a model can be set up as well as the level of skill required in terms of programming and simulation expertise. Flexibility relates to how adaptable any approach is in terms of being able to model a range of different types of problem. Figure 6-1 illustrates these trade-offs by way of four different categories that cover the range of methods available for building simulation models; general-purpose programming languages, simulation languages, generic simulators and domain-specific simulators.

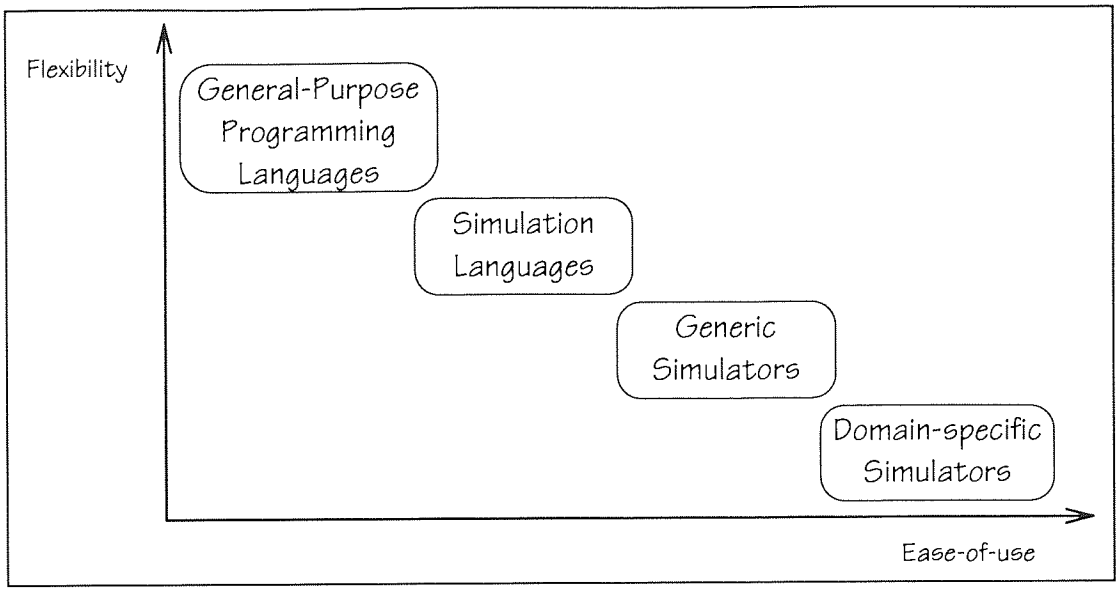


Figure 6-1 Trade-off between Ease-of-use and Flexibility in Approaches to Constructing Simulation Models

Since *general-purpose programming languages* such as C, C++ and Pascal are intended for developing practically any type of software they do not contain any simulation-related routines or constructs. Consequently they impose virtually no constraints on how a simulation model should be constructed. For instance, the model creator will be able to define even the most fundamental aspects of the model such a time-advance mechanisms, random-number generators, results generators and so on. In addition, the model-builder will have complete responsibility for deciding how elements of the real system should be represented in the model. For this reason, creating simulation models essentially from scratch using general-purpose programming languages provides almost unlimited flexibility in terms of the type of model that could be created. Theoretically, at least, one could therefore use this approach to build the type of model that is proposed for evaluating manufacturing system design decisions.

In practice, however, the feasibility of constructing a model of the scope and level of detail required for testing manufacturing system designs using a general-purpose programming language is questionable. Even if a design team could call upon the necessary programming and simulation skills for this type of project, building an adequate model is likely to take a considerable amount of time. McConnell (1993), for example, claims that in terms of commercial software development the industry-average productivity rate for programmers is only about 8 to 20 lines of code per

person per day. He explains by emphasising that although physically typing this amount of code is only likely to take a few minutes, a very significant proportion (McConnell suggests up to 50 per cent) of a programmers time is spent debugging code. Clearly at a rate of 8 to 20 lines per man day a simulation model will take a very long time to create unless only trivial models are built.

Simulation languages such as SIMULA and SIMSCRIPT evolved to reduce the amount of time and effort required to construct a model as compared to more general-purpose languages (Banks et al., 1991). The syntax of a simulation language also allows a model-builder to express the behaviour of a real system more naturally in the model of that system than would be possible using a general-purpose language (Seila, 1995).

Simulation languages do place some minor constraints on how models can be constructed by pre-defining some basic modelling functionality such as time-advance mechanisms and random-number generators. Nevertheless, these languages are still extremely flexible in terms of the scope of model that can be created. For example, Wood (1993), Hoefer et al. (1994) and Levine (1994) present models of administrative systems developed using simulation languages while Saunders & Novels (1994) have applied the technique to build models of a manufacturing system.

In terms of speeding up the model-building process, Pidd (1992a) suggests that a single line of code written in one of these languages can represent the equivalent of dozens of lines written in a more general-purpose language. Despite this level of improvement, however, simulation languages still place great demands on decision-makers in terms of the time and level of skill required to construct models. This is significant given that the previous chapter emphasised that if a modelling tool is to be used in practice to evaluate manufacturing system design alternatives then it should allow people with only a limited knowledge of modelling techniques to build models quickly.

Wood (1993), for example, claims that his model took a simulation consultant ten days to build using SIMAN/CINEMA yet the scope of his model extends to representing only one department within a financial services firm. If it takes a simulation expert this length of time to create a model of just one aspect of an organisation, then it would

certainly not be feasible for even a team of manufacturing system engineers to construct a model of a manufacturing business that includes many different departments.

A more realistic approach would be to use a simulation tool that relieved decision-makers of the programming aspects of model-building altogether. This would greatly improve the ease and speed of the model development process and provide a more practical way for decisions-makers to construct large simulation models. To this end, *simulators* have emerged as an alternative to both general-purpose and simulation languages. Simulators enable a parameter or data-driven approach to modelling. In other words, no programming is required to construct models; the user just configures a model from pre-defined logic elements that will already have been programmed in a lower level language by a simulation expert (Banks et al., 1991). A data-driven approach also implies that model-builders will be re-using tested code and verified model logic. This means that not only will the inevitably long debugging process be avoided, but model builders can also be confident that the model logic is more likely to properly describe the behaviour of entities in a real system.

Invariably, however, these benefits come at a price. For instance, the logic embedded in a simulator is unlikely to be able to represent characteristics of every type of system that one might wish to model. For this reason, a data-driven simulator will be less flexible in terms of the range of different types of model that can be created. This is especially true of *domain-specific simulators* which, as the name suggests, are designed to model a very limited range of problems such as those relating to manufacturing systems or transport systems. This is potentially significant in terms of identifying a data-driven simulator that will model both the material processing aspects of a manufacturing system as well as the document processing aspects of its support functions.

While *generic simulators* can model a broader range of systems, they tend to demand more on behalf of the model-builder in terms of having to abstract features of the environment being modelled into generic simulation terms and concepts. Ideally, therefore, manufacturing system engineers require a domain-specific simulator where that domain extends to simulating both the material and information processing aspects

of a manufacturing business. The remainder of this chapter will review specific data-driven simulation packages, both generic and domain-specific, with a view to modelling these different aspects of a manufacturing business.

6.3 The Disadvantages of Manufacturing Simulators

By implication, a manufacturing simulator will be domain-specific since it is intended to model a particular type of environment. In fact, the majority of domain-specific simulators have been developed for manufacturing applications (Law & Haider, 1989). For example, simulators like *ATOMS* (Bridge, 1990), *SIMFACTORY* (CACI, 1992) and *ProModel* (Harrell & Tumay, 1991) allow decision-makers to build models using concepts and terminology that relates to manufacturing such as machines, conveyors, operators, parts, routings and so on. As a consequence, however, the scope of these simulators also tends to be limited to modelling only material-flows and other shop-floor activities. This is presumably because the performance of manufacturing facilities has traditionally been evaluated in isolation. Nevertheless, it implies that the logic embedded in data-driven manufacturing simulators is not designed to model document or information processing activities.

Some manufacturing simulators do incorporate production 'information' into models but rather than representing document flows, this is limited to either production control data (Boughton, 1995) or machine control data for Computer Integrated Manufacturing applications (Young et al, 1988). *WITNESS*, on the other hand, is described as a more general-purpose simulator (Pidd, 1992b; Thompson, 1994) as well as a manufacturing simulator (Banks, 1995; Hlupic & Paul, 1995). This is because it allows models to be created using either manufacturing or more generic terminology. One might expect that this type of simulator would therefore be more suited to modelling manufacturing and support activities. In fact, Dwyer & Korwin (1990) describe the application of *WITNESS* to simulate different parts of an organisation ranging from the factory-floor to a drawing-office. It should be noted, however, that these models were constructed as part of completely unrelated simulation projects. In other words no attempt was made to integrate these two models into a single model of the whole business. Nevertheless, this example is still very relevant because the way that Dwyer & Korwin were forced to construct their drawing-office model emphasises some potential

limitations imposed even by general-purpose simulators such as *WITNESS*.

The purpose of the drawing-office model was to help identify the optimum number of CAD stations required by a product design team but the authors emphasise basic limitations of *WITNESS's* generic model-building style:

> *"The situation [being modelled] was adapted to entities which the software package could understand."*

In other words, as a consequence of being developed for generic use, even some of the basic simulation logic embedded in *WITNESS* did not actually correspond very well to anything in the real system being modelled. In practice, using this type of simulator will place additional burdens on model-builders, in terms of skill and time for them to abstract aspects of the real system into concepts that can be included in the model. In this case, for example, the authors were forced to treat the drawing-office as if it were a manufacturing facility:

> *"The design jobs were modelled as parts to be processed, the CAD stations were modelled as machines required to process the parts, and the engineers are modelled as the available labour to handle and run the machines."*

This also implies that if the model had been integrated with a shop-floor model then processing of parts and of documents would be represented in exactly the same way. The limitations of representing manufacturing and support activities this way will be discussed later.

As an alternative to persisting with inappropriate model logic, some manufacturing simulators allow users to extend the functionality and scope of the package by writing code or defining new logic constructs. *Taylor II*, for example, was developed for modelling manufacturing and logistics systems (Nordgren, 1994). With this system models are created from four basic building blocks; *elements*, *jobs*, *products* and *routings*. However, for all but the simplest of applications Taylor II requires that the user defines model behaviour using a macro-language, the *Taylor Language Interface*. For example the statement,

```
select 1 from 3, 4, 5 - utilisation[L]
```

defines that one product will be sent to either machine 3, 4, or 5 with priority being given to the machine with the lowest utilisation (Nordgren, 1994). While this syntax is intended to be easier to use than more general-purpose language constructs, it is still well removed from the kind of terminology that will be familiar to model-builders without simulation experience. More importantly, if programming is required simply to model manufacturing activities, then it is likely that a very significant amount of additional code will be necessary to model support functions as well.

In summary, this section has reviewed some specific data-driven simulation packages that are currently available. In particular, the review has assessed those packages that tend to be referred to most frequently in simulation and manufacturing system design literature, and are therefore potentially the most well known and well researched. As such, the study is not a complete review of every commercially available simulator; Elder (1995a), for example, suggests that there are around two dozen discrete-event simulation packages currently on the market. Nevertheless, it can be assumed that the systems that have been reviewed are at least representative of the best systems currently available and the study therefore provides an indication of the kind of model one might expect to be able to build using the latest simulation technology.

On this basis it can be concluded that none of the existing range of commercial simulators aimed at manufacturing system modelling can properly model support functions. This is perhaps understandable given that the traditional manufacturing system design process tends to ignore the impact of support areas on the performance of a manufacturing system. The remainder of this chapter will review other types of data-driven simulator, including some that have been developed specifically to model document processing activities, with a view to identifying an appropriate technique for evaluating manufacturing decisions.

6.4 The Limitations of Other Simulation Tools

6.4.1 Business Process Analysis

Business Process Re-engineering (BPR) is a concept that has gained a large amount of publicity over recent years, although different interpretations have been put forward as to what the concept actually involves (Patching, 1994). For example, Davenport &

Short (1990) describe BPR as:

> *"The analysis and design of work flows and processes within and between organisations."*

Davenport & Short also suggest that their interpretation of BPR could be combined with developments in information technology to create a 'new type of industrial engineering'. In other words, they view BPR as an analytical activity intended to improve overall business productivity in the same way as industrial engineering has traditionally been applied to improve manufacturing performance. This suggests that techniques such as simulation which are used to evaluate shop-floor performance should also apply to BPR. In addition, given that the redesign of support functions is invariably the main focus of attention within BPR projects (Hall et al. 1993; Hall & Stewart, 1994; Peppard & Rowland, 1995) the analysis tools used for this type of exercise are particularly relevant to this thesis.

6.4.1.1 Business Process Analysis Tools

A variety of different modelling tools are available for analysing business processes, ranging from flow-diagramming tools based on techniques such as IDEF, to simulation modelling tools (Spurr et al., 1994, Gladwin & Tumay, 1994). Nevertheless, using a data-driven simulation approach to evaluate business processes is justifiable on the same basis that simulation is also the most appropriate method for evaluating manufacturing system designs. In other words, simulation provides the necessary dynamic analysis (Tumay, 1995) and the model-building process can also foster a better understanding of the problem domain (Bridgeland & Becker, 1994).

However, given the broad scope of the business process re-engineering concept, modelling tools developed for this type of application are typically less domain-specific than their counterparts used for evaluating manufacturing systems. This means that while BPR modelling tools are better than manufacturing simulators in terms of flexibility, they are not likely to be able to model specific aspects of a business quite as well as a simulator designed especially with that aspect in mind.

SIMPROCESS III (Jones, 1995) is a case in point and represents a typical data-driven business process analysis tool. Models are constructed from three basic types of

building-block; *resources*, *tokens* and *processes*. Resources are used to represent things like people or machines while tokens represent objects that flow through the business. The notion of 'tokens' is intended to cover anything from physical objects such as parts to information-type objects such as documents (Jones, 1995). Therefore, in terms of a model of a manufacturing system and its support functions, a modelling tool such as *SIMPROCESS III* would essentially treat parts and documents in the same way.

Some conceptual differences, in modelling terms, between shop-floor activities and administrative activities will be demonstrated later in the thesis. This will include fundamental differences in the way documents and parts are processed. In addition, it will be demonstrated that the information that a support department processes actually affects actions taken on the factory-floor. This means that in order to properly evaluate how support departments will affect manufacturing system performance, the context of the information contained on each document should be included in the model.

These important characteristics imply that document processing cannot be modelled simply as series of stages whereby a resource takes a document from a queue, delays while processing the document, and then places new or amended documents into another queue. In other words, a modelling system that treats parts and documents in the same way, and in terms of 'tokens', will not properly capture the characteristics of support functions that will influence manufacturing system performance. While attributes such as 'cost' can be associated with tokens in a *SIMPROCESS III* model, this is not the same as modelling the type or context of production data that will be carried on these documents.

In terms of their applicability to the manufacturing system design process, the same criticism applies to most other business process simulation tools. For example, static process models based on the IDEF methodology can be imported into data-driven simulation tools such as *ServiceModel* (Shapiro 1994; Gladwin & Tumay 1994) and *WITNESS* (Lingineni et al. 1995; Thompson 1995). While this provides dynamic analysis the model is limited in detail in the sense that parts and documents are again treated in the same way, in terms of tokens.

6.4.1.2 Workflow Applications

Workflow technology relates to a type of software application that automates the movements of documents between staff within an organisation and can also be used to automate routine or administrative tasks (Norfolk, 1995). Workflow is synonymous with the principles of business-process re-engineering since information technology often forms an integral part of any re-designed process (Peppard & Rowland, 1995).

Since workflow essentially mimics activities that are normally performed by people, it follows that this technology does in the real business what a simulation model of that same business will also try to do. For this reason, while workflow itself is not another type of simulation technique, modelling and analysis capabilities are often integrated into workflow applications. For example, Miller et al. (1995), Norfolk (1995) and Howlett (1996) present different workflow packages that incorporate simulation features for analysing business processes.

Given that some workflow systems can actually process documents as well as simply moving them from one part of an organisation to another, the context of the information contained on each document must form an important aspect of this type of system. In other words, the fact that a purchase order might instruct a buyer to place a demand for *'250'* of *'PartXYZ'* with a given supplier should actually mean something to the workflow system. It follows that the simulation capabilities of a workflow package should be potentially better than the business process modelling tools described in the previous section which tend to model documents simply as 'tokens'. For example, a workflow system could potentially simulate the interactions, and by implication the delays, associated with just performing a single operation on a document in a way that would not be possible with other business process analysis tools.

However, despite being potentially able to model support functions, workflow systems are not able to model material processing activities to the same extent as a conventional factory simulator. Consequently a workflow application alone could not be used to predict how support departments will affect manufacturing performance. For this reason, workflow systems are not appropriate for evaluating manufacturing system design decisions. In fact, overall, while BPR or workflow simulation tools can

model a greater range of business activities than conventional manufacturing simulators, they are limited in many aspects of the detail that can be included in a model. Some BPR tools can include manufacturing and non-manufacturing elements in the same model, but the modelling of interactions between these elements is too aggregate to evaluate the effects of manufacturing decisions properly.

6.4.2 Hybrid Modelling Systems

The discussion has so far identified different types of simulator that can model material processing or document processing activities in isolation; none of the systems reviewed can model both sets of activities within a single model. One potential way of overcoming this major limitation would be to construct a hybrid model. In other words, two separate models could be created using different modelling systems to represent these different aspects of the business, and the models then combined together using some kind of bridging program. So, a manufacturing simulator might represent the firm's shop-floor activities while something like a workflow system would simulate its support functions. The bridging software would attempt to represent the interactions between entities in each model.

In practice, however, it is unlikely that two collaborating simulation models would be able to represent entity interactions properly. In particular, the timing of information being received by the shop-floor contributes to the complex dynamics of the real manufacturing system and so must be accurately represented in the model. A hybrid model is unlikely to be able to update the different models simultaneously and so the impact of timing will be distorted. This is evident from hybrid systems such as cost models that have been linked to manufacturing simulators but which actually generate costs after the factory simulation has been completed (Christy & Kleindorfer, 1990; Wu, 1990; Burhanuddin & Randhawa, 1992).

A hybrid model also implies that real-word entities will be constrained to one or other of the two models; either the factory or the support department model. This means that some important functions cannot be modelled properly. Material handling, for example, involves moving of materials as well as processing of goods inwards documentation or stores requisitions. Material handling personnel would therefore need to be represented in both models simultaneously.

A hybrid model also means that end-users would be confronted with three different modelling applications; a manufacturing simulator, a support department simulator and another application that attempts to link these both together. This is likely to make the model-building process significantly more complex than if a single modelling system were used. Overall then, it can be concluded that a hybrid model is not appropriate for the manufacturing system design process. Engineers requires a single data-driven simulation system that can combine both shop-floor and support activities within the same model.

6.5 Summary

This chapter has established discrete-event simulation as being the most appropriate technique for evaluating manufacturing system design decisions. This approach offers detailed modelling of dynamic behaviour and uncertainty, and also provides a closer match between entities in the model and objects in the real world. However, discrete-event simulation is also potentially demanding in terms of the time and level of simulation expertise required to build even simple models. This means that, in practice, engineers will need a completely data-driven simulator to build models of the size and scope proposed for the manufacturing system design process. This will greatly ease the model-building process but, potentially, at the cost of flexibility.

A range of commercial simulators have therefore been reviewed in terms of their ability to easily represent manufacturing and support functions in the same model. It was concluded that these applications either lack the scope to model support functions at all or do so in such a simplistic way that the interactions between these areas and a manufacturing system cannot be modelled properly. Consequently, a new type of domain-specific simulator is required if decision-makers are to be able to properly test their manufacturing system designs prior to implementation. This new simulator would have the ability to model shop-floor areas in the same level of detail that manufacturing simulators already model these activities, but it would also be able to model an equivalent amount of detail in relation to support departments, and incorporate both aspects into the same model. The next chapter proposes the development of a new simulator to satisfy these requirements.

7. The Potential of Object-Oriented Techniques

The previous chapter demonstrated the advantages of using data-driven simulation for evaluating the impact of support departments on the performance of a manufacturing facility. However, a review of the data-driven simulation tools that are presently available also revealed that none can properly model the interactions between these different parts of a manufacturing business. As a result, a new kind of simulator is required to perform this type of analysis. This chapter and those that follow will therefore describe the design and development of a new simulation system that fulfils these requirements. In particular, this chapter will demonstrate that object-oriented techniques are the most appropriate for developing a software application with the necessary scope to model this type of problem. The discussion will begin with an overview of more general software development issues.

7.1 Managing Software Complexity

The problems that software applications have been developed to address are inherently complex, such as the management of a firm's financial information or the preparation of production schedules and supplier orders. In addition, successive generations of most types of software seem to evolve to be even more complex in terms of the range of functionality they offer. For example, modern word processors are clearly very advanced in terms of the range of functionality they offer as compared to early text editors. A data-driven simulator that will be able to model both the shop-floor and support functions in a manufacturing organisation is another example of a more complex software application that is expected to provide an even greater range of functionality than ever before.

The role of software design is to bring a degree of order to this complexity so that the task of developing and maintaining an application is simplified. Overcoming complexity in this way generally means reducing a large complex system in smaller, more manageable parts (Booch, 1994). Software design has traditionally employed *structured design methods* in order to make software development more manageable. Structured design implies a top-down decomposition of complex systems into more orderly sub-systems (McConnell, 1993). This approach is also referred to as functional

decomposition and involves reducing the solution to a complex problem into successfully smaller units of functionality to the point where each unit can be easily defined by relatively small segments of code (Taylor, 1995). Figure 7-1 illustrates how functional or top-down decomposition reduces a business problem to more basic software constructs.

Figure 7-1 Top-Down Decomposition of a Complex Business Problem (after Taylor, 1995)

By adopting a structured approach, software development is reduced to providing data structures and separate functions or algorithms for manipulating that data. However, this type of design also means that the data and algorithms become separated and dispersed throughout an application; consequently, different aspects of the system's functionality become embedded in every part of the software, with the result that modifying or updating the system is very difficult without restructuring the whole design (Taylor, 1995). While this may be acceptable for small applications, the overhead imposed by restructuring often prohibits the modification or extension of large software systems (Booch, 1994).

Object-oriented methods for constructing software applications have evolved in response to the limitations imposed by structured design methods. Object-oriented

development is based on the premise that the more closely an application reflects the problem it represents, the better the application will be (McConnell, 1993).

The approach involves a bottom-up composition of a software application rather than the top-down decomposition approach imposed by structured design methods. In other words, the complexity of a problem is reduced by describing the problem domain in terms of *objects*. An object can correspond to something that physically exists in the real-world, such as a person or type of document. Alternatively, an object might describe a more abstract concept such as a telephone conversation or a master record held in a database. A software application can then be created from a library of objects that represent each aspect of the business problem.

An important feature of the approach is that objects contain both data and the algorithms (referred to in object-oriented notation as *methods*) required to manipulate that data and describe the object's behaviour. In other words, the object-oriented approach ensures that each aspect of a system's functionality is incorporated into a completely self-contained unit rather than being dispersed throughout an application. This means that modification to object-oriented applications is simplified because individual objects can be changed with minimal knock-on effects elsewhere in the software.

Object-oriented techniques clearly represent a fundamental shift in the way complex software is constructed and maintained, which is very relevant from the point of view of developing a new simulation package. The next section will therefore present some other important aspects of the concept in more detail.

7.2 Object-Oriented Concepts

Object-oriented software development is still a relatively recent philosophy but the essential features of the approach are already well documented. Brief descriptions of the major concepts relating to object-oriented software construction will follow. The descriptions are drawn from definitions provided by several authors who also provide more comprehensive discussions of object-oriented principles (Rumbaugh et al., 1991; Booch, 1994; Graham, 1994).

The notion of *abstraction* relates to the type and level of detail captured in an object-oriented representation of a real-world entity. In an object-oriented capacity planning application, for instance, machines might simply be represented in terms of parts produced per hour and changeover times. In contrast, the same machines are likely to be modelled very differently in a computer-aided engineering application. In this case, objects might describe machine behaviour in terms of vibration, cutting forces and tool wear. These two abstractions of the same real-world entity are quite different but are, nonetheless, both valid representations given the different types of application to which they apply.

Encapsulation is closely related to the principle of abstraction. It refers to the idea of compartmentalising all data and algorithms required to describe some aspect of a problem into a single object. This means that access to that data by other objects can be more strictly controlled. As a result, software applications tend to be both easier to understand and to maintain since changes to a particular object should have minimal knock-on effects elsewhere. The concept also corresponds to how objects in the real world interact. For example, an 'accounts clerk' will only be able to access the data stored in an 'information system' via the controlled user-interface of a computer terminal.

The concept of *inheritance* allows similarities and differences between objects to be represented in terms of a *hierarchy*. Inheritance also relates closely to the notion of abstraction since it enables objects to be described at different levels of detail. For example, two quite different abstractions of a machine were described earlier; a capacity planning model and a computer-aided engineering model. A object-oriented hierarchy would allow both these representations to inherit a more general model of a shop-floor machine that encapsulated the data and functions that each of them shared. For instance, all shop-floor machines are likely to be referred to by a name and be located in a particular department.

Finally, while the term *object* has been used so far, the notion of a *class* is normally used in practice to represent real-world entities. Objects generally refer only to instances of a particular class. For example, the concept of a *department* defines a class of entity found in a manufacturing organisation. All departments will contain

resources in terms of people, computers, machines and so on. Instances of this class might include a *Purchasing Department* or a *Goods Inwards Function.* In other words, these objects represent instances of the class *department.*

This section has briefly introduced the main concepts of the object-oriented approach. Although object-oriented techniques represent a major shift generally in the way many types of software ought to be constructed, simulation is perhaps one of the most natural applications of this type of approach (Pavicic, 1991). In fact, modern object-oriented principles originate from the development of SIMULA, an object-based simulation language (Dahl & Nygaard, 1966). The next section will therefore discuss the advantages of adopting these same principles to develop a new simulator for modelling manufacturing systems and their support functions.

7.3 The Application of Object-Oriented Methods to Simulation

A simulation model is intended to represent some aspect of the real-world and people view the real-world in terms of objects (Adiga, 1989). A manufacturing system and its support functions, for example, can be described in terms of objects such as departments, people, machines, parts, documents and so on. Consequently, an object-based simulation model will provide a more natural mapping between objects in a real system and objects in a model of that system. This presents advantages to both developers and users of simulations software.

For developers in particular, object-orientation helps to bring a more natural sense of order to inherently complex problems. For example, the logic that describes the behaviour of real-world entities can be more easily encapsulated within corresponding simulation objects. In effect, a library of simulation objects can be created to represent each of the different types of entity that exist in the system being modelled. The interactions between these different types of entity can then be represented as interactions between objects in the simulation model.

The principle of inheritance also means that tried and tested logic can be re-used by several objects. In a manufacturing firm, for example, materials and documents are similar in the sense that they both progress intermittently through their respective parts

of the organisation, stopping temporarily at each stage to be processed. In simulation terms, this implies that a base object might describe the path of each of these entities through the business, while descendant objects for documents and parts respectively will differ in terms of how each processing stage is defined. This concept will be described in more detail later in the thesis.

Taken together, the concepts of encapsulation and inheritance mean that an object-oriented simulator will be potentially more robust and easier to enhance or modify than a system developed using a more traditional approach (Ball, 1994; Pidd, 1996). This is important given that data-driven simulators are notoriously limited in the range of real-world objects that they can model. A developer might be asked, for example, to extend the range of documents that can be simulated. Assuming that the object hierarchy has been well designed, the developer should be able to slot a new class into this hierarchy with minimal knock-on effects elsewhere in the software.

Clearly object-oriented concepts offer many benefits from the point of view of developing simulation software but end-users can also benefit without even being aware of the technique (Pidd, 1995). This is evident from the popularity of operating systems such as *Microsoft Windows 95*, many users of which will be totally unaware of its underlying object-oriented architecture. In relation to the manufacturing system design process, the model-building features of an object-oriented simulator should map more naturally to end-user's perception of the elements in the system they wish to model and the system should therefore be easier to use.

Given the wide ranging advantages and opportunities offered by object-oriented simulation, it is proposed to develop a new tool for modelling manufacturing systems and support functions as an object-oriented simulator. Booch (1994) suggests that, in practice, the process of developing software this way should be addressed in three stages; object-oriented analysis, object-oriented design and object-oriented programming. The next section will discuss how these stages might relate to the development of a new simulation tool.

7.4 Object-Oriented Analysis, Design and Programming

While the concepts of object-orientation offer the potential to create better software applications, success depends very much on sound planning as well as good code implementation. In other words, analysis of the problem and design of an appropriate solution is at least as important as actual programming. For this reason, the overall development cycle depicted in Figure 7-2 is typical of accepted good practice in terms of object-oriented software construction.

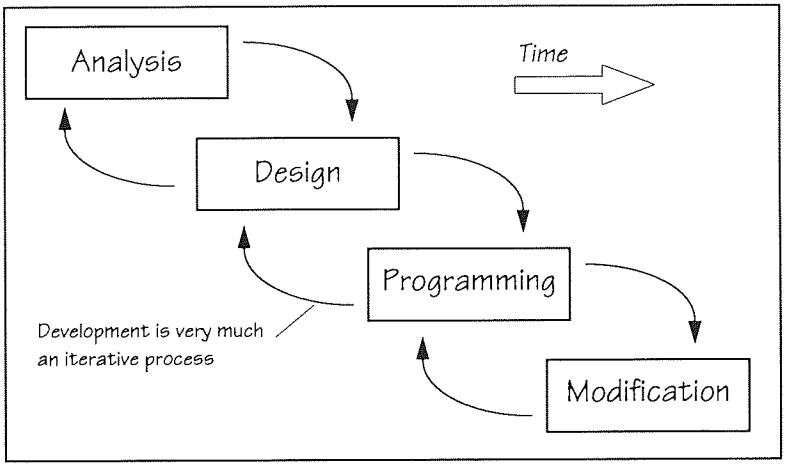


Figure 7-2 Object-Oriented Software Development Cycle (adapted from Booch, 1994; McConnell, 1993)

The illustration emphasises that while the main stages (analysis, design and programming) generally take place in sequence, the overall process invariably tends to be quite iterative. For example, coding problems encountered at the programming stage may force a review of some aspects of the software design. In practice this means that the boundaries between each development stage are sometimes unclear (Booch, 1994).

Object-oriented analysis (OOA) normally represents the first stage of the software development process and identifies the requirements of an application in terms of objects or classes found in the real world (Booch, 1994). The kinds of abstraction that are identified will vary depending on the problem domain, but will typically correspond to tangible items, interactions, events and so on that exist in the real-world. Relationships and collaborations between classes will also be identified. The main purpose of OOA is to provide a better understanding of the problem being addressed. Coding and implementation issues should not be considered at this stage (Rumbaugh et al., 1991). The output of OOA is an object model that describes the classes that should

be included in the application.

The *object-oriented design* (OOD) phase extends this object model to produce a schematic representation of how the various classes will later be implemented in a programming language. OOD differs fundamentally from more traditional structured design processes in that both the data relating to a real-world object and the logic that manipulates that data will be encapsulated within a class description. As a result, this stage also identifies how objects (instances of classes) will be created and destroyed within an application. Inheritance relationships between the different elements of the object model will also be defined as part of the OOD process in the form of a class hierarchy.

Object-oriented programming (OOP) involves actually implementing the design in a particular software development language. Ideally an object-oriented language should be used since the constructs of such a language will be very similar to the constructs of an object-oriented design (Rumbaugh et al., 1991). In other words, an object-oriented language will support concepts such as objects, classes and inheritance which should ease the process of translating the design into program code. An object-oriented language will also help to ensure that the design is implemented as intended.

Nevertheless, circumstances may dictate that some or all of an application be implemented in a non-object-oriented programming environment. Even so, in these circumstances the final application will still benefit from object-oriented analysis and object-oriented design (Rumbaugh et al., 1991). This is relevant in the context of the language used to implement the user-interface for the new simulator but will be discussed in more detail later in the thesis. Also important, however, is that the OOA and OOD development phases should be independent of whatever programming language is ultimately used to implement the software. This policy has been applied in the construction of the new simulator.

Of course, one of the proclaimed benefits of an object-oriented approach is the principle of re-use. In other words, classes that describe some aspects of the problem being addressed may have already been implemented by another developer in the form of a class library. Re-using these classes not only means that duplicated development

effort is avoided, but existing code can be incorporated into a new application with the assurance that the logic it describes has already been tested and verified. This means that the length of the debugging process, which McConnell (1993) suggests can account for up to 50% of development time, can be reduced by re-using existing code.

In simulation terms, a developer can only re-use a class-library if he or she shares the same perception of the system being modelled as the individual who constructed the class-library. Object-oriented simulation has already been applied to model the manufacturing activities of a firm in isolation (Mize & Pratt, 1991; Shewchuk & Chang, 1991; Ball et al., 1994) which means that an opportunity exists to re-use one of these class libraries to model that aspect of a manufacturing organisation. Development of the new simulator could then concentrate on the construction of new classes that will model support activities but which integrate with the existing manufacturing classes. The next section briefly introduces the features of one particular object-oriented simulator that fulfilled the author's requirements for modelling manufacturing activities but was also capable of being extended to model the support functions in a manufacturing business.

7.5 The Advanced Factory Simulator and WBS/Control

Research at Aston University has already exploited object-oriented techniques to produce a manufacturing simulator, known as the Advanced Factory Simulator (Ball, 1994). This application enables the physical aspects of a manufacturing system to be modelled. For example, the system allows decision-makers to describe their manufacturing system in terms of machines, machine-operators, part-routings, material handling devices and so on. This also means that the model-building features of the Advanced Factory Simulator (AFS) relate closely to an end-user's perception of a manufacturing facility.

AFS is also completely data-driven and so model-builders are relieved of the need to program or describe any aspect of a manufacturing system's behaviour in terms of logic statements. This means that AFS can be used by people who are not simulation experts to build models quickly. Invariably, however, this also means that what logic is embedded within AFS is unlikely to be able to represent characteristics of every type

of manufacturing system that one might wish to model. For this reason, AFS has been implemented as an *extendible* simulator in the sense that its object-oriented class architecture has been designed specifically to allow developers to more easily enhance the functionality of the system by adding new simulation classes.

It should be noted that the concept of 'extending' the simulator's functionality refers explicitly to the idea of a simulation software developer creating new classes that encapsulate the behaviour of any additional feature to be included in the model. In other words, it is not intended that end-users of the system, such as manufacturing system engineers, should be able to extend the scope of the tool as is required by some commercial simulators. This ensures that only tested and verified logic is included in the simulator.

Boughton (1995), for example, has extended the AFS architecture to create WBS/Control which enables alternative manufacturing planning and control systems to be modelled. WBS/Control objects can be combined with other AFS objects to produce a single model of a manufacturing system that includes its planning and control system. This allows a more comprehensive prediction of manufacturing system performance than would otherwise be possible if the physical aspects of the system were modelled in isolation. WBS/Control is also completely data-driven and so continues the theme of an easy-to-use simulation tool.

However, both AFS and WBS/Control are very limited in the extent to which they can simulate document or information processing delays. Certainly, none of the classes in either of these libraries could be used to predict the impact of these delays on the performance of a manufacturing system to anywhere near the extent proposed within this thesis. Similarly, AFS cannot model any of the more general delays generated by support functions such as a purchasing department or goods inwards function. Nevertheless, AFS's object-oriented architecture has been designed specifically with the addition of new functionality in mind, and can already model the physical aspects of a manufacturing system and its planning and control system. Consequently, this library of simulation classes represents a particularly suitable base for implementing the new simulator since development can concentrate on the construction of new classes that will model support activities but which integrate with the existing

manufacturing and production control classes.

Another factor that influenced the decision to incorporate existing AFS and WBS/Control classes into the new simulator was access to the source code for both systems as well as the opportunity for regular contact with their respective authors. This was likely to be particularly useful in gaining a better understanding of the overall structure of each system. In addition, both AFS and WBS/Control were essentially prototype software applications during the period that the new simulator was being developed. In practice, this was likely to mean that some minor modifications to existing classes might be necessary to fully exploit the potential of the object-oriented architecture.

The remainder of the thesis will describe the design and implementation of this new simulator. Specific features of AFS and WBC/Control will be introduced as necessary during this discussion.

7.6 Summary

This chapter has introduced the concepts of object-oriented software which is potentially an easier and safer way to develop any large software application but is particularly appropriate for developing simulation software. A simulator based on these principles will also be easier for manufacturing system engineers to understand since people naturally tend to view the real-world in terms of objects.

To create an object-oriented simulation application properly will involve three stages of development; analysis of the problem domain, design of the classes and class hierarchy and finally implementation of the design in a high-level programming language. The next part of the thesis will describe the actual development of a new data-driven simulator that will be capable of modelling both a manufacturing facility and its support functions. The development will re-use and enhance the functionality of an existing set of simulation classes developed at Aston University known as the Advanced Factory Simulator and WBS/Control. Aspects of these current class libraries will be introduced as necessary during this discussion.

8. Object-Oriented Analysis I : Modelling Administrative Activities

This chapter presents the first part of an object-oriented analysis of the support functions typically found in manufacturing organisations with a view to constructing a novel set of simulation classes that will eventually integrate with the Advanced Factory Simulator and WBS/Control. This will provide a simulation tool that allows a more complete evaluation of manufacturing system design decisions. Importantly, however, the relationship of the proposed support department objects with these existing simulation classes is intentionally avoided at this stage since the analysis phase should not be influenced by coding issues (Rumbaugh et al., 1991). The study will be limited to providing a more general understanding of how support department objects interact with shop-floor objects in the real-world. This part of the analysis will provide a general overview of support department activities, and will also include a more specific object-oriented analysis of the administrative aspects of these activities.

8.1 The Nature of Support Department Work

The term 'support function' was defined earlier as representing those functions within a firm that perform activities which do not involve the physical transformation of materials or sub-components into finished goods. In effect, this means that the activities performed by support functions may include a variety of different tasks that are carried out away from the main manufacturing facility; this will involve administrative as well as more physical types of work that cannot be attributed to specific products.

This section will establish the most important features of administrative-type work as well as the other kinds of support activity that a simulator should be able to model in order to evaluate manufacturing system design decisions. The discussion begins with an overview of administrative and office tasks.

8.1.1 Administrative-Type Work

For many years work measurement techniques have been an indispensable tool for managing shop-floor activities and these methods are well documented in production

and operations management texts (see, for example, Wild, 1995). Variations on these techniques, such a Predetermined Motion Time Systems (PMTS), have also been tailored specifically for measuring clerical work. For instance, Karger & Hancock (1982) describe MTM-C, a Methods Time Measurement (MTM) system designed for setting clerical work standards. Nevertheless, the application of work measurement in office areas is considerably less well established in comparison to applications in shop-floor environments.

One reason for this is the comparatively low visibility of office work (Revell, 1986). Large quantities of work-in-progress on a factory floor have a very obvious presence whereas equivalent quantities of uncompleted paperwork tend to be less conspicuous. As a result, less attention is generally paid to monitoring the work content of office tasks or 'paper-work inventories' as compared to the effort devoted to reducing work-in-progress and increasing productivity levels in manufacturing areas. This happens despite the potentially adverse effects of tardy support department performance on shop-floor activities described in preceding chapters.

Revell (1986) argues that this tendency not to apply work measurement methods in office areas arises from resistance by office personnel to having their jobs assessed in this way rather than from any significant limitations in the measurement techniques that are available. He identifies several mistaken but commonly held beliefs about office tasks to explain this resistance. For example, Revell acknowledges that creative or 'brain' work is considerably more difficult to measure than repetitive work since the time taken to perform apparently similar creative tasks varies greatly. He also argues, however, that the amount of creative work involved in performing office tasks as compared to shop-floor tasks is often exaggerated with the result that office work is often perceived as being too difficult or impractical to measure. In fact, Revell suggests that the majority of office work is actually quite repetitive and relatively easy to measure, which also means that these types of activity are suited to quantitative analysis using simulation.

Revell's analysis concurs with a previous study conducted by Hamill & Steele (1973) in order to identify different categories of office work. They identify three basic types of activity that occur in these areas: *routine work*, *inherently variable work* and

creative work. Hamill & Steel also indicate the relative proportions of each of these types as a percentage of the total work carried out in office environments. A summary is provided in Table 8-1.

Type of Work	Description	per cent of total
Routine	This relates to the most basic and repetitive office work that is often carried out at lower levels of an organisation. In work measurement terms, routine work is analogous to shop-floor production work and is easy to measure.	30
Inherently Variable	This category relates to those office tasks for which completion times are difficult to predict because interaction with other departments or companies is required. These interactions introduce a degree of uncertainty in terms of how long the person carrying out a task must wait for all the necessary pieces of information to be available.	60
Creative	Describes tasks where the element of thinking or originating information becomes significant. Creative work is typical of the tasks performed by managers, designers and drawing office staff.	10

Table 8-1 Categories of Office Work

Clearly information technology has had a very significant impact on the way office work is now performed since Hamill & Steele conducted their analysis. Nevertheless, advances in information technology have tended mainly to automate existing administrative procedures rather than change the fundamental nature of office work in terms of the different types of task that are carried out (Hammer, 1990; Davenport & Short, 1990; Peppard & Rowland, 1995). For this reason, the categories of office task proposed by Hamill & Steele can still be assumed to be representative of the type of activities performed today and thereby serve as a useful basis for describing office systems in terms of objects.

For instance, the majority of tasks (60 per cent) appear to be *inherently variable* in the sense that fixed cycle times are often difficult to associate with some activities since they generally cascade into a set of further tasks performed by third parties. In other words, the person or persons responsible for the completion of an activity might

depend on another individual, a department or even another company having supplied a crucial piece of information in order that a job can be finished. As a result, the overall cycle time for a single office task might be subject to many different sources of variability or uncertainty. This contributes to the dynamic behaviour of administrative functions. In modelling terms this suggests that objects should be identified that represent the different elements and tasks involved in these interactions. This would include both physical and information related aspects which, when configured together in a simulation model, will be able to replicate the subtleties and delays associated with different types of interaction.

In some ways the notion of an office task that involves the collation of different items of information is similar to the concept of a shop-floor assembly operation. Using this analogy, the interactions associated with inherently variable activities correspond to the ordering of components from suppliers, except that in an office environment, the 'components' relate to items of information rather than physical parts. Uncertainty, in the form of supplier lead times is therefore a factor in both manufacturing and support activities. This analogy to assembly work will be developed further in a later part of the chapter.

Another significant proportion of office work (30 per cent) is more repetitive or *routine* in the sense that it involves shorter and more deterministic cycle times since interactions with third parties are avoided. As a result, this type of work is even more analogous to shop-floor activities implying that some office work should be modelled in the same way as simulators currently model production processes. Fortunately creative work, which is the most difficult of office activities to measure, and consequently the most difficult to model properly, actually represents only a small proportion (10 percent) of office work. Nevertheless, modelling delays associated with this type of activity also need to be addressed in a support department simulator.

8.1.2 Other Types of Indirect Work

Clearly some support department activities will involve more than just office-type work. Functions such as goods inwards or stores, for example, will involve a combination of administrative work and material handling. Similarly, a maintenance department will respond to requests to fix shop-floor machines. While, strictly

speaking, these represent indirect tasks, their obvious closeness to more direct activities means that some aspects of material handling or plant maintenance are already considered in the manufacturing system design process. In modelling terms this means that these types of support activity are effectively a combination of manufacturing and administrative functions. Consequently, these activities can be represented in a new simulator by combining existing simulation classes intended for modelling the physical aspects of a manufacturing system with new classes that describe office-type work.

In conclusion, this general overview has identified the most important features of office work and other support activities that a simulator ought to model in order to evaluate manufacturing system design decisions. The next section will discuss how the characteristics of office work, in particular, can be in described in terms of objects.

8.2 An Object Model of Administrative Systems

8.2.1 Document-Related Classes

The previous section introduced the concepts of *inherently variable work* and *routine work* and suggested that paper-flow systems are analogous to shop-floor assembly processes, in that documents and supporting information arrive at 'work centres' to be 'assembled' into new documents. In other words, administrative 'operations' convert 'components' of information into finished documents.

Clearly a business will employ a range of different types of document, each with their own information and processing requirements. One approach to implementing this functionality using objects would be to define a range of classes that described the different types of document typically found in a manufacturing business. These classes would differ in the type of information that each document contained. For example, a document class such as *PurchaseOrder* might contain information like *part number*, *order quantity* and *due date*. Another document class such as *Invoice* might contain these same information types together with other data items like *balance due* and *settlement discount*. In other words, the principle of inheritance could be used to define a hierarchy of document classes.

A limitation of this approach, however, is that since manufacturing organisations invariably differ in the types of document and information that they process, it is unlikely that a range of document classes could be designed to match all those used in even a small sample of companies. For instance, the information contained on a purchase requisition or purchase order in one firm may be quite different in another firm; some organisations may even combine both these documents into a single form.

This restriction is particularly significant given that the majority of office work is *inherently variable* and so most document processing times are affected by variable completion times for other tasks or responses to requests for information that relate to a document. This implies that the uncertain delays associated with obtaining these pieces of information should form a critical part of any support department model. It follows that model builders must be able to accurately represent the different types of data and document found in the business being studied in order to properly assess the impact of these various sources of uncertainty on manufacturing system performance. This would not be possible if the model builder was restricted to constructing models from a set of document classes provided by the developer of a simulation application.

As a result, the concept of a *DocumentTemplate* class enables simulation models to more closely represent the actual documents found in the company being studied. An object of this class will be created by the model-builder to represent each type of document that will be included in the simulation experiment; for example, *DocumentTemplate* objects might be created to represent a *GoodsReceivedNote*, an *Invoice*, a *Cheque* and so on. For each *DocumentTemplate* object, the model-builder will configure the types of data that will be contained on that particular style of document.

These document data types therefore represent the basic elements from which documents are constructed and will include items such as dates, quantities, prices, sign-offs, part numbers and so on. In object-oriented terms, the different types of information used by manufacturing organisations can be categorised as follows:

- *Date information*; nearly all business documents make some reference to dates or times, even if only to record when the document itself was

actually issued. Date information might also represent part of an instruction, such as a deadline for the completion of a particular task.

- *Names* identify entities in a business to which documents relate. For example, a stores requisition will refer to the part number of the item being requested and the name of the department or workcentre to where those items should be delivered.

- *Numeric data* conveys quantitative information and is closely related to the names of other entities that appear on documents. In the example described above, for instance, a stores requisition will also include a quantity to identify how many items of a particular part number are required. Financial information such as unit costs or sales revenue is another example of numeric data.

- *Authorisation information* controls the progress through an organisation of documents or of entities to which the document relates. For example, a purchase order will typically be 'signed-off' by a senior buyer prior to its release to a supplier. A purchase order will therefore carry authorisation details in addition to other types of information. In the same way, company policy may dictate that all bought-in items be inspected before release to the shop-floor. Goods received documentation would therefore need to carry information about whether or not the items to which they relate have been inspected.

Based on this analysis of document data types, a library of *DataObject* classes can be defined to represent the different types of information likely to be contained on documents within manufacturing organisations. Figure 8-1 depicts some typical data types from a library of *DataObject* classes.

The reason for identifying specific information types such as *QtyToOrder* and *PartPricePerUnit*, rather than a more general type such as *NumericValue*, is to embed some knowledge of what an item of information represents into the object model. This is necessary since the values contained on documents are intended to affect how the model progresses. Consequently other entities in the model, such as people, must be

able to interpret different types of document that will only be configured by a user at model-build time. By creating *DocumentTemplate* objects from specific data types, the developer can more easily embed the know-how to handle different kinds of information into classes that represent these other entities.

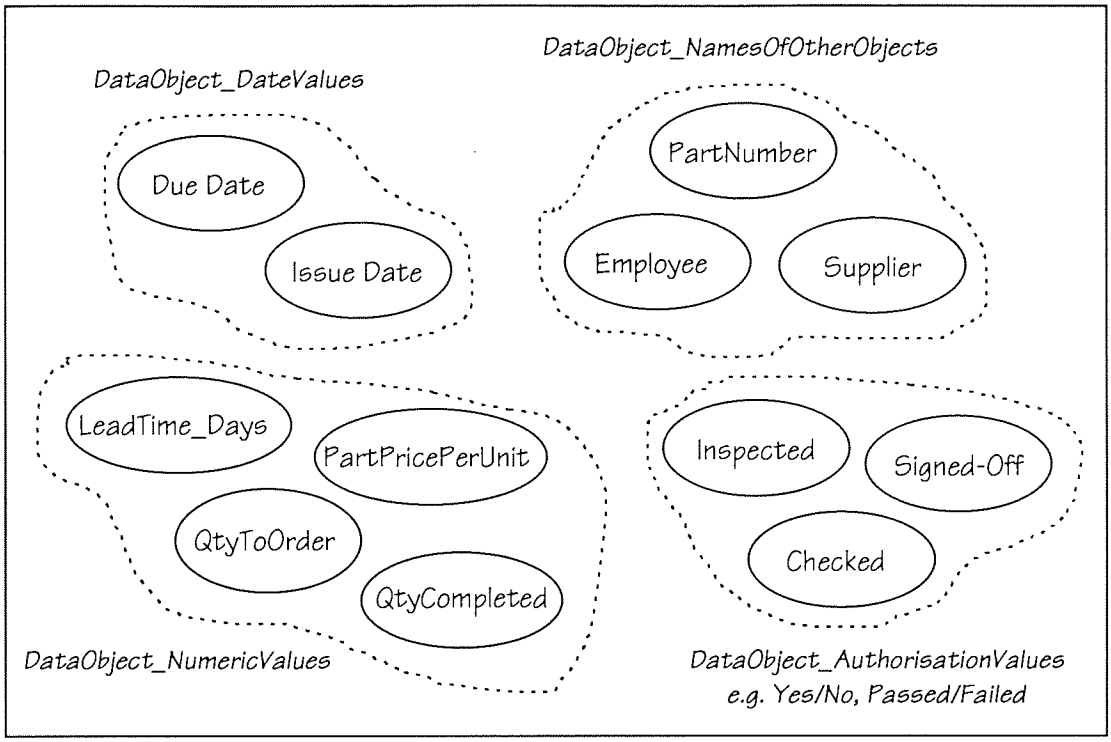


Figure 8-1 Sample Document Data Object Class Library

Clearly, the notion of a *DocumentTemplate* represents an abstract concept in the sense that it has no real-world equivalence. Nevertheless, it represents a convenient and very flexible way of enabling model-builders to more closely describe the types of information and documentation that get processed in a particular organisation. The term also relates closely to concepts such as documents and information that do have a real-world equivalence and should therefore be easy for model-builders to understand. A potential limitation of this approach, however, is that the range of *DataObject* classes identified by the developer will not be enough to represent every type of document found in a manufacturing business. Nevertheless, the impact of this limitation can be minimised by careful object design and will be discussed later in the thesis.

In addition to a *DocumentTemplate* class an object model must also include a *Document* class. Instances of this class will represent the actual or physical documents

that flow around the business. In other words, while a simulation model will contain one *DocumentTemplate* object for each style of document to be modelled, many instances of the *Document* class will be created and destroyed during a simulation as examples of each type of document.

The structure of a *Document* object, in terms of the types of information it will contain, will be defined by its respective *DocumentTemplate* object. In other words, one of the user-defined *DocumentTemplate* objects will be associated with each *Document* object and will specify which types of *DataObject* will be contained on that *Document*. Each of these *DataObjects* will correspond to a particular value or item of information that would be held on the document in a real business. Figure 8-2 illustrates the relationships between *DocumentTemplate*, *Document* and *DataObject* classes in a simulation model using a simple purchasing example.

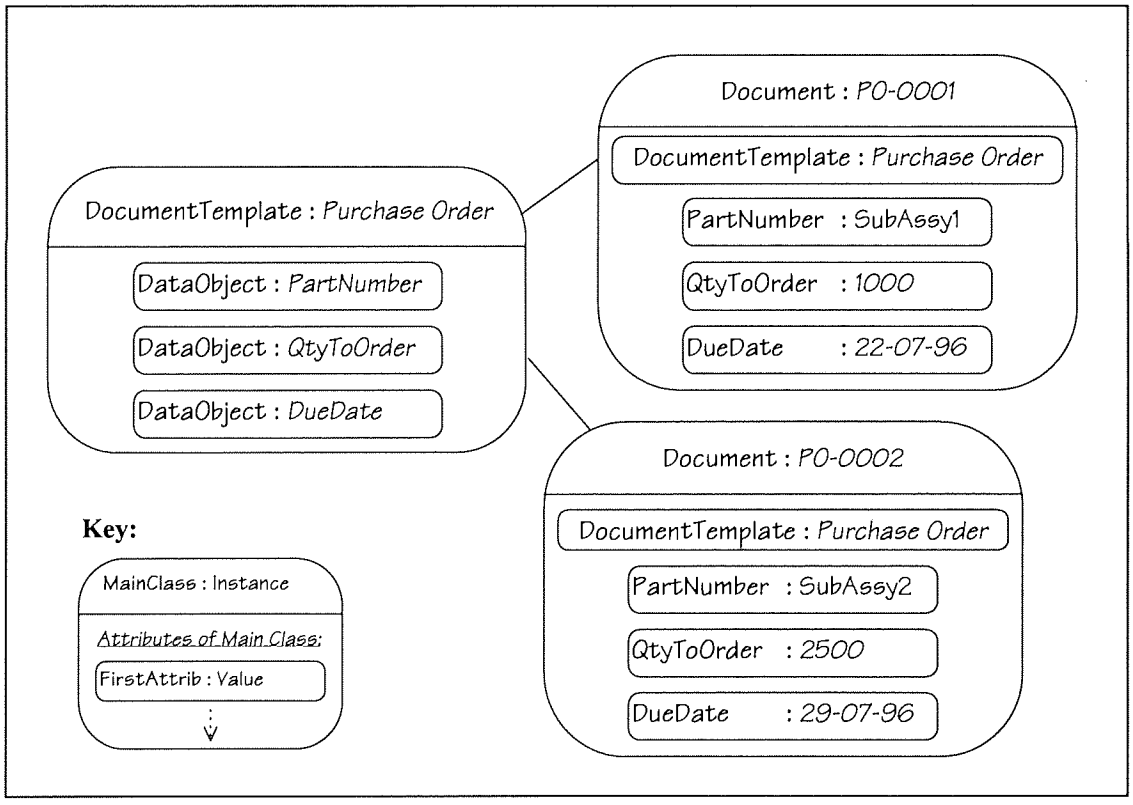


Figure 8-2 The Relationship Between Document Template, Document and Data Object Classes

The illustration shows an instance of the *DocumentTemplate* class named *PurchaseOrder*. In this case the model-builder has defined a purchase order to contain three types of information; *PartNumber*, *QtyToOrder* and *DueDate*. These represent the part number of the items to be supplied, the number of items required and the date

by which the items should be delivered. Two instances of the *Document* class are also shown in the diagram and referred to as *PO-0001* and *PO-0002*. These are dynamic objects in the sense that they are created as part of the simulation rather than by the model-builder, and represent the actual purchase orders that progress through the organisation.

Both of these *Document* objects carry the same types of *DataObject* or information as the *PurchaseOrder* template. For example, the purchase order referred to as *PO-0001* contains a *PartNumber*, *QtyToOrder* and *DueDate* as does its template. However, unlike the *PurchaseOrder* template, since *PO-0001* represents an actual document that passes through the firm, its *DataObjects* actually contain values that represent the information that would be carried by this document in the real business. In this case, for example, the values held by the *DataObjects* mean that *PO-0001* corresponds to a request for *1000* units of a part referred to as *SubAssy1*. These items are required by *22 July 1996.*

The other purchase order, *PO-0002*, holds the same types of information as *PO-0001* but the actual values are different. Consequently, the receivers of *PO-0001* and *PO-0002* will respond differently to each document, as would happen in the real business. In simulation terms this means that the speed at which *PO-0001* is processed and passed to a supplier may affect whether or not the *1000* units of *SubAssy1* can be delivered to the shop-floor by their due-date. In turn, this will affect how quickly these parts can be processed into finished goods and shipped to the customer. In other words, by modelling documents in this way, engineers can simulate the consequences of support department delays on manufacturing system performance in a way that would not be possible if documents were modelled simply as 'tokens', as the previous chapter illustrated was the case with existing commercial simulators.

In practice, some firms may use workflow systems or electronic-data-interchange (EDI) in administrative areas, and so certain types of document may exist in electronic form rather than as physical pieces of paper. In modelling terms, however, these two information mediums differ only in how a document moves from one part of the business to another. Consequently the *Document* class can represent documents that exist in both these formats while separate logic will describe exactly how these documents move through a business, and therefore represent the different kinds of

delay incurred in each case. This additional logic will be described in the second part of the analysis, in relation to modelling the transportation of documents.

8.2.2 Business Process Classes

Having identified a way of allowing model-builders to represent a wide range of business documents in terms of objects, a mechanism is required for describing how these documents, and the information they carry, will flow through an organisation and thereby support a firm's manufacturing activities. Again the analysis will attempt to describe this aspect of real manufacturing firms in terms of an object model. The section has been entitled 'business process classes' since the concept of a business process is often used to describe a set of related activities and/or information flows within a manufacturing firm. It should be noted, however, that this term is used more as a reference point for the discussion that follows and does not actually form part of the final object model.

8.2.2.1 A 'Paper-Factory' Analogy

Firms holding ISO9000 or BS5750 accreditation, or firms that have carried out any kind of business process re-engineering analysis, will usually formally record information flows in terms of procedures or business processes. Nevertheless, even companies that do not formally record office procedures will still associate certain styles of document with particular business activities. For instance, a simple make-to-order process will typically involve the creation of quotations, sales orders, purchase orders, works orders, inspection reports and despatch notes for finished goods.

In this example, each type of document will relate to a particular stage in the make-to-order process, in the same way as different shop-floor components and sub-assemblies relate to specific operations in a production routing. In this respect, office procedures are analogous to shop-floor routings and so support functions can be perceived as a kind of 'paper-factory' or an 'information-factory'.

In terms of objects, the concept of a *DocumentRouting* class therefore aptly describes a collection of individual document processing stages, or *DocumentOperations*, where new documents are prepared from information components. A process time attributed to each *DocumentOperation* will reflect the delay imposed by that activity. Both of

these classes are closely related to the *DocumentTemplate* and *Document* classes described earlier since particular types of document will be associated with specific processing stages. As a result, a *DocumentRouting* object can be viewed as an attribute of a *DocumentTemplate* or *Document* object since it defines the movements of each different style of document between processing stages. The relationships between these classes are depicted inFigure 8-3.

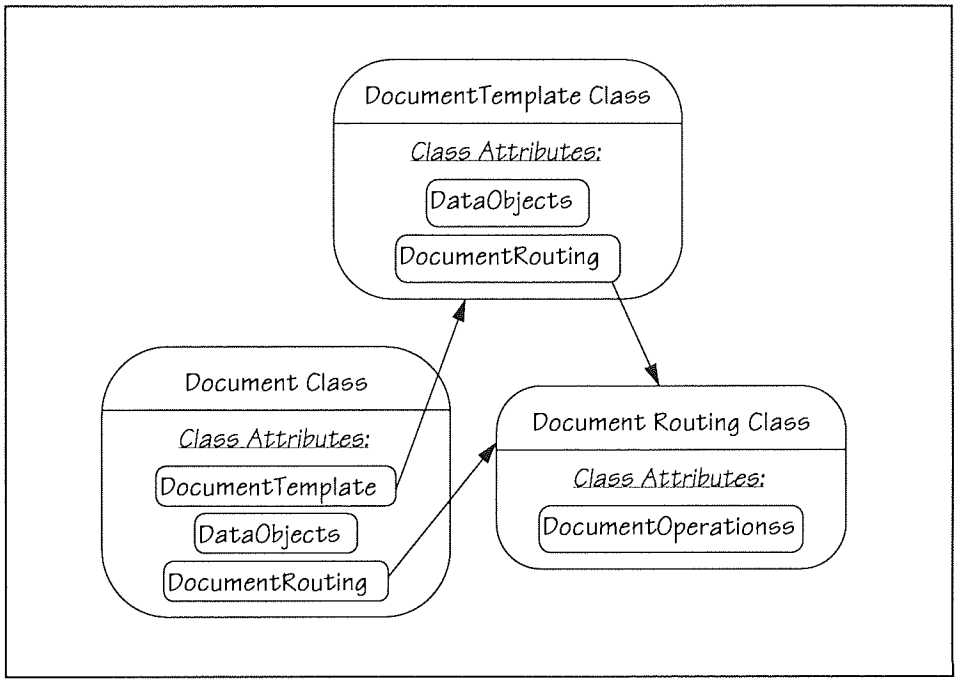


Figure 8-3 The Relationship Between Document Routing and Document Template Classes

The diagram shows that the original structures of the *DocumentTemplate* and *Document* classes that were introduced earlier have now each been extended to include a *DocumentRouting* object. So model-builders will actually define a particular style of document in terms of both the types of information it contains and also the different processing steps that this type of document will take through the organisation. When *Document* objects are created during a simulation, they will adopt both the types of *DataObject* and *DocumentRouting* attributed to their respective template.

It should be noted that in a shop-floor environment *part routings* relate only to the manufacture of a part. When that part is subsequently being used as a component in the manufacture of another item, its movements are defined by the routing for that other item. In other words, the overall procedure for manufacturing any finished product is likely to involve several different routings. This analysis is also applicable

in the context of routings and operations that relate to office work. In other words, a *DocumentRouting* should only include those *DocumentOperations* involved in the preparation of a particular type of *Document*. This may involve just one operation to add the basic information values to a document or may include several operations if, for instance, a series of authorisations are required before a document can be considered 'complete'. When that document is subsequently used as a 'component' in another operation, say, to create another document, its movements will be defined by the routing for that other document.

8.2.2.2 Extending the Paper-Factory Approach

Thus far, modelling office processes in terms of objects has been likened to a production routing and the notion of a 'paper-factory' introduced. There are, however, some significant differences between manufacturing and support department activities that affect how closely administrative procedures match the conventional factory-oriented perception of operations and routings.

One important difference is that while in a manufacturing process all components described by a particular part number are inter-changeable, the 'information components' used in administrative processes cannot be exchanged in the same way. For example, while an accounts clerk may check invoices against goods received notes before issuing payments to suppliers, the clerk cannot check just any goods received note against any invoice. Instead, the checking process depends on corresponding documents being available at the checking station.

Another factor is that in shop-floor operations, component parts are always physically consumed to make finished goods and no longer exist in their original form after the finished goods have been manufactured. In contrast, 'component documents' will not be physically consumed or transformed to create new documents. Instead, the information on the component document will be duplicated onto the new document. The component document will still physically exist after the new one has been created and may even be re-used in other operations. For example, the information contained on an invoice might be used to prepare a cheque for payment of a supplier. The invoice will still physically exist in its original form after the cheque has been prepared. This represents a major conceptual difference between manufacturing and

support department activities and means that not only must any 'component documents' required for a *DocumentOperation* be incorporated into a model of that activity, but their destination after the operation is complete must also be included. In a real business this will typically involve sending the 'used' document to either another department or to some form of document archive and should therefore be represented this way in the model.

Closely related to the idea of being able to re-use information is the notion of duplicate copies of certain types of document existing simultaneously in different parts of a business. These copies will be produced either for action or for information only. In practice, formal administrative systems, such as purchasing, typically prepare certain types of document using pre-printed forms that create several copies simultaneously (Bailey, 1991). Copies are often printed on different coloured paper or uniquely identified in some other way to facilitate sorting. In modelling terms, copies of a particular type of document need to be simulated if they are to be actioned by different entities or at different processing stages within the simulation. As a result, another feature of a *Document* object will be an attribute that uniquely identifies it in some way from other 'copies' of the same object.

Another conceptual difference between manufacturing and administrative routings relates to the inherently variably nature of office work; while documents may take only a few minutes to create or to process once all supporting information has arrived, delays for this information might range from a few hours to several weeks. For example, a buyer may have to agree the cost of requisitioned items with suppliers before a purchase order can be issued. While the purchase order will probably take the buyer just a few minutes to fill out, written confirmation of prices could take several days to arrive by post. In practice, rather than idly waiting to receive this information, a buyer would place the document in a pending file and begin another task. When all the necessary information has arrived the processing of that document would be restarted. In effect, many documents will simultaneously exist in semi-processed states throughout a firm's various support functions and will be processed in parallel.

While semi-processed jobs or work-in-progress is also a feature of shop-floor operations, the tasks performed in office areas are not supported by a control system in

the same way that manufacturing activities rely on material planning systems to ensure materials are available when required. Arguably, the way that information needed for office tasks is demanded as and when required is comparable to an informal version of a 'pull-type' production control system more typically used in manufacturing areas. In other words, information is supplied on demand rather than being held as 'inventory' prior to use. Nevertheless, the way that 'materials' are supplied to administrative functions is fundamentally different to their counterparts in manufacturing areas.

Overall, these different characteristics imply that document processing cannot be modelled simply as series of stages whereby a clerk takes a document from 'stock' held in an input tray, delays while processing the document, and then places new or amended documents into an output tray. This would correspond to a purely manufacturing-oriented interpretation of administrative-type work. Instead, the *DocumentRouting* and *DocumentOperation* classes must be able to encapsulate additional behaviour in terms of the ability to track and identify matching sets of documents or information. Between them, these classes must also be able to represent semi-processed documents and describe the events that will eventually allow processing of these documents to be completed. Each *DocumentOperation* will also need to describe the types of information to be added at a particular processing stage as well as the source for each of these data components.

As a result, each *DocumentOperation* will need to encapsulate features of an office task that are significantly more complex than an equivalent shop-floor assembly operation. In effect, a *DocumentOperation* should be more analogous to a British Standard BS5750 procedure, defining explicitly how a particular task should be carried out. In fact, Figure 8-4 illustrates how the concept of a *DocumentTemplate* has been extended to describe the different processing steps or *DocumentOperations* that a *Document* of this type will go through.

The illustration depicts a *PurchaseOrder* that has been defined as containing five different types of information; *PartNumber*, *QtyToOrder*, *Supplier*, *PartPricePerUnit* and *Signed-Off*. Another attribute of this *PurchaseOrder* is a *DocumentRouting* object that describes how the values of each of these five *DataObjects* will be attributed to *Document* objects during the simulation. The illustration shows that two processing

stages or *DocumentOperations* are involved in this case to prepare the *PurchaseOrder*, namely *Create Order* and *Sign-Off Order*. The structure of the first of these *DocumentOperation* objects is also shown.

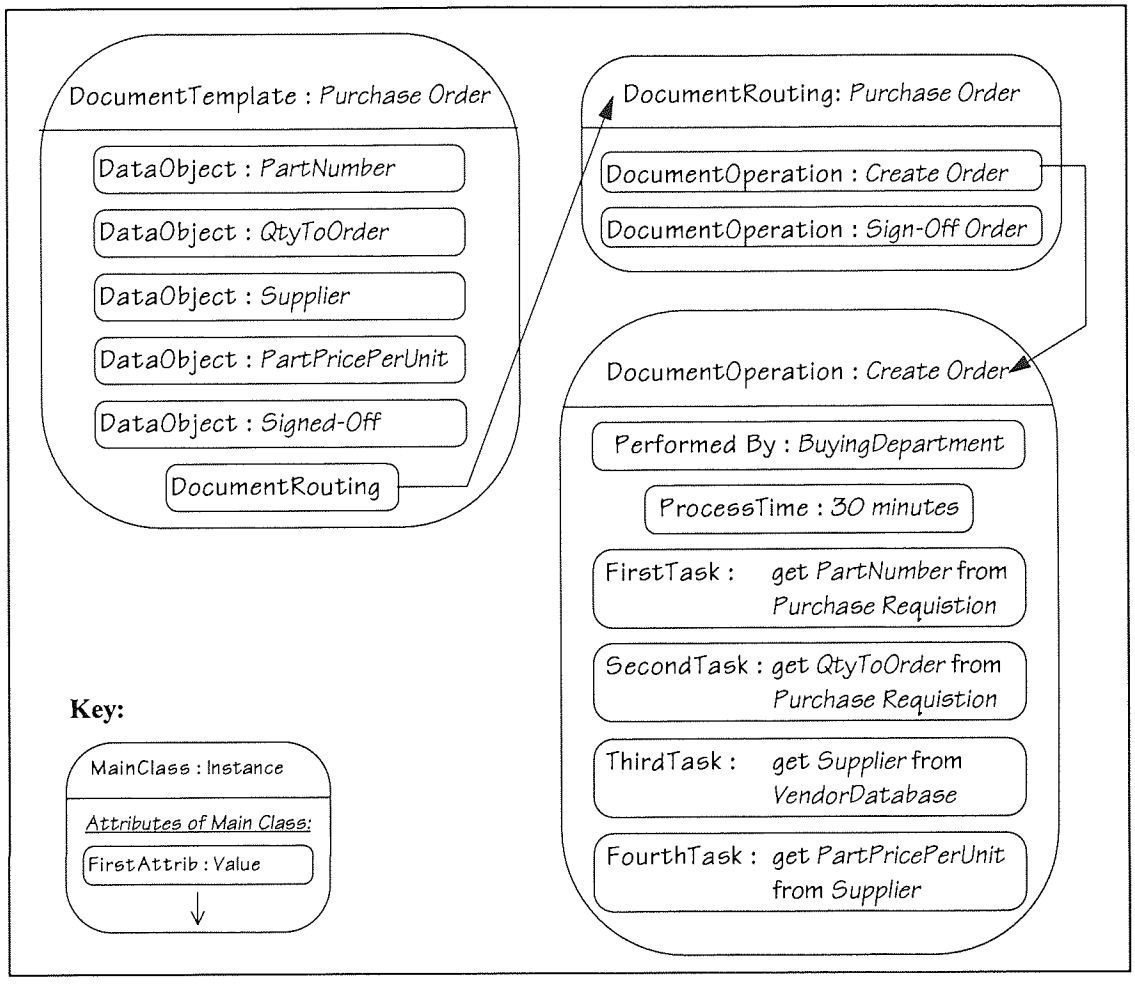


Figure 8-4 Example of Document Template and Document Routing Definition

The *DocumentOperation* appears as a quite explicit set of instructions or mini-tasks, analogous to a formal working procedure that might typically be found in a real business. In this particular example, the first *DocumentOperation* is associated with adding four of the five *DataObject* values to a *PurchaseOrder*, and also identifies how each of these values should be obtained. In this case, the values of *PartNumber* and *QtyToOrder* are taken from a *PurchaseRequisition* document, while the total cost of these parts will be provided by a *Supplier* object.

Describing administrative tasks in this way is important for several reasons. Firstly, entities in the simulation model, such as people, must be able to generate these data values since they will affect the progress of the model. In other words, documents are represented in this type of model as more than just 'tokens', something that tends not

to be the case with other commercial simulators. In addition, model-builders need to represent the different kinds of delay that are likely to occur while this information is being made available in order to properly evaluate the impact of these delays on the performance of a manufacturing system. This means that the consequences of using different kinds of information source should be included in the model. Finally, the notion of a *DocumentOperation* as an explicit set of instructions closely matches the concept of working procedures used in real administrative systems.

In the example just described, other *Document* objects represent one source of information for a *DocumentOperation.* Clearly a task will not be able to be completed until that source document has progressed through its own processing stages to eventually arrive at the appropriate desk or department. Similarly, other entities in the model represent another source of information. In effect three potential sources of information for office tasks can be identified:

- *Transcribed Data* represents information that is copied or transcribed from one document to another. For example, a cheque might be created as payment against a supplier invoice. The name of the payee on the cheque would be taken from the name of the supplier on the invoice. In this case, any delay will depend on the time taken for the invoice to be prepared and moved to the current location.

- *Requested data* represents information that must be formally requested from another company or another function within the same business. A 'request document' will be sent and the receiving object will return another type of document that contains the required information. In this case, any delay will depend on how quickly the receiving object acquires the request document and replies to it.

- *Looked-up data* refers to information that administrators will obtain from a data-base, log-book or similar look-up device. In this case, delays will be affected by the availability of the look-up device.

In modelling terms, these different sources of information can therefore be represented in terms of three *OperationConfiguration* classes. These classes will define the logic

of each task within a *DocumentOperation* in terms of how each piece of information required for that stage will be obtained. When a full set of matching information components is available the operator will finish preparing the document and pass it on to the next process stage in the routing. For modelling purposes, another attribute of a *DocumentOperation* is therefore a process time, indicating how long an activity is likely to take to complete once all the necessary information is available. This particular *DocumentOperation* class structure means that complex administrative delays and dynamic behaviour can be simulated, even though model-builders will construct models of these activities from quite basic elements that relate to real office systems.

8.3 Summary

This completes the first part of the object-oriented analysis of support functions typically found in a manufacturing organisation. The chapter has provided a general analysis of support department activities, and has also identified more specific features of administrative or document processing systems. These features, which have been defined in terms of a set of simulation classes, contribute to the complex delays and dynamic behaviour exhibited by this type of support activity that can ultimately affect manufacturing system performance. The second part of the analysis will establish characteristics of different types of resources actually involved in carrying out administrative-type work, as well as describing other types of support activity in terms of objects.

9. Object-Oriented Analysis II : Modelling People and the WorkPlace

This chapter presents the second part of an object-oriented analysis of support functions. The discussion will focus mainly on describing people, the work-place and the non-administrative aspects of support department work in terms of objects.

9.1 Modelling People

Since people represent a major aspect of any manufacturing system or support department, they should also form an important aspect of any model of that system. This section describes a set of simulation classes that will represent the personnel and skills within manufacturing organisations, and in particular, within their support functions.

9.1.1 Representing Administrative Skills

In modelling terms, all people are essentially identical in terms of their physical attributes, perhaps differing only in relation to the speed at which they can perform a given task. People do differ more significantly, however, in terms of the types of job they are expected to carry out. In a manufacturing environment, for example, a person may be trained to operate a particular machine, to transport materials from stores, or to supervise a production cell. Similarly, in an office environment employees will be trained and authorised to perform certain types of support activity. A clerk employed in an accounts department, for example, is likely to be responsible for processing invoices.

In terms of an object model, this implies that *Person* objects can be differentiated by the *Skills* they possess. In this context a *Skill* represents the know-how and authority to perform a certain task, such as repairing a machine or raising a purchase order. In practice, since the types of document used by firms will vary, the know-how to process these documents will also differ between firms. More importantly, because the object model described so far implies that document styles will be defined by users at model-build time, the developer cannot embed specific know-how or *Skill* classes into the simulator that define how these various types of documents should be processed.

Fortunately, however, the concept of a *DocumentOperation* was introduced earlier as being an explicit set of instructions describing how each particular style of document used in an organisation should be prepared. In other words, the know-how for processing different types of document, or *DocumentTemplate* objects, will actually be defined by the model-builder as a set of *DocumentOperations* for each template. In modelling terms this means that the people (or *Person* objects) in a support function can be differentiated according to which *DocumentOperations* they are responsible for carrying out. It follows that a more generic *OfficeSkill* class can be established that allows a range of tasks or *DocumentOperations* to be attributed to a *Person* object in the model.

Figure 9-1, for example, depicts a set of objects that represent different staff in a support department using the *Person* and *OfficeSkill classes*. The skills represent each person's ability to complete one of the sample administrative tasks described in the previous chapter.

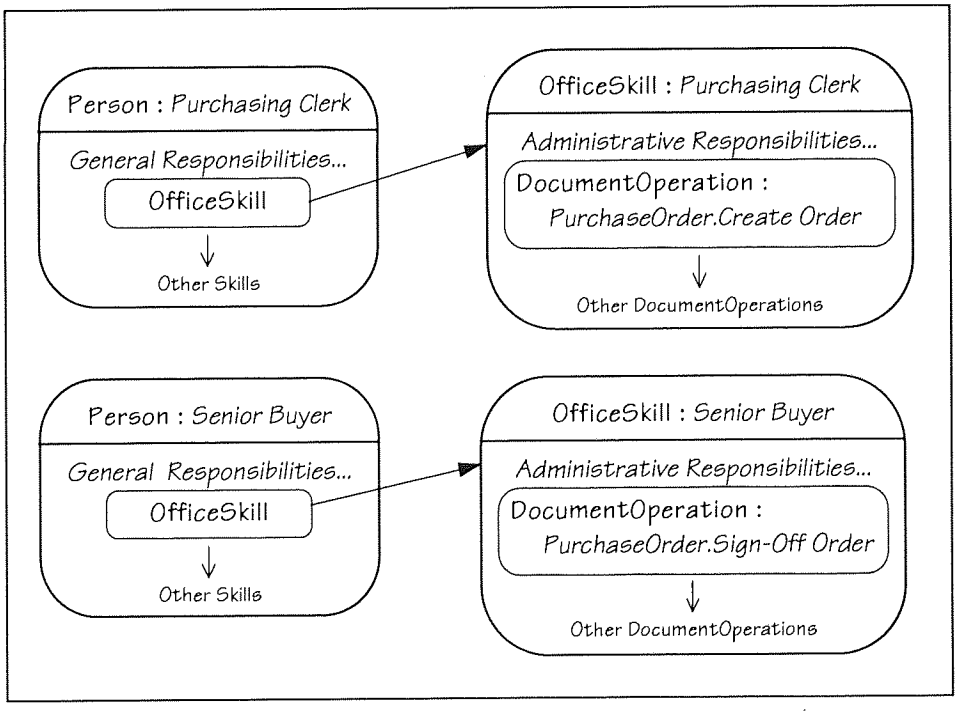


Figure 9-1 Example of Person and Office Skill Objects

The diagram describes two people, a *Purchasing Clerk* and a *Senior Buyer*, that work in a particular purchasing department. Both of these *Person* objects have been attributed a range of skills, including their own particular *OfficeSkill*. For each of these *OfficeSkills* the model-builder has described which tasks each person will perform in the model by attributing *DocumentOperations* to their respective skill object. The

Purchasing Clerk, for example, has been configured by the model-builder to have the knowledge and authority to create purchase orders, while the *Senior Buyer* is deemed responsible for signing-off these orders. This means that during the simulation, when the clerk has finished creating an order, the order must be passed to the senior buyer for final approval.

Using this set of objects, the model-builder could easily re-configure this model to assess how differently the purchasing department might perform if, for example, the clerk were able to sign-off his or her own purchase order documents. Overall then, the notion of a generic *OfficeSkill* class provides a flexible way for model-builders to accurately reflect, in the simulation model, the different administrative responsibilities of personnel within their own particular organisation.

9.1.2 Representing Other Human Aspects of Support Departments

The analysis of the human aspects of support departments has so far established how the ability of office staff to prepare different types of document can be represented in terms of *OfficeSkill* objects. In fact, all of the object-oriented analysis described so far in this chapter relates to the *repetitive* and *inherently variable* categories of office work defined in the previous chapter. However, creative or managerial-type work was mentioned earlier as another contributor to support department workloads and should also therefore be included in a support department model. It was also established that this type of work is generally harder to measure, and consequently more difficult to model properly in comparison to the other kinds of office task. Fortunately, however, creative work typically accounts for only a relatively small proportion (10 per cent) of an office system's workload which means that the possibility, and indeed significance, of invalidating a model by misrepresenting this type of activity is greatly reduced.

For these reasons it seems more practical that manufacturing system engineers should model this type of activity in much less detail than the other work categories described so far. In other words, attention should focus on how the incidences of people taking part in creative work, such attending meetings or doing management-related tasks, will affect their availability to complete other administrative duties. The actual information generated by this type of work will therefore be less significant from a manufacturing

system design or simulation point of view.

In modelling terms this means that a person's ability to participate in this kind of work should be represented as a special kind of office skill, or in effect, a 'creative' skill. Given that information processing in this context does not actually need to be modelled, the main objective of this skill will be to affect a 'creative' person's availability to complete other administrative duties that are likely to have a more significant impact on manufacturing system performance. In other words, model-builders can use this skill to define a pattern of behaviour such as a number of hours per day or per week when that person is involved in creative work or otherwise busy, and is therefore unable to process the types of *Document* or *DocumentOperation* described earlier. In practice, this is likely to be particularly useful for modelling the availability of managers or team-leaders to sign-off orders or other similar documents. It is very important that this behaviour can be defined in terms of a probability distribution to reflect the variability and uncertainty likely to be involved in relation to exactly when and for how long a person will be 'unavailable'.

In practice, the notion of regular or irregular patterns of behaviour is likely to be applicable to other aspects of a support department model. In a real manufacturing system or support function, for example, people might participate in quality circles or training programmes on a regular basis that would effectively mean them being temporarily relieved of their normal duties. Affecting resource levels in this way is likely to contribute to complex and dynamic variations in the output of the departments to which these people belong, and should therefore be included in a support department model. Given the general applicability of this class, the term *MiscellaneousSkill* is therefore a more appropriate and generic description of the functionality it represents.

Finally, since most documents are ultimately intended to generate some course of action, a way of describing how other people in a manufacturing business will act on the contents of a document also needs to be established. In other words, some kind of skill that enables people to 'action' particular types of document is required. As with the *OfficeSkill* class, however, the main problem from a development point of view is to provide a generic solution since document types will only be defined by users at model-build time. As such, the solution to this aspect of the new simulator is

essentially more of a design issue than an analysis problem and will therefore be discussed in more detail as part of the object-oriented design phase.

9.2 Modelling the Work-Place

The term 'work-place' is used here to describe any location where some type of business activity takes place. For several reasons a simulation model needs to be able to represent the kinds of work-place where administrative tasks are carried out. For example, where information or documents are physically moved between different work-places, such as departments or companies, delays will be incurred that may affect shop-floor performance. Consequently, these delays must be included in the simulation model. In addition, it was established earlier that people view a real business in terms of objects that also includes work-places like desks, machines, departments, other companies and so on. Including these entities in a model will therefore add more realism to the simulation process.

In the context of the manufacturing system design process, a business can be readily described in terms of objects that represent the work-place at different levels of aggregation. It should also be noted, that in an object-oriented approach, the concept of aggregation is normally developed in a bottom-up fashion (Taylor, 1995). At the lowest level, therefore, a manufacturing firm can be viewed as different assortments of resource objects, grouped together to perform related business tasks, such as processing invoices or transforming raw materials into sellable products. In an office environment, in particular, these resources will include entities such as people (or *Person* objects) and *Desk* objects. In other words, a *Desk* represents the lowest-level or most basic type of work-place object in a support department model.

At the next level of aggregation, these different assortments of resources can be viewed in terms of *Department* objects, typically including purchasing, production control, manufacturing, finance and so on. In terms of an object model of support functions, a *Department* therefore represents a collection of *Person* and *Desk* objects. A model-builder can use this generic notation to more closely describe the actual departments found in a manufacturing firm. This would not be possible if the model builder was restricted to constructing models from a pre-defined set of department classes, such as

purchasing department or *finance department*, that were included in the simulator by a developer.

Finally, at the highest level of aggregation an entire business can be viewed as a work-place, or *Company* object, that comprises an assortment of different *Department* objects. By implication, the firm carrying out a manufacturing system design study must include itself as a work-place in the simulation model, in the form of a *Company* object and different *Department* objects. In practice, the real firm is also likely to interact with other *Company* objects, typically suppliers and customers. In terms of shop-floor performance, customers will affect the timing of orders being placed for the firm's goods, while suppliers will affect the arrival time of materials needed to produce those goods. Consequently, these other companies should also be included in the model to assess how their interaction with support functions will affect manufacturing system performance.

It is unlikely, however, that a manufacturing system engineer will need to, or even be able to, model the internal activities of suppliers or customers in detail. For this reason, it is more appropriate that these entities should be represented as aggregate work-places. They will still need to be able to process the same types of *Document* object and materials as the main company being modelled, but in a less detailed way. In modelling terms these entities can therefore be represented as specific types of the *Company* class, namely as *Supplier* and *Customer* classes, that encapsulate a limited version of the *OfficeSkill* logic described earlier.

Figure 9-2 uses a very simple example, to illustrate how the physical work-place can be represented using the different classes that have just been described. For clarity, the other physical entities described so far, such as *Document* and *Person* objects, are not included in this diagram. Nevertheless, the illustration emphasises how these different aspects of a business environment can be represented in terms of interacting work-place objects that relate closely to the real-world.

Another aspect of any business environment will be a mailing system that enables documents to be moved, physically or electronically, around a company as well as between that company and its suppliers and customers. In modelling terms a

conventional internal mailing system handling paper documents will involve *Person* objects with specific know-how or *Mailing* skills to collect and distribute those documents around the business. In terms of inter-company transfer of paper documents, it is more appropriate that aspects of this same skill be modelled as an aggregate *PostalSystem* class rather than modelling the internal activities of such a system in detail. In both cases, however, delays will be incurred as items are physically mailed between locations.

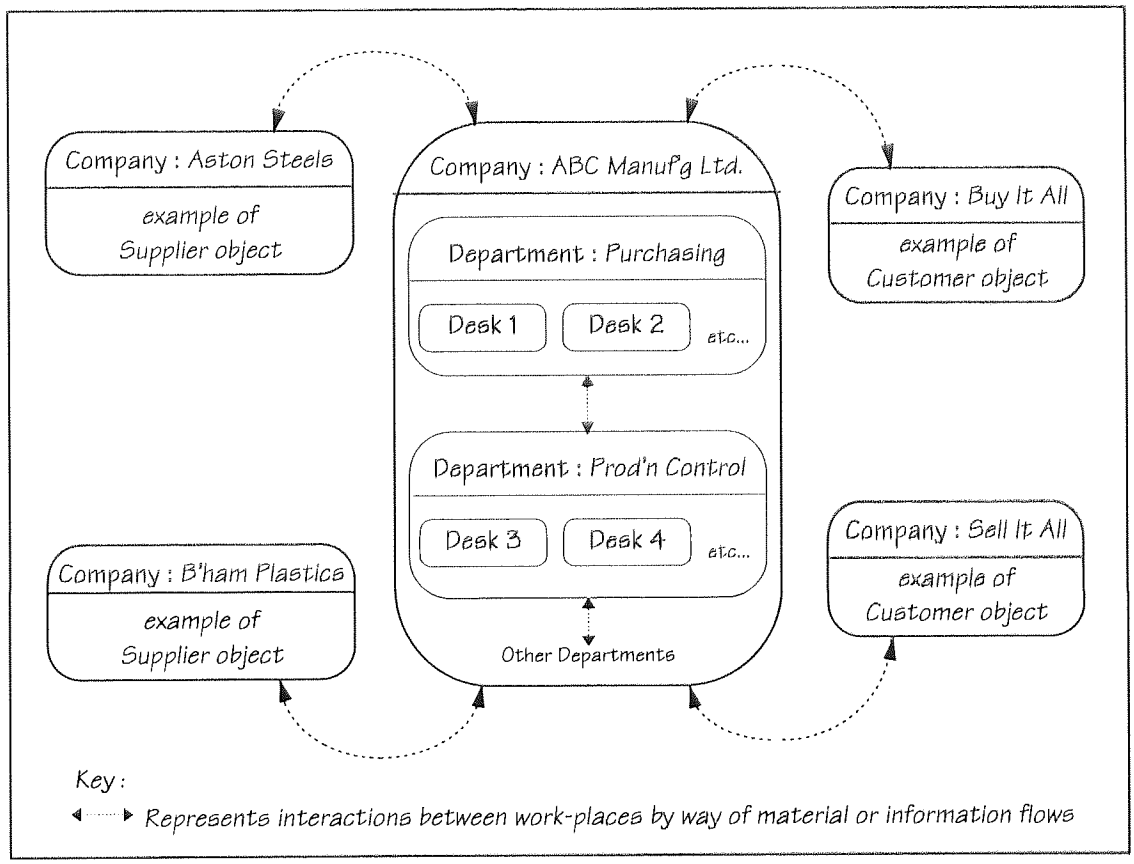


Figure 9-2 Examples of Company, Department and Desk Objects

In addition, the concept of an *ElectronicMailSystem* class aptly describes how documents are transferred electronically within and between organisations. Again, this class is similar to an aggregate version of the *Mailing* skill, although it can be assumed that no significant transfer delays will be incurred if documents are sent electronically. In other words, while a model-builder should be able to use these mailing classes to construct models of document-handling systems that closely match real-world systems, the underlying modelling logic within each class will be essentially the same. The next chapter will discuss the document-transfer aspect of administrative activities in more detail.

To conclude, a set of work-place classes and mailing system classes have been established to describe the wider context in which administrative activities are carried out. This section essentially completes the object-oriented analysis of office-type work in manufacturing organisations; the remainder of the chapter will discuss other types of support activity in more detail.

9.3 Modelling Other Types of Support Function

It was established earlier that some support activities involve more than just office-type work. For example, functions such as goods inwards and despatch will involve a combination of administrative work and material handling. In terms of evaluating manufacturing system performance, model-builders need to understand how delays incurred in these storage areas will affect the shop-floor. Given the generic nature of the document-related classes described so far, the administrative aspects of parts storage can be easily modelled using these document classes. Additional classes only need to be established to represent the entities that contribute to material handling delays.

Aside from administrative duties, a real goods inwards or despatch area is essentially a collection of people, material handling devices and physical storage areas for holding part inventories. Given the context of the modelling problem described in this thesis, it is not actually necessary to represent physical storage areas, such as shelving or racking systems, in any detail. In other words, this aspect of a stores function can be represented as an aggregate *Storage* class that simply holds quantities of different parts. In contrast, the time taken to add or to remove parts from a *Storage* object will influence manufacturing system performance and should therefore be modelled in more detail. Delays will mainly be affected by the availability of people and material handling devices. As a result, the interactions of these entities with *Storage* objects need to be included in an object model of this type of support function. Figure 9-3 illustrates people and material-handling devices as well as other entities needed to represent the non-administrative aspects of stores functions in terms of objects.

In fact, the illustration introduces several new simulation classes and also describes material-handling activities in the context of the manufacturing system model, as well

as the wider business model, by reference to some of the work-place classes described earlier. It should be noted, however, that Figure 9-3 only depicts those activities that introduce materials into a manufacturing organisation and to shop-floor areas, as well as those activities that pass finished goods onto customers. In other words, any material-handling that occurs within the boundaries of a manufacturing system is ignored since it does not constitute a support activity in the context of the thesis.

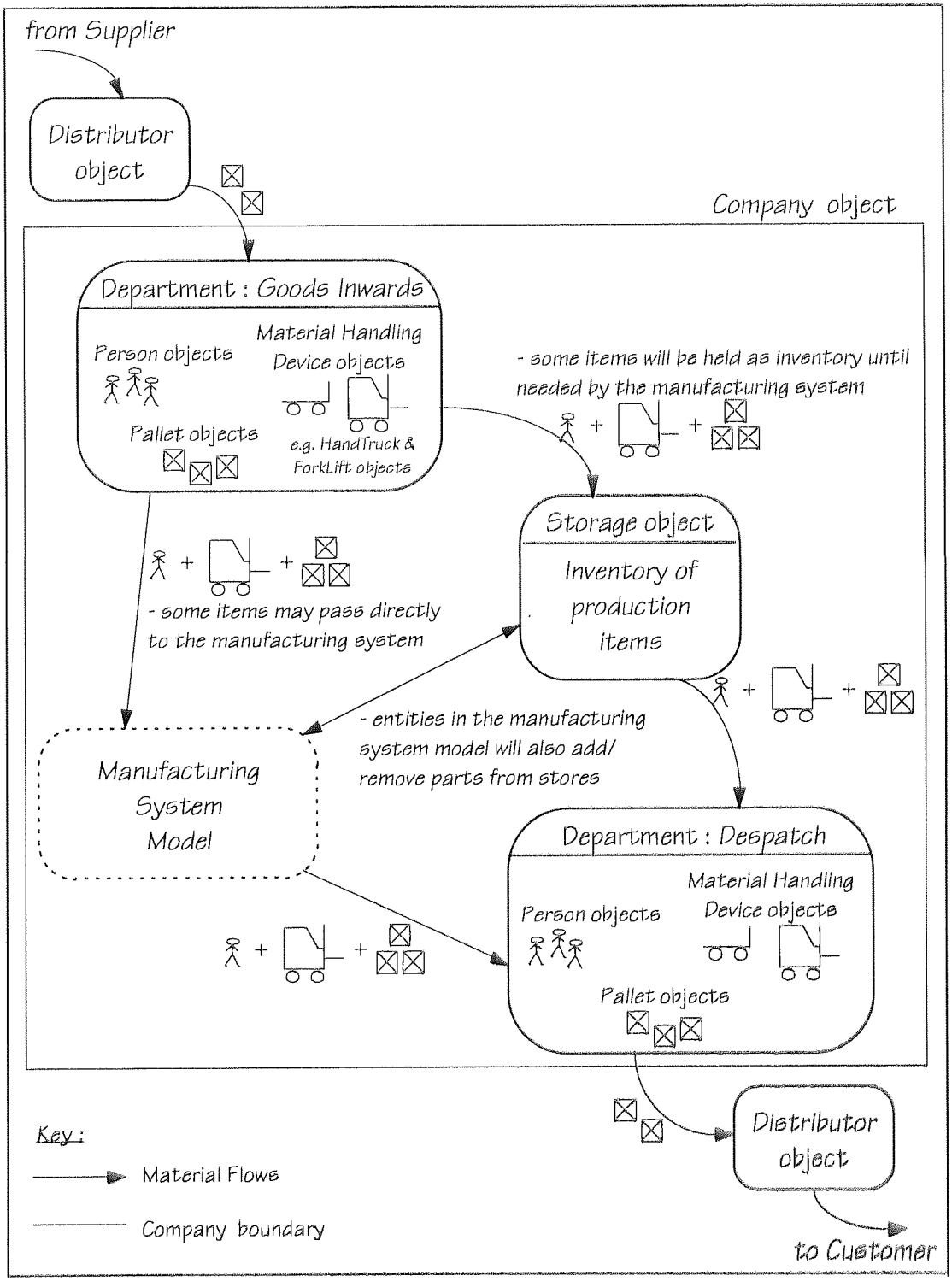


Figure 9-3 An Object Model of a Stores Function

In terms of introducing material into a firm, the concept of a *Distributor* class represents the means by which raw material is transferred from *Supplier* objects to the *Company* being modelled. Another instance of this class enables finished goods to be transported from the organisation to *Customer* objects. The *Distributor* concept presented here is intentionally quite simple in the sense that it represents only the basic functionality and logic needed to move materials between two manufacturing organisations, or *Company* objects. Nevertheless, this class helps to ensure that the model relates closely to how this particular task is performed in real manufacturing firms.

Within the actual company being modelled, the concept of a generic *Department* class that was introduced earlier has been re-used to represent the *Goods Inwards* and *Despatch* functions. However, rather than containing just *Person* and *Desk* objects, these work-places now also contain *Pallet* objects as well as *Material-Handling Devices* in the form of *HandTruck* and *ForkLift* objects. These entities interact with *Storage* objects to provide material-handling support to the manufacturing system. For example, *Pallet* objects are intended to represent containers of production items waiting to be added to or removed from a *Storage* area. A combination of a *Person* and either a *HandTruck* or *ForkLift* will move *Pallets* between stores and the manufacturing system. In modelling terms these two types of material handling device differ in terms of their carrying capacity and speed of moving materials but both need to be operated by a *Person* object. In other words, one aspect of the delay imposed by stores functions on the shop-floor can be modelled in terms of the speed and availability of different types of material handling device employed in these areas.

Another aspect of material-handling delays relates to the time required by a *Person* to actually decant or place items in physical storage. In modelling terms this type of delay can be represented as an attribute of a *Storage* object since the delay will depend on how each type of part is physically stored. In other words, the time required to add or remove a certain quantity of particular part can be defined by a model-builder as an attribute of a *Storage* area. It should also be noted that while Figure 9-3 shows only a single *Storage* object, in practice a model might need to contain several instances of this class to represent alternative storage locations where perhaps different categories of part are maintained. Consequently another attribute of this class should be a list that

defines which particular items it can store.

The maintenance and repair of production equipment was also mentioned earlier as a another example of a support activity that involves more than just office-type work. In terms of the manufacturing system design process, this kind of activity is already associated more closely with shop-floor performance than the other indirect activities describes so far. Indeed, by its very nature machine maintenance tends to be carried out in-situ, and so its impact on manufacturing system performance is invariably more visible to decision-makers concerned with design or operational issues.

For this reason, many aspects of machine maintenance are already addressed in manufacturing system design methodologies and factory simulators. For example, Lucas Engineering & Systems (1989) identify the importance of preventative maintenance and tool management strategies in the context of an overall 'manufacturing systems' approach. Furthermore, the Advanced Factory Simulator is already able to model the impact of machine breakdowns and repair times on manufacturing system performance, although it cannot model any administrative delays associated with this type of activity. Nevertheless, it means that combining the new support department classes described in this chapter with existing shop-floor classes will be sufficient to model a maintenance function in terms of objects. Consequently, no additional classes need to be developed to describe this aspect of a manufacturing firm's activities.

9.4 Summary

This chapter completes the object-oriented analysis of the support functions typically found in a manufacturing organisation. Overall, the analysis has identified a range of objects for describing administrative systems, including the people and the working environment where this type of activity normally takes place. The analysis has also addressed other types of support activity related to manufacturing system performance, such as goods-inwards, stores and maintenance. The author is unaware of any other simulation tool that represents different types of support activity in this way.

Any discussion of the Advanced Factory Simulator or WBS/Control has been

intentionally avoided or been very limited at this stage to avoid biasing the output of the analysis toward the existing architecture of those systems. In effect, the analysis has identified an ideal set of objects that reflect the kind of support activities contributing most to manufacturing system performance. Nevertheless, the object-oriented design that develops from this analysis will ultimately be implemented using the existing simulation classes that represent the shop-floor and production control aspects of a manufacturing business. The next chapter therefore describes the progress from this object-oriented analysis to an object-oriented design, and places this ideal set of support department objects more in the context of the Advanced Factory Simulator and WBS/Control.

10. The Design of WBS/Office

This chapter describes the object-oriented design of the new simulator that has been proposed in preceding chapters. The design builds on the output of the object-oriented analysis phase and incorporates that object model into an existing library of simulation classes, namely the Advanced Factory Simulator (AFS) and WBS/Control. The new simulator, which has been called WBS/Office, extends the scope of this existing class library in order to simulate the support activities in a manufacturing business.

10.1 Introducing WBS/Office

It was explained in the introductory chapter that the research described in this thesis has been completed within the context of a broader research agenda proposed by Love et al. (1992). This proposal relates to the development of a novel modelling tool, referred to as the Whole Business Simulator (WBS), that will be able to predict the financial consequences of different types of decision in manufacturing organisations.

It follows that a dynamic model of the office functions and other support departments within a manufacturing firm will represent a very important part of WBS. As a result, while the context of this thesis is the manufacturing system design process, and therefore the simulator described here is aimed primarily at improving that process, certain aspects of the simulator have also been influenced by the needs of the broader research agenda. Most obviously, the name *WBS/Office* has been selected to reflect the WBS influence, but some aspects of the design have also been developed with future WBS requirements in mind. These specific features will be introduced as necessary during the discussions that follow.

10.2 Overview of the AFS and WBS/Control Class Libraries

Since this chapter will describe the design of WBS/Office in the context of the existing AFS and WBS/Control class libraries, it is necessary to provide an overview of both these systems. A short summary will therefore follow of the most relevant aspects of each system from a development point of view, although the reader is referred to Ball (1994) and Boughton (1995) for more complete descriptions of their respective system

architectures. It should be noted that while AFS represents a standalone application in its own right, it is now also effectively a sub-set of WBS/Control in the sense that WBS/Control is an enhanced version of the core AFS architecture. For this reason, the description of AFS that follows also relates to the core of WBS/Control.

10.2.1 The Macro Architecture of AFS

To end-users, AFS appears as a single software application even though the system actually comprises three separate applications; a *Core Simulator*, a *Graphical Display* and a *User-Interface*. These different parts interact with each other to produce the overall system.

As its name suggest, the *Core Simulator* represents the essence of the AFS system. It contains all the classes that are fundamental to any discrete-event simulator such as a simulation executive, a random number generator and a results collator. In other words, it includes those classes needed to mimic the progress of real time or dynamic behaviour in a model, to introduce variability and uncertainty into models of stochastic systems, and finally to present simulation results in an appropriate format.

Clearly these are simulation-specific classes in the sense that they have no real-world equivalence. Nevertheless, this part of AFS also contains the classes that do correspond to real objects in a manufacturing organisation. More specifically, this core part of the system includes classes that describe the logic and behaviour of elements that make-up the shop-floor aspects of a manufacturing system such as machines, operators, parts, material handling devices and so on. Consequently, by also including the new classes that will correspond to support functions into the core simulator, WBS/Office will be able to re-use both the simulation-specific classes as well as those that describe shop-floor activities. In other words, the core simulation logic that currently enables shop-floor objects to exhibit dynamic and stochastic behaviour can be re-used by any new support department objects.

The *Graphical Display* allows users to visualise what is happening during a simulation experiment. This part of the system responds to instructions from the core simulator, such as a machine changing state from running to idle, and displays the appropriate image on the user's computer screen. Fortunately, the core simulator has been

designed in such a way that its own functionality can be extended without changes to the graphical display. Consequently, this part of AFS will not need to be modified to accommodate WBS/Office.

The *User-Interface* controls user access to the core simulator and graphics display parts of AFS. In other words, the interface allows model-builders to create objects from the library of classes in the core simulator that correspond to entities in a manufacturing system. Unlike the core simulator and graphical display, the implementation language for the user-interface is not object-oriented. Nevertheless, this part of AFS has still been developed in a object-oriented style such that model-builders are presented with individual menus and dialogues that correspond to particular classes in the core simulator. While a user-interface represents an extremely important element of any data-driven simulator, the structure of this part of the system actually depends very much on the underlying class designs. For this reason, examination of WBS/Office's user-interface will be very limited in the context of the object-oriented design presented here. The next chapter, however, describes the implementation of WBS/Office and will therefore describe, in more detail, new features of the user-interface needed to model support functions.

10.2.2 Architecture of the Core Simulator

10.2.2.1 Overview of Class Hierarchy

The entire *Core Simulator*, including those classes within it that represent machines, parts, operators and so on, has been implemented in the form of a single class hierarchy. This hierarchy has been specifically designed to ease the task of extending the scope of the core simulator to model other domains, and is based around four main classes; *tRootAFS*, *tSimulation*, *tPhysical* and *tActivity*. These classes are related to each other through the principle of inheritance as illustrated in Figure 10-1. For example, the *tPhysical* class will have its own particular attributes but will also inherit all the attributes of the *tRootAFS* and *tSimulation* classes.

These four are all abstract classes in the sense that they have no equivalence in a real manufacturing business. In modelling terms, however, they each encapsulate particular features that describe aspects of many different types of entity that do exist

in a real business. Other classes representing these real-world entities will therefore appear in the main hierarchy, descending ultimately from whichever of the four main classes best encapsulates their particular characteristics.

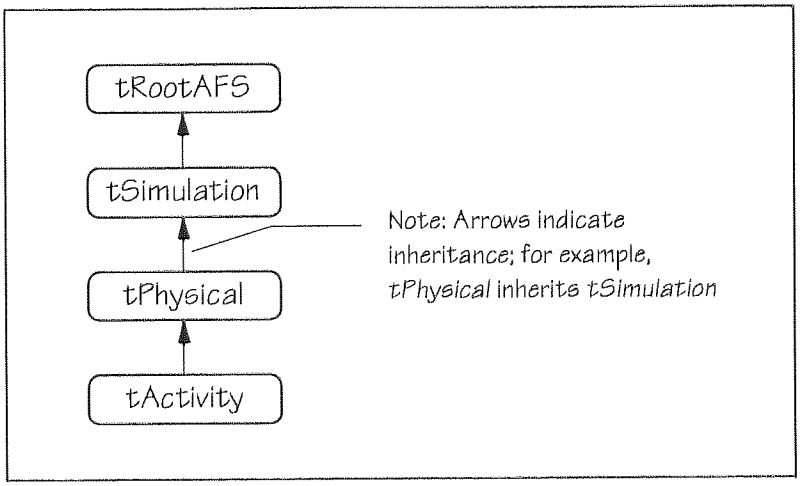


Figure 10-1 The Core Simulator Class Hierarchy in AFS

For example, the commonest and most basic attributes that uniquely identify different entities in a model, such as a name, are encapsulated in *tRootAFS*. This represents the base class, all other classes in the AFS architecture are ultimately descendants of *tRootAFS*. The *tSimulation* class encapsulates additional properties that enable objects to exist in different states such as idle, waiting, processing and so on, and also allows these objects to change their state as the simulation progresses. Clearly many of the resources in a real business, such as people and machines, will need to be modelled by inheriting these properties as well as the other more basic attributes.

The *tPhysical* class lets objects that have a physical presence in the real-world to also physically exist in the simulation world. In other words, objects descendant from this class will be attributed a position and size, as well as an icon so that they can appear in the graphical display. The *tActivity* class provides the capability to send and receive 'messages'. Messaging is a mechanism included in AFS that allows objects in a model to be loosely coupled in the sense that they possess minimal knowledge of one another and yet at the same time are able to interact. This means that new classes can be added to the library with minimal knock-on effect elsewhere in the system.

Importantly, end-users of AFS are not aware of the existence of any of these abstract classes, and will deal only with 'end' classes such as *tMachine*, *tOperator* and

tPartNumber that correspond to real-world objects. In addition, since the properties encapsulated in these four main classes are quite generic they can be applied to model entities from both shop-floor and support areas, and are therefore very relevant to the development of WBS/Office.

10.2.2.2 The Role of Class Managers

Another important feature of the core simulator architecture is the concept of class mangers; for every class in the main hierarchy, such as *tPartNumber* and *tOperator*, there is a corresponding class manager, in this case *tPartNumberManager* and *tOperatorManager*. These class managers are used only for managing the model-building process; they are not intended to have any real-world equivalence nor will they take any active part in a simulation run. In fact, class manager objects represent the link between the user-interface and the core simulator, although model-builders will be unaware that these additional objects even exist. Each class manager is responsible for creating, editing and destroying objects of the appropriate class, as well as collating simulation results for these objects.

Within the simulator, the library of class managers has been implemented as an object hierarchy that effectively mirrors the main simulation class hierarchy. In terms of extending the scope of AFS to model support functions, these class managers represent an unavoidable development overhead; designing them provides nothing in the way of novel or new functionality for the simulator. As a result, the class managers that have had to be developed to support the new simulation classes in WBS/Office will not be described in the main body of the thesis. Descriptions are provided, however, in Appendices 1 & 2 and the reader should therefore be aware of the additional work that creating these manager classes involved, even though it has not been necessary to discuss this aspect of WBS/Office in detail.

10.2.3 Features of WBS/Control

The significance of a production control system in determining shop-floor performance was emphasised earlier in the thesis, and so the design or choice of this system represents a very important part of the manufacturing system design process. WBS/Control encompasses a library of simulation classes for modelling different types

of production planning and control system. This includes not only conventional control systems such as material requirements planning (MRP) or Kanban, but also hybrid systems and potentially even completely novel production control techniques (Boughton, 1995). This means that WBS/Control can be used to model the control systems in a wide range of different manufacturing organisations.

Since a control system also represents a specialist type of support activity it should also be included in a model of a firm's support functions. In fact, the classes that make up WBS/Control will form a very important part of WBS/Office for several reasons.

Firstly, a control system model will be able to generate production information, such as suggested works orders or purchase orders, that more closely matches the information produced by a real manufacturing business. This is crucial since in WBS/Office the information values carried by documents are intended to affect how the model progresses. The new classes represented by WBS/Office will add more realism to this part of the model by simulating the delays involved in producing and transferring production control information to the shop-floor and other parts of the business.

Another reason for incorporating WBS/Control into WBS/Office is that a real control system will generate many of the transactions to be handled by real support departments. The workload placed on a purchasing department, for example, will be affected by the frequency and number of purchase requisitions raised by a control system. Since the capacity of any support function will be limited, the level of demand for their services will affect how quickly this information can be processed, which in turn will affect the performance of any manufacturing system that depends on this service. The model must also be able to replicate this type of behaviour.

WBS/Control classes have been designed around the AFS class hierarchy, specific examples will be described later. This means that WBS/Office classes can be more easily designed to fit into the same hierarchy and thereby model the types of delay incurred between a production control system and the shop-floor more realistically.

This completes the overview of AFS and WBS/Control which has described the core object-oriented architecture shared by these systems. The remainder of this chapter

describes how the object-oriented analysis of support functions that was described in preceding chapters has been translated into a set of simulation classes that integrates into this existing architecture. More specific aspects of AFS and WBS/Control will be introduced as necessary. It should also be noted that to improve the readability of the text, the names attributed to some of the AFS classes that will be introduced may differ slightly from the actual names used to implement the software. Similarly, the prefix '*t*', as in *tPerson*, will be used to indicate a class name or type as opposed to the more general class descriptions used so far.

Furthermore, while the concept of uncertainty and variability will not normally be mentioned explicitly in relation to any particular class, this characteristic forms a inherent part of nearly all aspects of the design of WBS/Office. In other words, where class attributes such as speed, time or efficiency are introduced, it is generally implied that these parameters will actually be defined at model-build time in terms of probability distributions rather than as fixed or deterministic values. Specific examples will be illustrated in the next chapter.

10.3 Modelling the Business Environment

10.3.1 Work-Place Classes

The previous chapter introduced the notion of a 'work-place' to describe any location where some type of business activity is performed. This same concept has been implemented in AFS in the form of a set of *OperationStage* classes. These classes have also been designed to support the principle of modelling a business or work-place at different levels of aggregation. In fact, AFS defines three levels of aggregation; workstation, workcentre and department.

A *workstation* refers to objects such as machines or inspection benches and therefore represents the lowest-level or most basic type of work-place in a factory model. A *workcentre* groups together workstations that perform similar tasks. In a model of a machine shop, for example, lathes and milling machines might be grouped together into different workcentres. Finally, a *department* groups together related workcentres; in the above example, the machine-shop represents a department object. Figure 10-2 illustrates how these concepts have been implemented in terms of an object hierarchy

and also shows how new WBS/Office classes specifically introduced to model support functions have been incorporated into this structure.

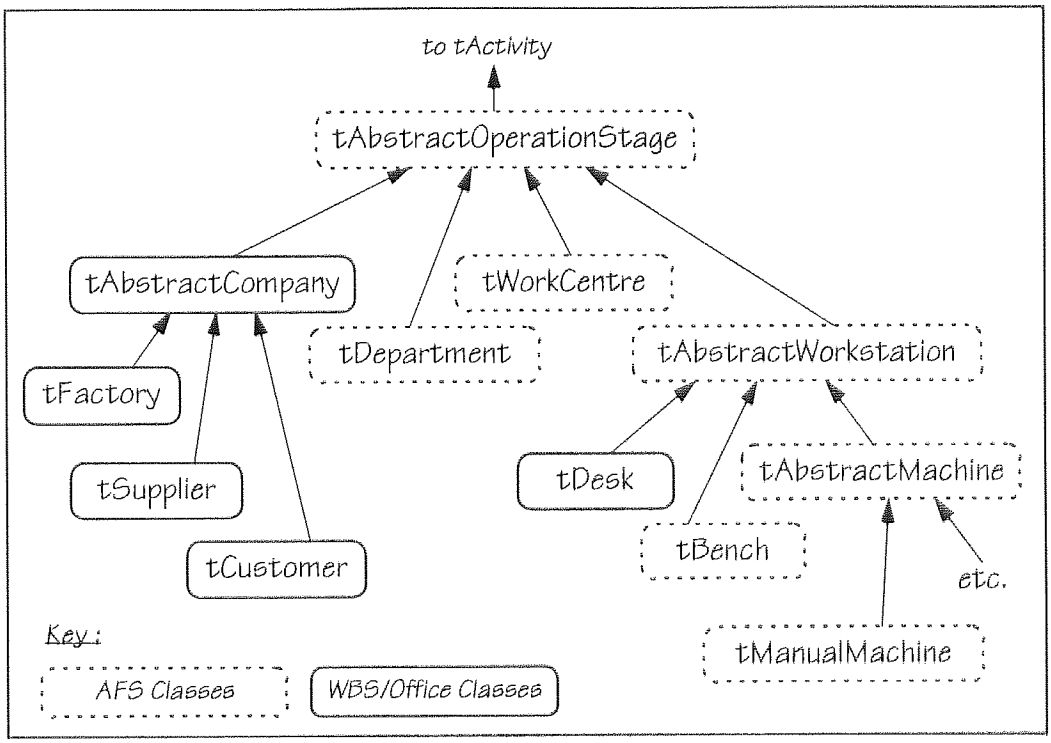


Figure 10-2 Modelling the Work-Place in WBS/Office

It should be noted that, in modelling terms, the concept of aggregation describes *part-of* relationships between different types of object; for example, a workstation is viewed as being part-of a workcentre. In contrast, Figure 10-2 shows that the *OperationStage* classes have been implemented in AFS using *type-of* inheritance relationships. The *tBench* class, for instance, is presented as a type-of *tAbstractWorkstation* which in turn describes a type-of *tAbstractOperationStage*. In other words, the actual classes that the original developer of AFS included in this architecture to represent workstations, workcentres and departments are actually only related in the sense that they are all types of the same parent class. Aggregation is not the primary relationship between these classes in the hierarchy.

This apparent contrast between a model-builder's view and a developer's view of aggregation is due to limitations of the programming language used to develop AFS which only supports type-of relationships. In practice, this has meant that developers of these *OperationStage* classes have had to represent their 'part-of' relationships in some other way. Consequently, an attribute of each aggregate object is a list,

configured by the model-builder, that contains those work-place objects that are part of the aggregate object. For example, model-builders will construct models of the various departments in their business by defining a list of workcentres in each department.

This same principle has been adopted for the new WBS/Office *OperationStage* classes. In other words, the new classes have been incorporated into this existing structure according to type-of relationships while aggregation relationships are represented as additional attributes of each class. For example, in view of the object-oriented analysis, the concept of *Company* objects has been introduced. In fact, three company classes have been designed; *tFactory*, *tSupplier* and *tCustomer*. The *tFactory* class enhances the existing AFS concept of a manufacturing system by describing the overall business to which the system relates as an object. In terms of aggregation, a *tFactory* object represents a collection of *tDepartment* objects such as purchasing, production control, manufacturing and so on. This new class therefore provides a more complete description of the whole business and also means that the impact of interactions with other firms that support the manufacturing process can be included in the model.

In fact, the *tSupplier* and *tCustomer* classes are intended to represent specific types of business that will provide either information or materials needed by the manufacturing system. These are also aggregate classes but in the sense that they represent these other companies in a less detailed way than the firm to which the manufacturing system relates is modelled. In other words, while model-builders will describe their own business, or *tFactory*, in terms of departments, workstations, people and so on, *tSupplier* and *tCustomer* objects will encapsulate a much more limited representation of these same entities.

A *tSupplier* object, for example, will only be able to process purchasing-related information and provide a range of parts to the *tFactory* object. In contrast, a *tCustomer* object will only be able to generate sales or demand-related information and accept goods produced by the *tFactory* object. To model suppliers and customers in this way, their respective classes encapsulate limited versions of the same logic used by person objects to process documents and information. The design of classes that represent people and their skills will be described in more detail later in the chapter.

WBS/Office also re-uses the existing *tAbstractWorkstation, tWorkcentre* and *tDepartment* classes to represent different views of support functions in the same way as AFS describes a manufacturing system. Re-using these classes encourages a more consistent interpretation of the manufacturing and non-manufacturing working environments within a business.

A new *tDesk* class has also been designed as an additional type of workstation to provide a location in the model where administrative activities will be carried out. This class has intentionally been designed to be much simpler than its shop-floor equivalent. Unlike a *tManualMachine*, for example, a *tDesk* object represents a 'passive' entity in the sense that it cannot be 'idle', 'running' or 'broken'-down'; it merely provides a location for person objects to process documents and so aptly represents similar entities in real office systems. Three document storage areas are also attributed to each *tDesk* object; an *InTray*, *OutTray* and *PendingTray*. In fact storage areas are attributed to all office-related work-place objects. During a simulation these will contain documents in varying states of completion and will provide interfaces to work-place objects whereby documents can be moved between different locations within an organisation.

10.3.2 Transport System Classes

In modelling terms, therefore, another important aspect of any business environment will be different kinds of transport or conveyance system that move either materials or documents between work-place objects. In fact, the object-oriented analysis established a range of classes for representing this aspect of an organisation in terms of objects. Again, for consistency, the new WBS/Office classes have been designed to integrate with AFS's own range of transport classes that are specifically intended for modelling the material-handling activities within a shop-floor environment. Both existing and new classes are illustrated inFigure 10-3.

The illustration includes a new *tForkLift* class; since the analysis of stores areas established the need to model fork-lifts and hand-trucks as alternative material handling devices, this new class has been designed to compliment AFS's existing range of transport devices. This enhances the system's ability to evaluate the impact of

alternative handling devices on the performance of a goods inwards or despatch function, and consequently their effect on manufacturing system performance.

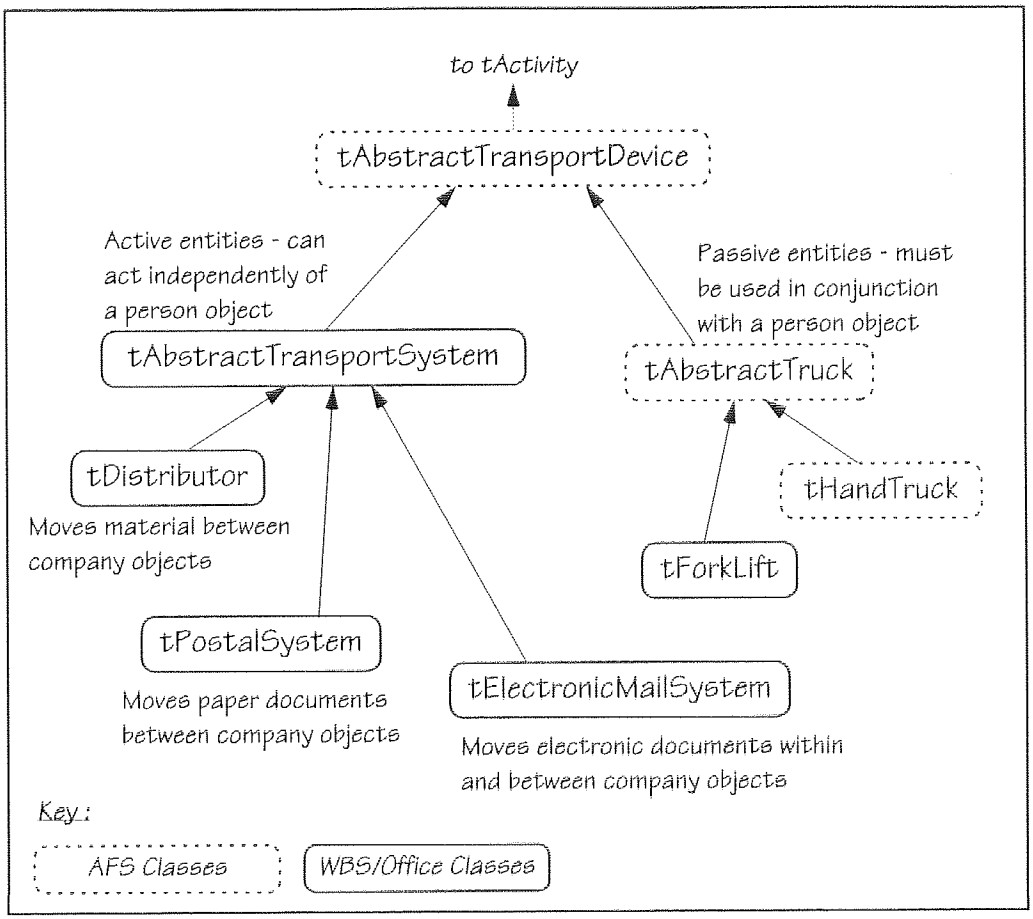


Figure 10-3 Material and Document Transport Classes

This new *tForkLift* class and the existing transport device classes represent 'passive' entities in as much as they are designed only to be used in conjunction with a more 'active' or intelligent person object. In this respect, the other WBS/Office transport classes shown in Figure 10-3 are fundamentally different since they represent more aggregate transport 'systems' rather than transport devices, and encapsulate enough know-how or intelligence to move material independently of person objects.

The *tDistributor* and *tPostalSystem* classes in particular represent aggregate objects in the sense that they transfer materials and paper documents respectively, between the company being modelled and its suppliers and customers. The *tElectronicMailSystem* provides essentially the same functionality except, as its name suggests, it transfers models of electronic documents between different companies, as well as between work-places within the same organisation.

All three of these classes correspond to objects identified in the analysis phase and are intentionally quite simple, providing only basic functionality and logic for moving materials or documents between different types of workplace. Nevertheless, they all contribute to providing a simulator that can model the same kinds of delay incurred by the real systems that these classes represent. For example, the *tElectronicMailSystem* mimics instant transfers of documents between work-place objects while the *tDistributor* and *tPostalSystem* classes simulate finite transfer delays that are configured by the model-builder. The style and naming of these transport classes also helps to ensure that users can construct models using concepts that relate closely to their perception of a real manufacturing business.

Importantly, none of the transport classes described so far can represent the movement of physical or paper documents within a business. In fact, the analysis and design of this aspect of WBS/Office is conceptually quite different from the other transport classes. More specifically, the logic needed to simulate this aspect of an administrative system has been designed as a new mailing skill that can be attributed to people objects in the model and which is intended to be used independently of any other transport devices or systems. Consequently, it has not been included in Figure 10-3. Importantly, however, from a design point of view much of the basic transport logic in WBS/Office is shared by this new skill and the other transport classes discussed here. This logic will be described in more detail later in the context of people-related classes and the new mailing skill.

10.4 Document-Related Classes

The object-oriented analysis also identified a range of objects that describe the administrative aspects of support department work. More specifically, it established a very flexible way of representing the different types of information and documents processed in manufacturing organisations in terms of objects.

10.4.1 Modelling Information

The analysis emphasised that information processing requirements vary and so the classes that represent this aspect of business activity must be flexible enough to describe the actual types of information used in a range of firms. Specific information

types, or *DataObject* classes, therefore need to be included in WBS/Office to represent the range of information used in real manufacturing companies and to also embed some knowledge of what different items of information in the model actually represent. The analysis established four general categories of information; *dates*, *names*, *numeric data* and *authorisations.*

To implement the *DataObject* concept in an absolutely object-oriented fashion would involve developing an extensive hierarchy of classes, based on these four categories. Figure 10-4, for example, illustrates how this type of hierarchy might be designed and includes a sample *DataObject* to demonstrate the basic structure of an instance of one of these classes.

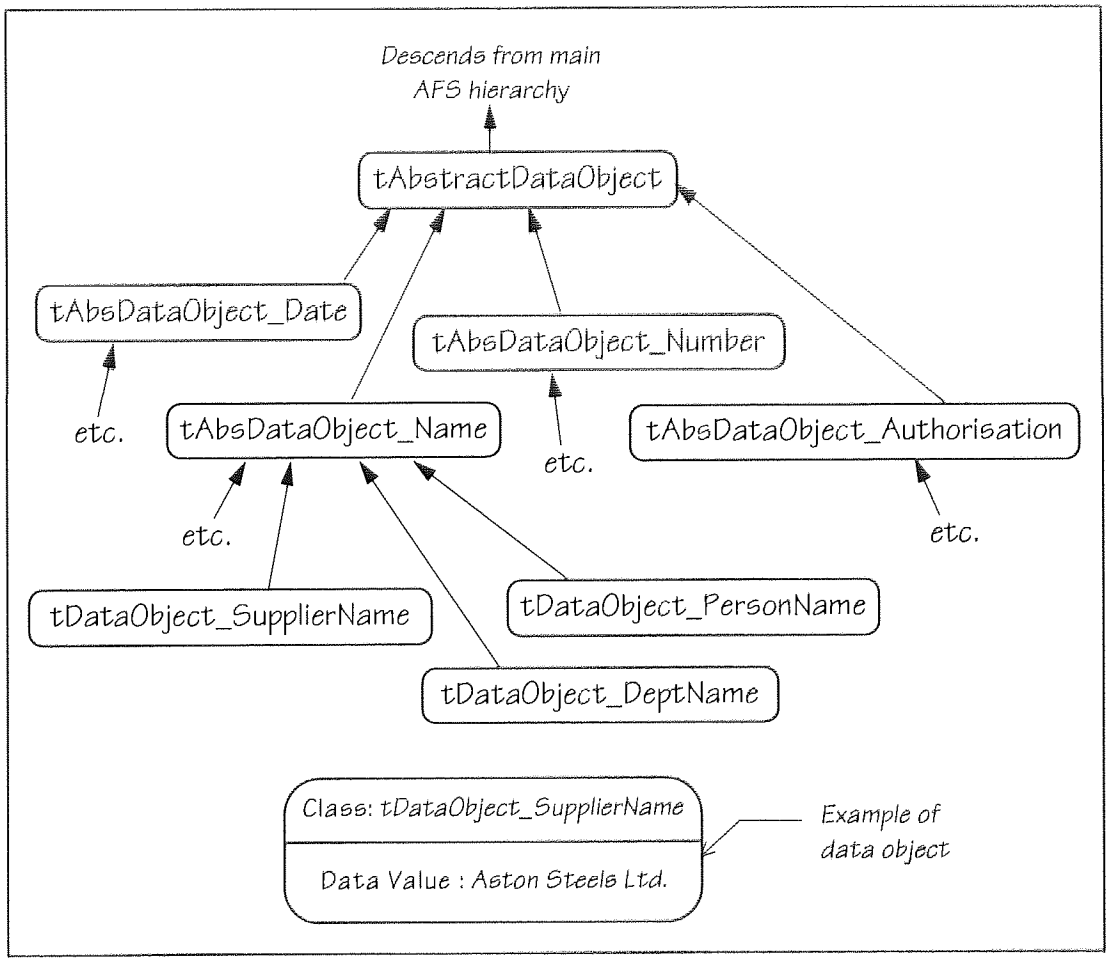


Figure 10-4 Defining Information Types in Terms of a Large Data Object Hierarchy

The illustration shows that from an *tAbstractDataObject* class might descend four more abstract classes that represent the four information categories. In turn these general classes would be inherited by an assortment of more specific information classes. For clarity, Figure 10-4 includes only a few of these specific classes

representing the names of suppliers, departments and people. Invariably, however, this approach would actually require a very large and complex sub-hierarchy within the main AFS class hierarchy. It would also mean that each information object would be quite large and complex since it will encompass several layers of abstract data object classes. This complexity would also have to be repeated in the class manager hierarchy that was briefly mentioned earlier. In practice, however, since many instances of these *DataObject* classes will be created as part of any support department model, each object should be as simple and as compact as possible. A more practical design is therefore illustrated inFigure 10-5.

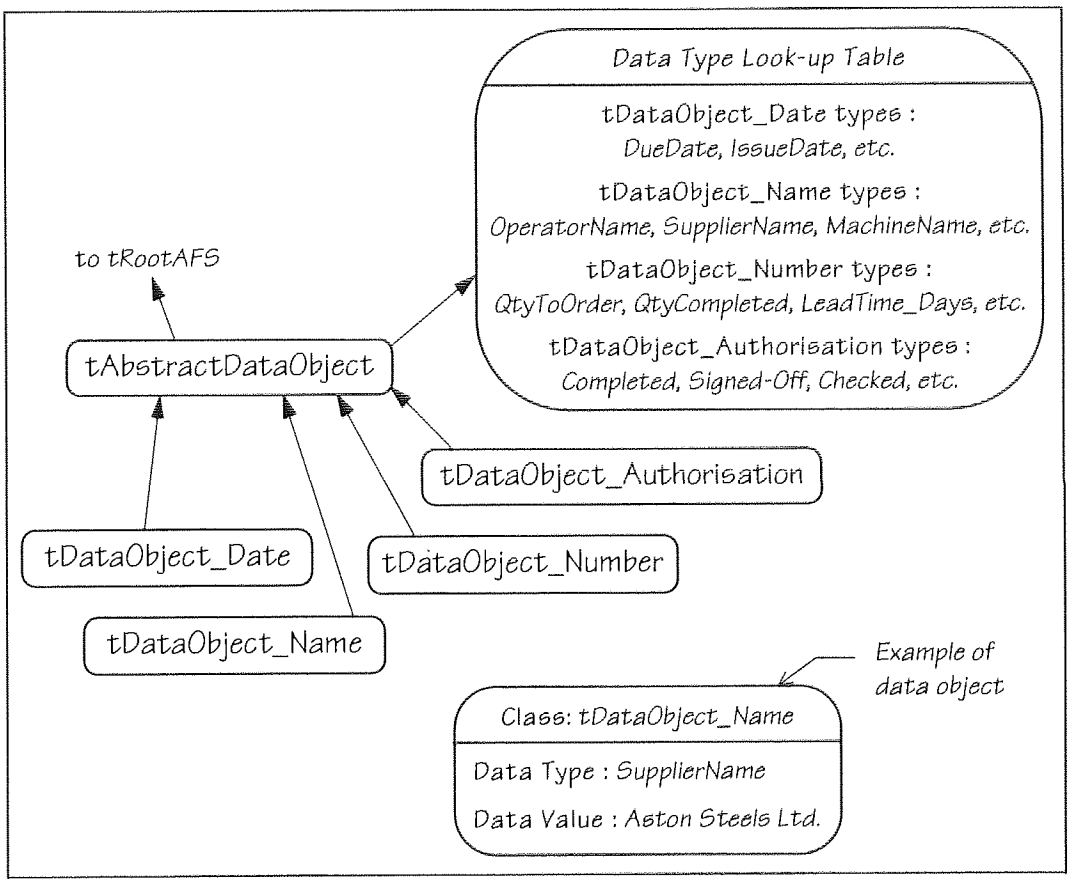


Figure 10-5 A More Practical Approach to Implementing Information Types

This design restricts the number of *DataObject* classes to just those that represent the four main information categories. Instances of these more general classes are differentiated by specific data-types attributed to each class. These types are maintained in a single look-up table that is referenced by all of the information classes. An example data object is also included in Figure 10-5 to demonstrate this alternative class structure. While this design requires each object to maintain an additional data-type attribute, the overall size and complexity of each object will actually be greatly

reduced since the overhead imposed by several layers of abstract parent classes is avoided.

A potential limitation of this design is that any range of data-types identified by the developer will not be enough to represent every type of document found in a manufacturing business. This is particularly significant, for example, in the context of the Whole Business Simulator that was mentioned earlier since this system will need to model financial information which has not yet been included in the new simulator.

WBS/Office has, however, also been designed to minimise the impact of the developer not having included not enough different *DataObject* types. More specifically, since the four *DataObject* classes included in the design cover the full range of information categories, developers can implement a new data-type simply by declaring it in the global look-up table. Referring to the table in Figure 10-5, for example, a new data-type such as *PartName* could simply be added to the list of *tDataObject_Name* types to extend the range of information supported by WBS/Office. A developer may also need to add some simple logic to other WBS/Office classes to interpret this new type of information, but this process has also be simplified by careful design and will be described later.

Overall, this particular design provides a practical but flexible way of incorporating different types of information into models of administrative activities. It should be noted that the examples presented in this section represent only a small selection of the full range of *DataObject* types actually included in the design of WBS/Office. A complete listing is therefore provided in Appendix 3. The next section will describe the context in which this document-based information will be generated and moved around a model of a real business.

10.4.2 Modelling the Flow of Information

The concept of *DocumentTemplate* and *Document* objects was established earlier to describe the medium by which information is carried through a manufacturing organisation. In addition, the notion of *DocumentRouting* and *DocumentOperation* objects was introduced to explain exactly how this data is processed and then transferred as a *Document* between different functional areas. The analysis also drew

parallels between administrative and shop-floor processes. As a result, the document-related classes in WBS/Office have been designed to match as closely as possible the way part-related classes are already implemented in AFS, while still providing other features specifically needed to model administrative tasks. This encourages a more consistent interpretation of the way manufacturing and support activities are represented in WBS/Office, which is beneficial from both a development and model-building point of view.

Figure 10-6, for example, illustrates how some of these new document-related classes have been integrated into the existing AFS architecture and also shows the corresponding part-related classes. As with other WBS/Office classes, the hierarchy can only represent type-of relationships and so other part-of relationships have had to be defined within each class by the developer.

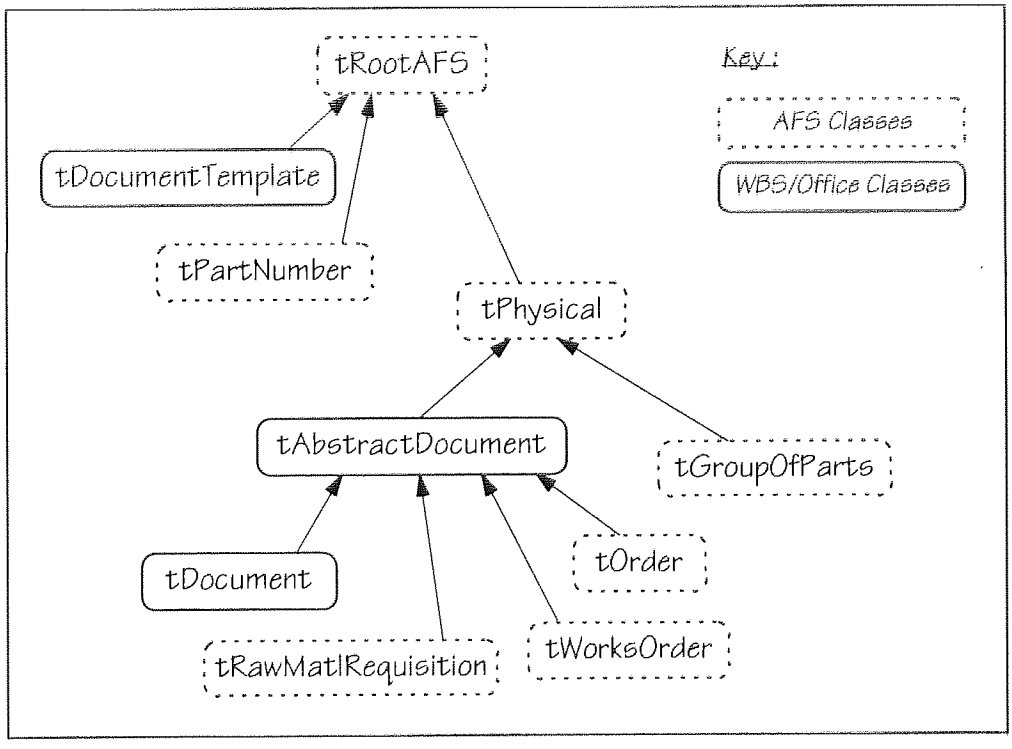


Figure 10-6 Document-Related Classes in WBS/Office

To model a manufacturing facility, for example, model-builders define the different types of part involved in terms of *tPartNumber* objects. This class describes the attributes of each type of part in the same way as a *tDocumentTemplate* object describes the structure of each type of document. During a simulation, the system will generate *tGroupOfParts* objects as physical examples of these parts in the same way as

tDocument objects are created as examples of their respective template object. Consequently, both of these classes must also inherit the properties of the abstract *tPhysical* class in order to have a physical presence or location in the model.

An important conceptual difference between *tPart* and *tDocument* classes is that the latter is intended to represent documents that exist in either electronic form or as physical pieces of paper. Clearly there is no equivalent in terms of shop-floor parts, but in terms of modelling administrative activities these two information mediums differ only in how a document moves from one part of the business to another. In other words, whether a purchase order exists in paper or electronic form, it will still carry specific types of information that will need to be acquired and processed in order for a manufacturing system to be able to operate. The design of the *tDocumentTemplate* and *tDocument* classes is therefore such that they can be used to model both forms of document.

Another aspect of the design relates to three document-type classes already included in the original AFS architecture, namely *tOrder*, *tWorksOrder* and *tRawMatlRequisition*, that carry specific production-related information needed by entities in the factory model. These represent examples of the kind of document that would be produced by support departments in a real business and should therefore be modelled as if they were produced this way. For example, the *tOrder* class essentially represents customer orders that determine what the factory model will produce. Within AFS, however, instances of this class have to be created by the user at model-build time rather than being generated as part of the simulation. As a result, the kind of delays that would be involved in producing this kind of information in a real business and then passing it to the shop-floor cannot be simulated in a conventional AFS model.

Ideally these AFS classes should not appear at all in WBS/Office since they effectively hard-code specific types of business document into models of support functions. Nevertheless, they have had to remain part of the new simulator because other shop-floor related entities in the original AFS architecture expect them to exist. All three original classes have, however, been completely re-designed and converted to the new WBS/Office document structure. In other words, by inheriting the new *tAbstractDocument* class their composition, in terms of data types, is now defined by

three corresponding *tDocumentTemplate* objects rather than being hard-coded into each document class. In fact, WBS/Office has been designed to automatically generate three default *tDocumentTemplate* objects that contain exactly the same types of information that were attributed to the three original AFS document classes.

This now means that delays incurred while creating and processing these specific types of document can be modelled in the same way as other types of business document will be modelled in WBS/Office. These classes also provide a strong link between the manufacturing and support department models because the progress of an AFS factory model is already affected by the information contained on these documents. Interestingly, the conversion from AFS to WBS/Office-style documents was actually quite easy to complete given the range of *DataObjects* available. The conversion process therefore demonstrates the flexibility of the approach to modelling documents that has been adopted in WBS/Office.

Figure 10-7 illustrates some of the other important WBS/Office classes needed to model the flow of information. More specifically, it includes the *tDocumentRouting* and *tDocumentOperation* classes that represent how different types of document are actually processed and consequently the kinds of delays that might be incurred.

The diagram also emphasises the similarities between the design of these classes and their shop-floor equivalent. As well as encouraging a more consistent interpretation of manufacturing and support activities, this similarity means that some of the existing logic for modelling shop-floor areas can be re-used for modelling office systems. In addition, however, the object-oriented analysis described earlier also established many conceptual differences between administrative and shop-floor work, and so the structures of these new classes have also been designed to fulfil other requirements. In effect, a *tDocumentRouting* describes a collection of *tDocumentOperation* objects, each of which in turn represents an explicit set of instructions for processing a particular type of document, or *tDocumentTemplate*. In other words, each *tDocumentOperation* object will comprise a list of sub-tasks or *OperationConfiguration* objects that define how each piece of information required for that stage will be obtained.

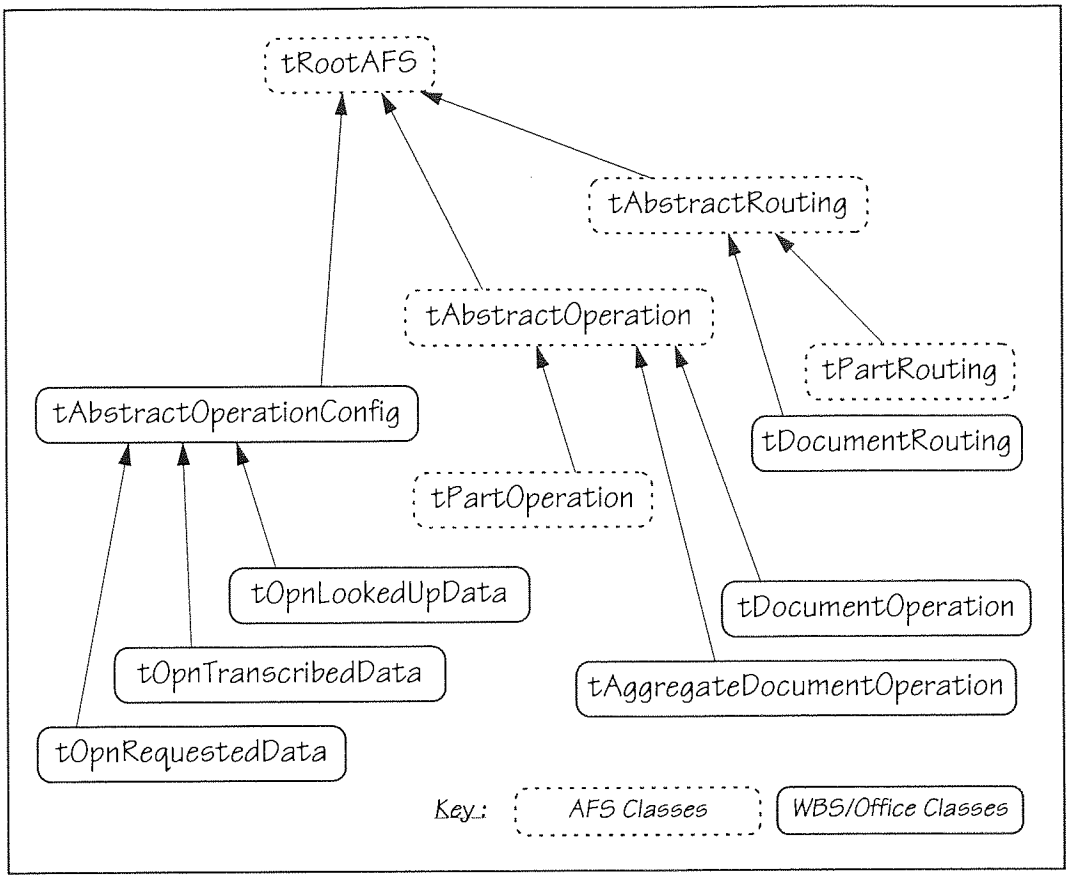


Figure 10-7 Information-Flow Classes in WBS/Office

The object-oriented analysis established three likely sources of information needed to process documents were identified earlier; transcribed, requested and look-up data. They have been implemented in WBS/Office as three *OperationConfiguration* classes; *tTranscribedDataConfign*, *tRequestDataConfign* and *tLookedUpDataConfign*. More specific examples of how these classes differ from a modelling point of view will be described later in relation to the implementation of WBS/Office.

10.5 People and Skills

In order to actually simulate the processing of documents and to properly represent real support departments, people need to be introduced into models of these systems. The object-oriented analysis established that, in modelling terms at least, people differ only in relation to the kinds of jobs they are able or expected to carry out. This applies to people working throughout a manufacturing organisation, whether they are shop-floor or support department employees.

10.5.1 Document-Related Skills

AFS already models people involved in shop-floor activities as instances of a *tPerson* class, and allows model-builders to attribute a range of different manufacturing skills to these *tPerson* objects. These skills have themselves been implemented as objects, such as *tOperateSingleMachine*, *tRepairMachine*, *tInpsection* and so on. As a result, this same *tPerson* class can be re-used in WBS/Office, in conjunction with new skill classes designed specifically to reflect a person's ability to process and move documents around a business.

During the object-oriented analysis, in fact, the concept of a generic *OfficeSkill* was introduced that enables *tPerson* objects to interpret specific *tDocumentOperation* instructions. This same idea has been incorporated in WBS/Office, in the form of a new *tOfficeSkill* class. Importantly, however, the functionality of this new skill has actually been split into three sub-classes, namely *tOfficeSkillConfiguration*, *tOfficeSkillLogic* and *tOfficeSkillProcess*. In other words, these three skill elements collectively form part-of an *tOfficeSkill* object. The relationships between these classes are illustrated in Figure 10-8. It should be noted that this aspect of a *tPerson* object will still appear to model-builders as a single office skill object, even though the object is managed internally by separate sub-classes.

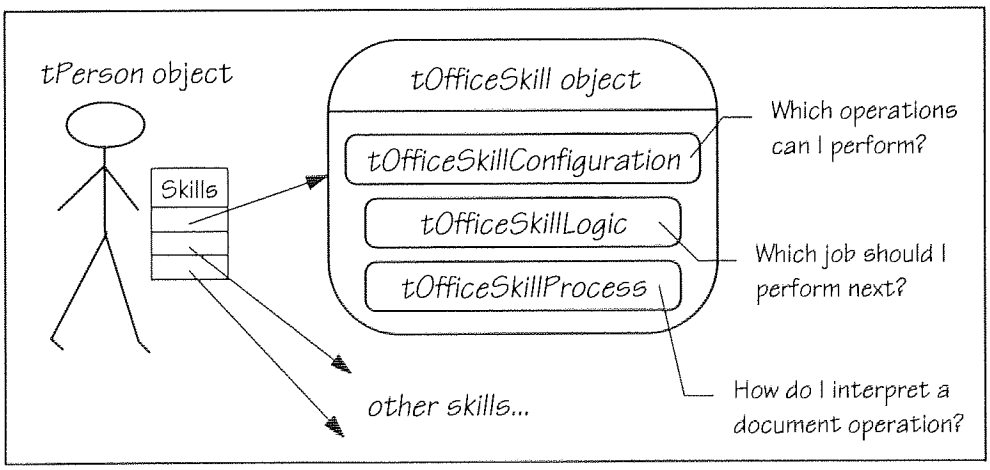


Figure 10-8 The Structure of Office Skill Component Classes

The reason for sub-dividing the new *tOfficeSkill* class in this way is to enable the same skill components to be re-used by other entities in WBS/Office. More specifically, the concept of aggregate company objects such as suppliers and customers was introduced earlier. These classes need to encapsulate limited versions of the same logic used by

tPerson objects to process documents and information in order to interact with support departments. Consequently, these office skill components have also been re-used in the design of the *tSupplier* and *tCustomer* classes, although model-builders will be unaware that these component classes even exist.

While a *tOfficeSkill* object provides *tPerson* objects with the responsibility for processing documents, the notion of a 'mailing' skill that was introduced earlier to allow *tPerson* objects to perform the task of moving *tDocument* objects between different areas within a model. This functionality is included in WBS/Office as a new *tDocumentTransportSkill* class that has been designed to simulate a person taking a particular route through the business, at set times of each day, collecting processed documents and delivering them to their respective destinations. An example of how this new skill might be used is illustrated inFigure 10-9.

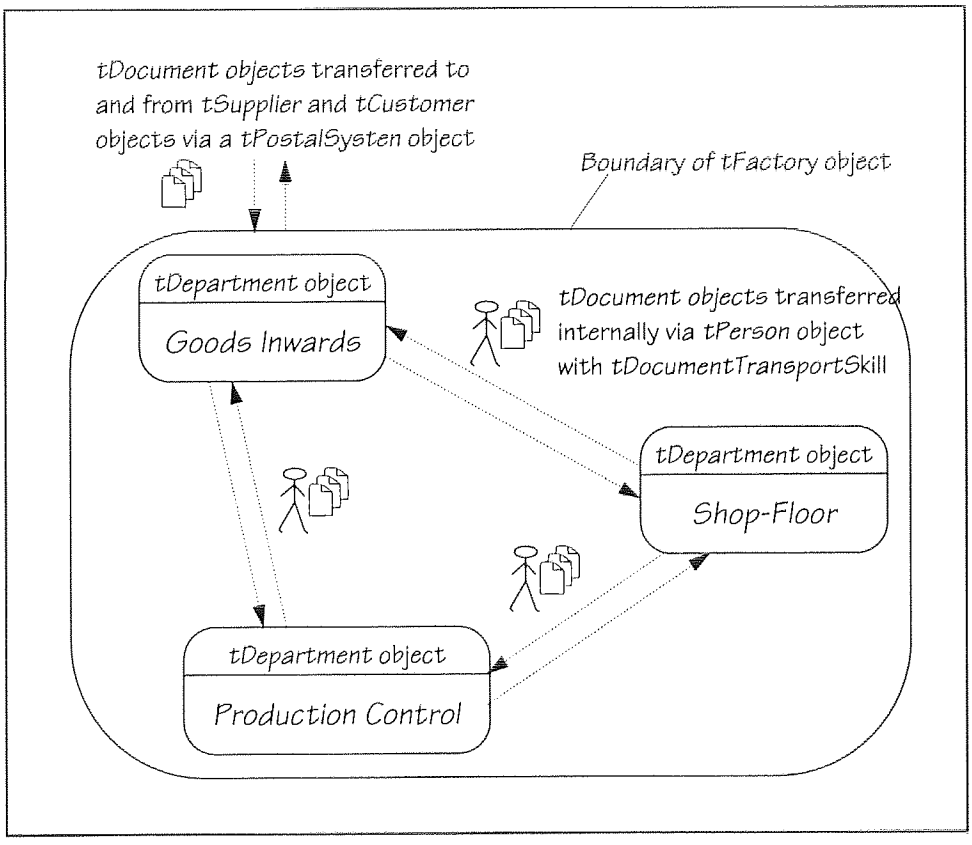


Figure 10-9 Example of Document Transport Skill

Like the office skill, the functionality of the *tDocumentTransportSkill* has been sub-divided into component skills in the form of *tDcmtTransportConfiguration*, *tDcmtTransportLogic* and *tDcmtTranportHandling* classes, as illustrated in Figure 10-10. Again the design is such that model-builders need not even be unaware that these

lower-level classes even exist, but by breaking down this functionality into smaller objects other WBS/Office classes can re-use aspects of the same logic and functionality. For example, the aggregate *tPostalSystem* and *tElectronicMailSystem* classes that were described earlier also encapsulate limited versions of this functionality.

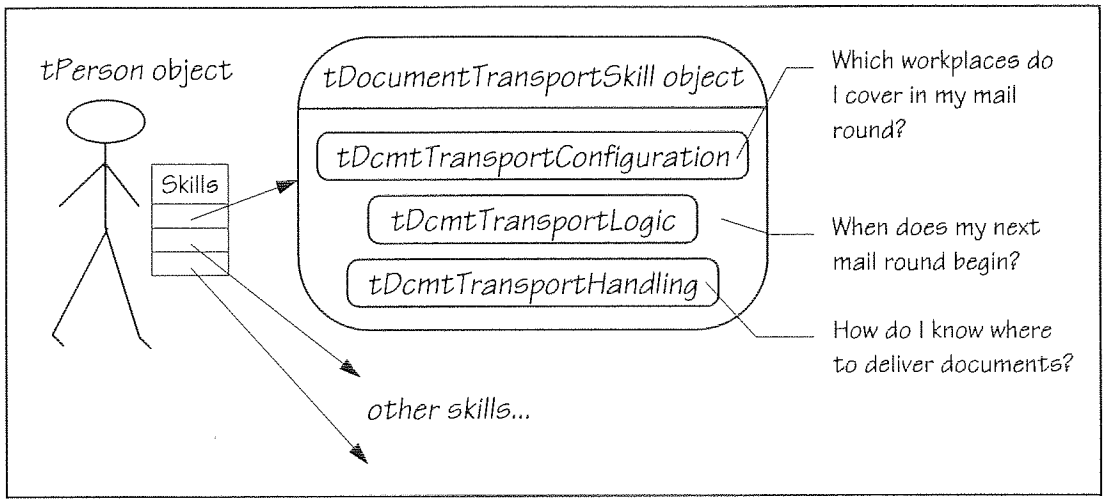


Figure 10-10 Structure of the Document Transport Skill

Finally, the notion of a 'creative' or *tMiscellaneousSkill* has also been designed. This can be attributed to *tPerson* objects in addition to other skills to simulate regular or irregular patterns of behaviour, whereby a person is temporarily busy or otherwise unavailable to complete their administrative duties. In practice this skill is likely to be particularly useful for modelling the availability of managers or team-leaders to sign-off orders or similar documents. The structure of the class is intentionally quite simple. Using parameters defined by a model-builder, this class can force a *tPerson* object to enter an 'unavailable' state for variable periods of time during a simulation. The user can define stochastic behaviour patterns in terms of a number of minutes per day or per week when that person is unavailable.

In summary, the generic *tOfficeSkill* and *tDocumentTransportSkill* classes, and the *tMiscellaneousSkill* class, combine to provide a way for model-builders to represent the different personnel involved in carrying out a firm's administrative duties. In other words, these new classes reflect a person's ability to process and move documents around a business. The next section describes the design of other WBS/Office skill classes.

10.5.2 Document Action Skills

One of the main purposes of this thesis is to develop a way of modelling the interactions between a manufacturing system and its support functions. Closely related to this principle is the notion that documents produced in office areas are ultimately intended to trigger some kind of action on the shop-floor or in other non-administrative areas, actions that will invariably affect manufacturing system performance. In fact, the object-oriented analysis has already emphasised that a way of generating actions from completed *tDocument* objects must be included in the WBS/Office system.

Again, however, the main problem from a development point of view is to provide a generic solution since document types will only be defined by users at model-build time. Nevertheless, the range of data-types or *DataObjects* from which model-builders can configure document types (or *tDocumentTemplate* objects) will always be limited to those included in WBS/Office by the developer at design time. Consequently, the ability to establish what action a user-defined document might imply during a simulation can be linked at design-time to what sets or types of data are likely to be contained on *tDocument* objects. In addition, the context or environment in which different types of data might be used can help in determining the kind of action that any document in a real business containing this information is likely to represent.

For example, if a *tDocument* that contains a *PartNumber* and *QtyToOrder* is received by a *tPerson* object in a manufacturing cell, it suggests that the document is an instruction to manufacture that quantity of the given part. In other words, the document represents some kind of works order. On the other hand, if a *tDocument* carrying the same types of information were received by a *tSupplier* object it is more likely to represent a purchase order, or in effect, an instruction to supply that number of the given part. Similarly, it can be assumed that a *tPerson* in a stores function receiving this same information would most likely interpret it as having to issue that number of parts from stores to the shop-floor.

Designing a flexible way of attributing this kind of 'intelligence' to existing AFS classes and to new WBS/Office classes will provide a natural link between the manufacturing and non-manufacturing elements of a model. AFS already represents

the different types of people that would be expected to act upon these documents in a real business by attributing specific shop-floor skills to *tPerson* objects. Consequently, extending the functionality of these existing skill classes would enable them to interpret the contents of a *tDocument* object during a simulation, and trigger an appropriate action that a person with that particular skill might typically perform in a real business. For this reason, a range of *DocumentAction* classes have been designed that can be re-used and attributed to existing skill objects and, indeed, to other aggregate classes such as *tCustomer* and *tSupplier* that also need to be able interpret WBS/Office's generic document style.

These new classes have been designed as 'skill components' to be used by other skill classes rather than being attributed to *tPerson* objects directly. This means that users will not be aware that these additional classes even exist, thereby simplifying the model-building process. Essentially each action skill component encapsulates a form of 'intelligence' or logic that can determine what a person or other entity should do as result of receiving a completed document containing particular types of information. An example of how a *DocumentAction* component skill might be used to enhance the functionality of an existing AFS skill is illustrated inFigure 10-11.

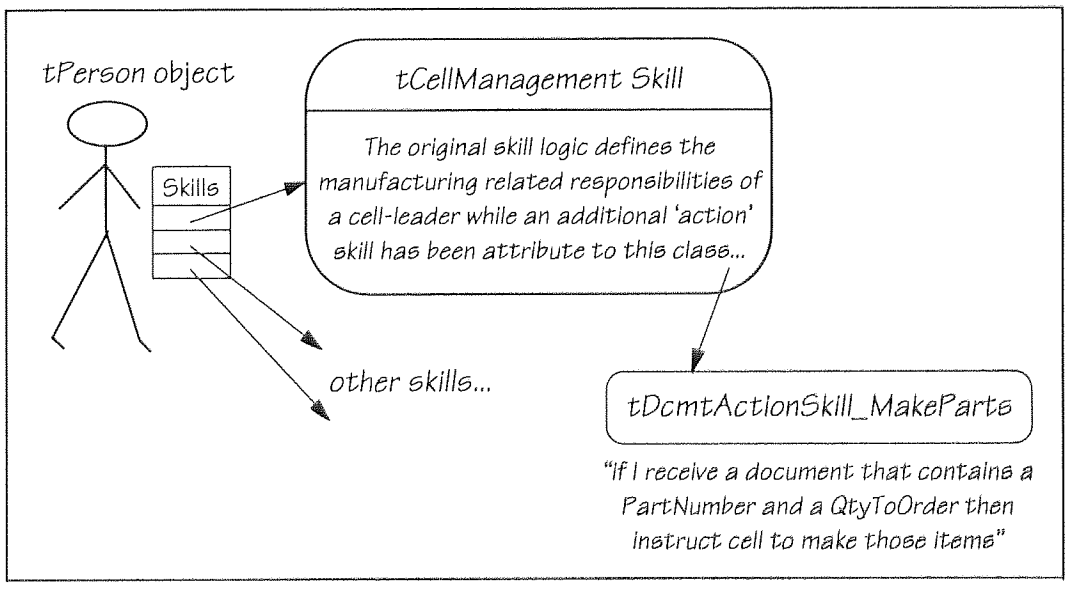


Figure 10-11 Example of a Document Action Skill

The reason for designing this aspect of WBS/Office in terms of 'mini-action skills' is two-fold. Firstly, by categorising different aspects of the logic or 'intelligence' into self-contained classes, the same functionality can be re-used in the design of various

skills and other parts of the simulator. Secondly, the idea of constructing this aspect of WBS/Office from a pool of mini-action classes provides more opportunity for extending the functionality of the simulator at some point in the future.

For example, it was mentioned earlier that a potential limitation of WBS/Office is that the range of *DataObject* types identified by a developer may not be enough to represent every type of document found in a manufacturing business, and so this aspect of the simulator has been designed such that it can be easily extended to include new *DataObject* types. The generic nature of the *tOfficeSkill* class is such that it will not need to be modified to cater for new information types. Other classes, however, that will be expected to generate some kind of action as a result of receiving a document that contains this information will need to be modified accordingly. Consequently, having already implemented similar functionality in the form of a pool of specific *DocumentAction* classes, further action classes can simply be created by a developer to interpret documents containing any new types of information or *DataObjects*.

10.6 Linking to WBS/Control

It was emphasised earlier that since a production planning and control system represents a specialist type of support activity it should also be included in a model of a firm's support functions, and in fact, WBS/Control already provides a library of simulation classes for modelling different types of control system. Consequently, another important aspect of the object-oriented design described here relates to how these control system classes will be incorporated into WBS/Office.

In modelling terms, many aspects of WBS/Control are conceptually quite different to the features of AFS and WBS/Office that have been described so far. For example, the classes that make-up WBS/Control essentially correspond to 'components' of knowledge or information systems that represent the different aspects of real control systems, such as bills-of-materials, master production schedules and so on. Model-builders can configure these components together to represent a variety of different production control scenarios. This is quite different to AFS and WBS/ Office which both represent the rather more 'material' aspects of a manufacturing organisation, the term 'material' in this context referring to entities such as parts and documents that

have a physical presence in a real business.

From a WBS/Office point of view, WBS/Control classes can therefore be used to simulate different production control configurations, to show how the different types of production control information generated by these systems can affect manufacturing performance. Other classes within WBS/Office can then be used to enhance this feature of the simulator by modelling the delays incurred while processing and transferring this information to and from shop-floor areas. This can be achieved by replacing WBS/Control's own document-type classes with the more generic WBS/Office type classes, as illustrated inFigure 10-12.

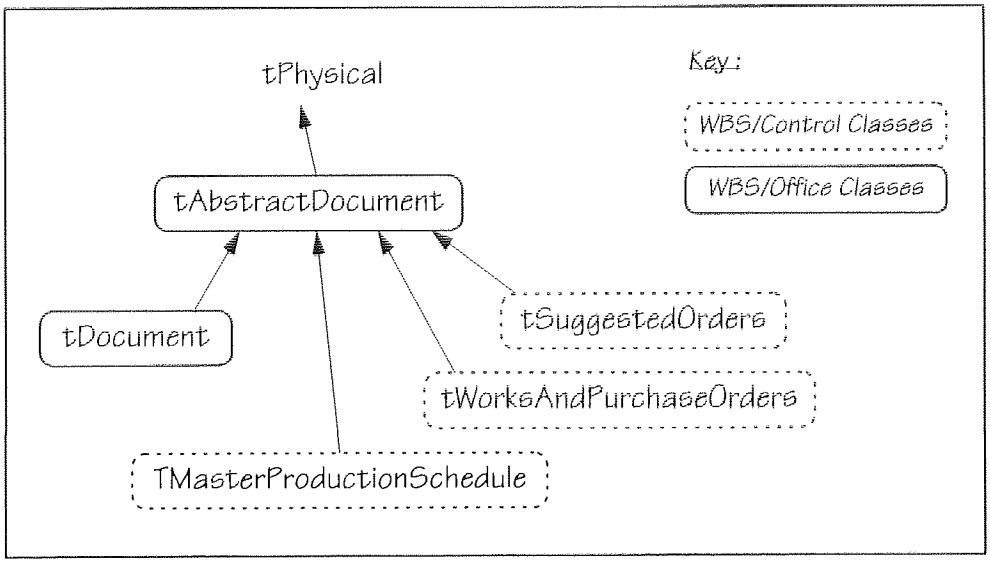


Figure 10-12 WBS/Control Information Classes

Ideally the WBS/Control classes shown in Figure 10-12 should not appear at all in WBS/Office since they effectively hard-code specific types of document into models of support functions. Nevertheless, they have had to remain part of the new simulator because other classes in the original WBS/Control architecture expect them to exist. All three original classes have, however, been completely re-designed and converted to the new WBS/Office document structure.

Finally, the object-oriented analysis of support functions also established a need to model resources in goods inwards or despatch areas, in order to evaluate their impact on manufacturing system performance. Designs for most of the new classes needed to provide this functionality in WBS/Office have already been described; in other words, the *tPerson*, *tDistributor*, *tForkLift* classes and so on. However, a new skill class also

needs to be defined to provide *tPerson* objects with the know-how and authority to work in this type of environment.

It was established earlier that this type of work will involve a combination of material-handling and administrative duties. WBS/Control already includes its own *tStorePerson* skill that enables *tPerson* objects to update the inventory records that will be used during the production planning cycle. Any other administrative responsibilities associated with a store person can therefore be modelled by attributing a *tOfficeSkill* object to that *tPerson* object and relating it to whatever document-processing tasks are performed in a particular stores function. To also model the material-handling aspects of this type of work, however, the original *tStorePerson* class has had to be modified to also include elements of the functionality already included in AFS for simulating material-handling tasks within a manufacturing facility. As a result, a model-builder can attribute instances of both this new *tStorePerson* class and the configurable *tOfficeSkill* class to represent personnel employed in stores areas.

This section completes the design of the WBS/Office class architecture that extends the functionality of the existing core simulator. Notably, discussion of a user-interface that would allow model-builders to interact with this extended class library has be very limited so far. Nevertheless, the next chapter describes the implementation of WBS/Office and will therefore describe, in more detail, features of the user-interface needed to model support functions.

10.7 Summary

This chapter has described the design of WBS/Office, an object-oriented simulator that fulfils previously established requirements for a system that can model both the manufacturing and support activities within an organisation. The design has involved the creation of a range of new classes that extend the functionality of an existing library of simulation classes, namely the Advanced Factory Simulator (AFS) and WBS/Control. Indeed, the design has intentionally re-used as many features of the existing AFS and WBS/Control system architectures as possible to encourage a more consistent interpretation of the way manufacturing and support activities are represented in WBS/Office. This is beneficial from both a development and model-

building point of view.

The next chapter will describe how this overall design has been implemented in a software development language. In a particular, the discussion will explain features of WBS/Office's user-interface that allow model-builders to easily construct simulation models using this extended class library.

11. The Implementation and Application of WBS/Office

Preceding chapters have described the object-oriented design of WBS/Office in terms of a library of new simulation classes. This chapter will describe the issues involved in implementing the design to provide a usable data-driven simulator. The discussion will relate to the choice of programming languages used to implement WBS/Office, as well as features of the system's user-interface from the point of view of how a user would construct a simulation model. Example models that have been created using the new system will also be presented to demonstrate the feasibility of including both shop-floor and support activities in a discrete-event simulation model.

11.1 The Software Development Environment

The choice of programming language and, in particular, limitations imposed by the target operating system have had implications for the way WBS/Office has been implemented. What follows is an overview of these aspects of the new simulator.

11.1.1 The Choice of Programming Languages

It was mentioned earlier that AFS has been implemented as three separate applications or 'programs' that interact with each other to produce the overall system; these elements are referred to as the *core simulator*, the *graphical display* and the *user-interface*. Some developers provide 'hooks' into their programs to allow third-party developers to add extra functionality to an application without modifying or having access to the original source code. The graphical display, for example, has been designed such that its features can be accessed from within the core simulator and so the graphical part of AFS has not had to be modified to accommodate WBS/Office. In contrast, both the core simulator and user-interface in AFS have been designed in such as way that adding new functionality actually involves replacing the original programs with re-compiled versions of the same source code that has been modified to include any new classes.

In practice, this approach allows future developers to have more control over how their new classes will fit into the existing architecture of both programs, and also makes it

easier for any new classes to themselves be further developed at a later point in time. It also means, however, that any new functionality such as that provided by WBS/Office must be implemented using the same programming languages originally used to construct these applications. For this reason, WBS/Office has been implemented using Borland's *Object Pascal* and Microsoft's *Visual Basic* by virtue of the fact that the core simulator and user-interface were originally implemented in AFS using these particular languages (full references for all commercial software used in the development of WBS/Office are provided in Appendix 5).

In fact, the original implementation of AFS was completed in 1994 (Ball, 1994). At that time the IBM-compatible personal computer (PC) was already established as the standard desktop computer, and *Microsoft Windows 3.1* (referred to hereafter as *Windows*) was recognised as the standard operating system that would be used on this type of PC. For this reason *Object Pascal* was chosen at that time as an appropriate programming language for implementing AFS as an object-oriented application that would run under *Windows*, while *Visual Basic* enabled the application to also include the now familiar *Windows*-style graphical user interface.

11.1.2 Overview of the Core Simulator

Both AFS and WBS/Office have been designed as object-oriented simulators. Ideally an object-oriented programming language should be used to implement an object-oriented design since the constructs of the language will be very similar to the constructs of the original design (Rumbaugh et al., 1991). In this respect, Borland's *Object Pascal* supports object-oriented concepts such as abstraction, encapsulation and inheritance that were used extensively in the development of the WBS/Office class hierarchy. As a result, the language has provided a way of producing a core simulator that very closely matches the design described in preceding chapters; a summary of all the source code is provided in Appendix 1. In fact, the main difficulty encountered during the coding of WBS/Office arose from limitations imposed by *Windows*, the target operating system for the simulator, rather than from the translation of the object-oriented design into an object-oriented application.

Windows is a 16-bit operating system that uses a 'segmented memory architecture' to

manage access to a computer's memory. This means that the system divides memory into 64 kilobyte (or 64K) segments and restricts each application to including all its global variables and stack within a single segment. Consequently, the notion of a single memory segment, or *common data segment*, places a finite limit on the size and amount of data that can be handled by any application running on that operating system, regardless of how much physical memory is actually available on an individual computer (Richter, 1994).

As an expandable simulator, both the scope and size of the original AFS class library will potentially grow to be quite large. For this reason, many of the original data-types and data-structures used to construct AFS were specifically designed to make efficient use of the limited amount of memory made available to the application by *Windows*. Otherwise, if memory constraints prevented the construction of large models or expansion of the system's functionality, the simulator might not be appropriate for solving real manufacturing system design problems. This design consideration has meant, for example, that WBS/Control (Boughton, 1995), which represents an extended version of AFS's original functionality, has been successfully implemented to operate within this common data segment limit and can still therefore be run as a *Windows* application.

The purpose of WBS/Office is to expand this functionality even further which invariably increases the application's overall size and places additional data into its allocated common data segment. Consequently, the risk of the simulator exceeding the memory limitations imposed by *Windows* will be increased in WBS/Office's case, particularly if large models are to be constructed. In fact, this problem was frequently encountered during the coding and testing of WBS/Office. As a result, several modifications were made to data-structures in some of the original AFS architecture in an attempt to include the full range of new classes in the simulator within the constraints imposed by the operating system. Nevertheless, while these modifications did enable a significant proportion of the object-oriented design described earlier to be implemented, it appears that the amount of additional functionality that WBS/Office proposes to offer is such that the simulator cannot be fully implemented as a 16-bit application.

Since the original implementation of AFS, 32-bit operating systems have become available to PC-users in the form of *Microsoft Windows 95* and *Microsoft Windows NT*. These use a non-segmented or 'flat memory architecture' that overcomes the common data segment limitation of the older *Windows* operating system (Richter, 1994), and therefore present an opportunity to implement the entire WBS/Office design to still run on a standard PC operating system. However, this improved memory model is only available to applications that have been specifically re-programmed to take advantage of the new 32-bit architecture. In practice, this means that a full implementation of WBS/Office would involve converting all of the existing *Object Pascal* source code to a 32-bit programming language to be re-compiled into a 32-bit application.

While updated compilers are now available for porting 16-bit *Object Pascal* source code to take advantage of the new 32-bit memory model, it has not been feasible to complete the conversion of the entire AFS and WBS/Control class library within the time frame imposed by this research project. Nevertheless, it has still been possible to implement a substantial proportion of the object-oriented design described earlier, albeit as a 16-bit *Windows* application. This provides an opportunity to demonstrate the feasibility of using discrete-event simulation to model both the material and information processing aspects of a manufacturing system design.

Specifically, the current implementation of WBS/Office does not incorporate the existing WBS/Control class library for modelling production control systems, or the range of classes designed to represent the non-administrative aspects of support department work, such as goods inwards handling. As a result, the conversion to a 32-bit application presents an opportunity for further research and development of WBS/Office to include this additional functionality. Further work will be discussed again later.

11.1.3 Overview of the Graphical User-interface

The user-interface represents an extremely important part of any data-driven simulator. In the case of WBS/Office, in particular, it must allow model-builders to easily interact with the core simulator to create instances of the classes maintained by that part of the

application. The relationships between the model-builder, the user-interface and the core simulator are illustrated inFigure 11-1.

The illustration emphasises that WBS/Office appears to end-users as a single application, even though it actually comprises several smaller programs. The user-interface, core simulator and graphical display programs communicate using DDE (Dynamic Data Exchange), a client/server protocol through which programs can exchange data and control other applications (Pfaffenberger & Wall, 1995). While hidden from the end-user, it is this generic data-transfer mechanism that enables model-builders to create, for example, an instance of a *tDesk* or *tSupplier* object in the core simulator.

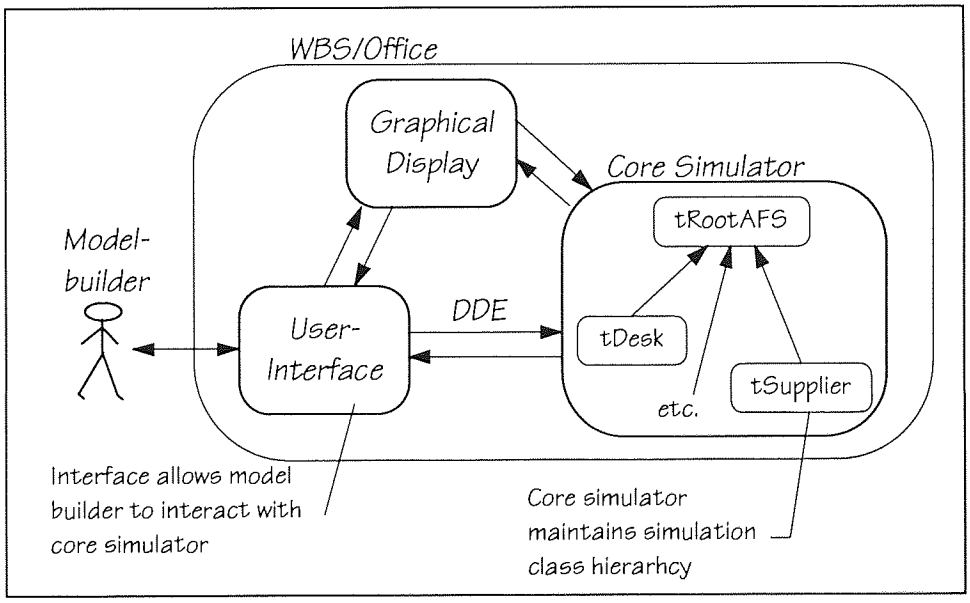


Figure 11-1 Relationships between the Model-builder, User-interface and the Core Simulator in WBS/Office

While *Object Pascal* provides a very suitable platform for implementing the WBS/Office core simulator as an object-oriented application, it is quite limited in terms of providing developers with an easy way of creating a graphical user-interface. Consequently, this aspect of the simulator has been implemented using another programming language, namely Microsoft's*Visual Basic.*

Although the particular version of *Visual Basic* originally used to code this aspect of AFS (and therefore WBS/Office) is not actually an object-oriented language, this part of the system has still been developed in an object-oriented style such that model-builders are presented with individual menus and dialogs that correspond to particular

classes in the core simulator. In fact, a variety of new dialogs or menus have been added to the original AFS interface to allow users to edit the attributes of new WBS/Office objects. Specific examples of these new dialogs will be illustrated later but in terms of their general design, a very important consideration was that end-users be presented with support department concepts that they readily understand.

Another consideration was that as much as possible of the interface be designed in such as way that it can be re-used by any new classes added to the system at some point in future without the need to change the interface. In other words, the different elements of the user-interface have been treated very much as objects in the same way as aspects of the core simulator are also defined as objects. For example, each user-interface dialog has been designed to have only minimal knowledge about the particular simulation class that it represents. This means, for instance, that new document data-types or even new data object classes can be added to WBS/Office's core simulator without having to modify the user-interface.

This completes the general overview of the new core simulator and user-interface that make-up WBS/Office. The remainder of this chapter will describe specific features of the user-interface in more detail from the point of view of how a manufacturing system engineer might use the system to construct a simulation model of the support activities in a manufacturing business. Example models will also be presented.

11.2 Creating Models in WBS/Office

This section will present the main features of WBS/Office's graphical user-interface that are involved in the creation of models to represent support functions in manufacturing organisations. In practice, models will also need to represent shop-floor and production control activities but the reader is referred to Ball (1994) and Boughton (1995) for descriptions of how the shop-floor and control system elements of a model are constructed.

It should also be noted that since WBS/Office is intended to allow these different aspects of a manufacturing organisation to be combined into a single model, the system does not distinguish between the shop-floor, control system and support department functionality. The user-interface only categorises the simulation objects that make up a model according to their class; in fact, to avoid presenting model-builders with

simulation or programming jargon, the interface actually uses the terms *functionality* and *entity* in place of class and object respectively. For example, the main model-building dialog which is illustrated in Figure 11-2 lists some of the main classes (or functionality) typically used in the construction of a WBS/Office model. The list includes classes representing manufacturing activities, such as batches and machines, as well as those representing non-manufacturing activities such as documents and desks.

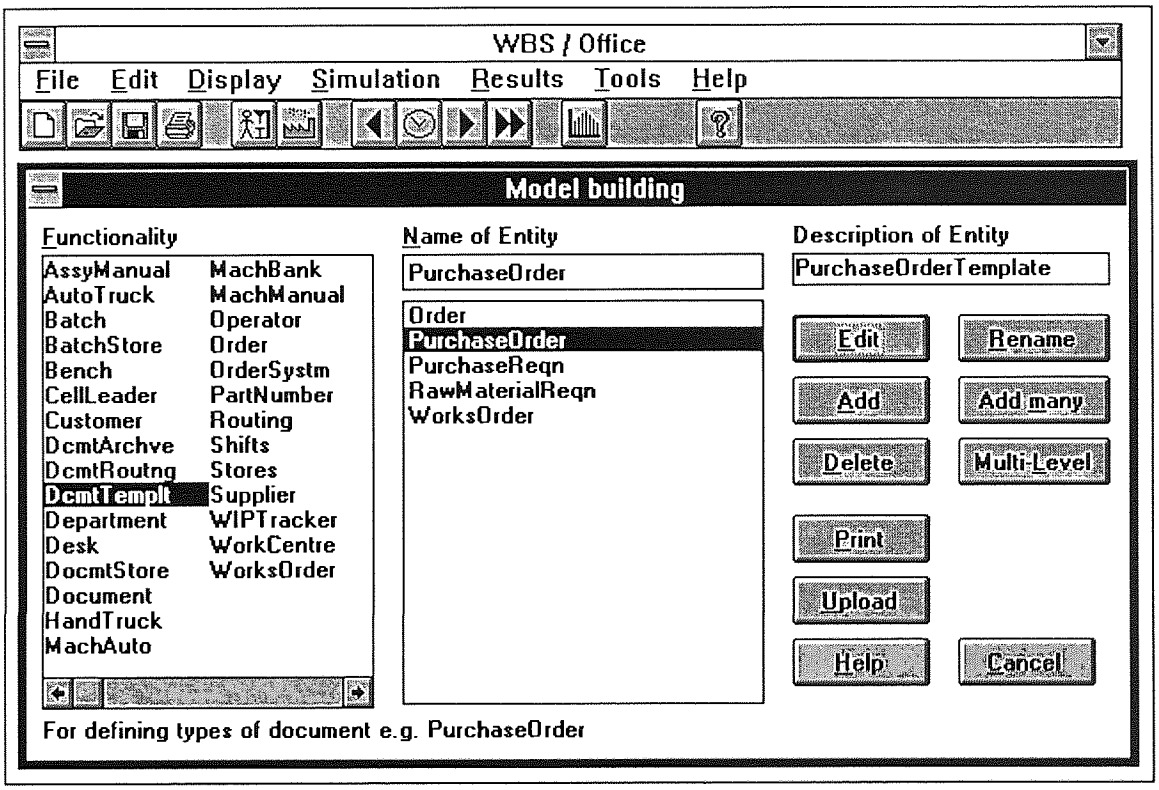


Figure 11-2 Main Model-Building Dialog

During the model-building process, a user will first select from this list a class of simulation object to be created or edited; the system will then list all instances of that class in the current model. In this case, for example, the user has selected the *DocumentTemplate* class and is presented with a list of all the document template objects (or entities) that are currently included in the model such as an order, a purchase order and so on. By selecting the *PurchaseOrder* template and then on clicking the 'Edit' or 'Add' button the user will be presented with another dialog that relates specifically to the *DocumentTemplate* class. Examples of these class-specific dialogs will now follow.

11.2.1 Defining Documents Types and Document Flows

The concept of configurable document templates and data objects has already been emphasised as a key feature of the way WBS/Office enables administrative activities to be modelled. Figure 11-3 illustrates how a user can configure a document template object. Using this dialog the user defines a document's characteristics by selecting from the range of available data-types (a complete listing of the document data-types included in WBS/Office is provided in Appendix 3). Since data-types are themselves objects, the user is also able to rename specific instances of these objects to more closely match the terminology used on that type of document in the firm being modelled. For example, an instance of the generic *Authorisation_1* data type might be renamed *ProdnSupervisorSignOff* if applied to a stores requisition document. This should not only simplify the model-building and simulation process, but also help to provide a more meaningful way of communicating new ideas to those people that will be affected when a new facility is implemented; in other words, people will be able to relate more directly to terminology used in the model.

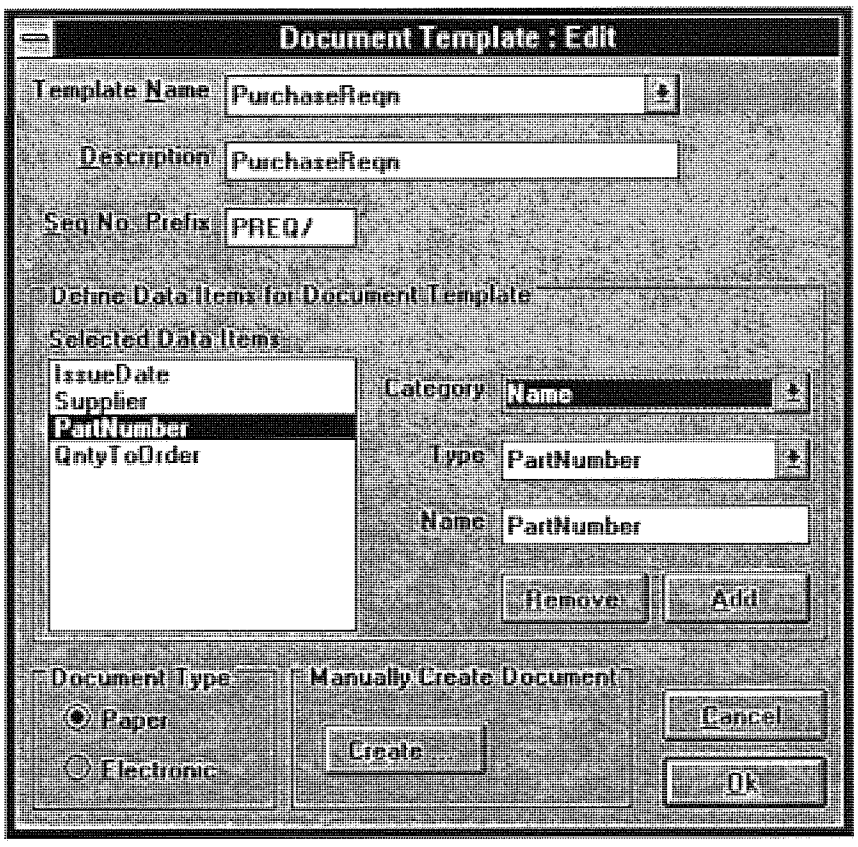


Figure 11-3 Configuring a Document Template

The object-oriented design also established the notion of *tDocumentRouting* and

tDocumentOperation classes for describing how different types of document will progress through a business. A routing represents a collection of operation objects, each of which in turn correspond to an explicit set of instructions for processing a particular type of document. In other words, each operation will comprise a list of sub-tasks that define exactly how each piece of information required for that processing stage will be obtained. The user will define data to be either transcribed, requested or looked-up. Figure 11-4, for instance, shows the first operation for a purchase order where the user has defined that prices for ordered parts will be formally requested from supplier objects.

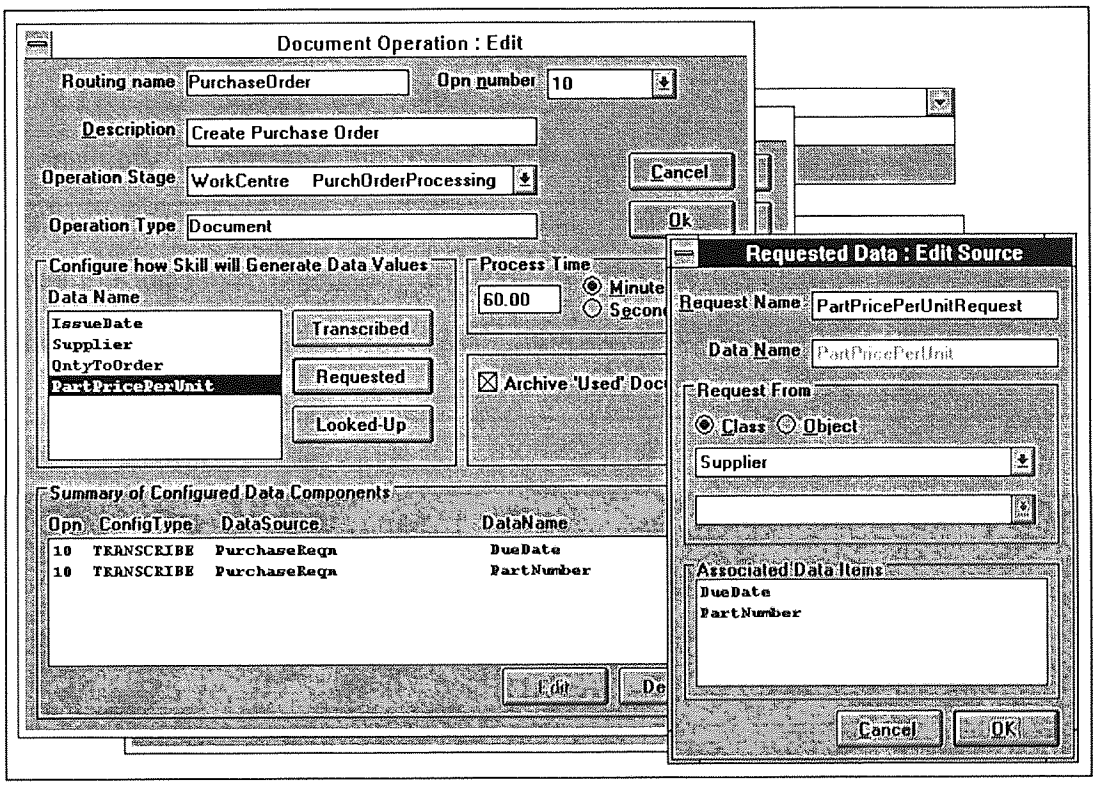


Figure 11-4 Configuring a Document Operation in WBS/Office

This means that during a simulation, a *PartPricePerUnitRequest* document will be generated by a person object, such as a buyer, and then passed to a supplier object in the same model. In this way the system can be used to simulate any delays associated with the supplier generating this information and returning it to the buyer. In turn this may affect how quickly bought-in items are delivered to the shop-floor and will therefore impact on manufacturing system performance.

11.2.2 Describing People and their Administrative Skills

In order to carry out the processing of documents and to properly represent real support departments, people need to be introduced into models of these systems. The object-oriented analysis and design phases established that, in modelling terms at least, people differ only in relation to the kinds of jobs they are able or expected to carry out. Consequently, a new *tOfficeSkill* class was designed to differentiate between models of people working in administrative areas, and to enable each person object to process certain types of document.

Each instance of this generic skill is configured by the model-builder to indicate which persons are responsible for carrying out a range of document operations. For example, Figure 11-5 illustrates a person object (a buyer in this case) who has been attributed the responsibility of carrying out *PurchaseOrder10*, the first operation defined for a *PurchaseOrder* document in this particular model. This means that during the simulation the buyer object will interpret the specific instructions defined as part of that document operation in order to add information to the purchase order.

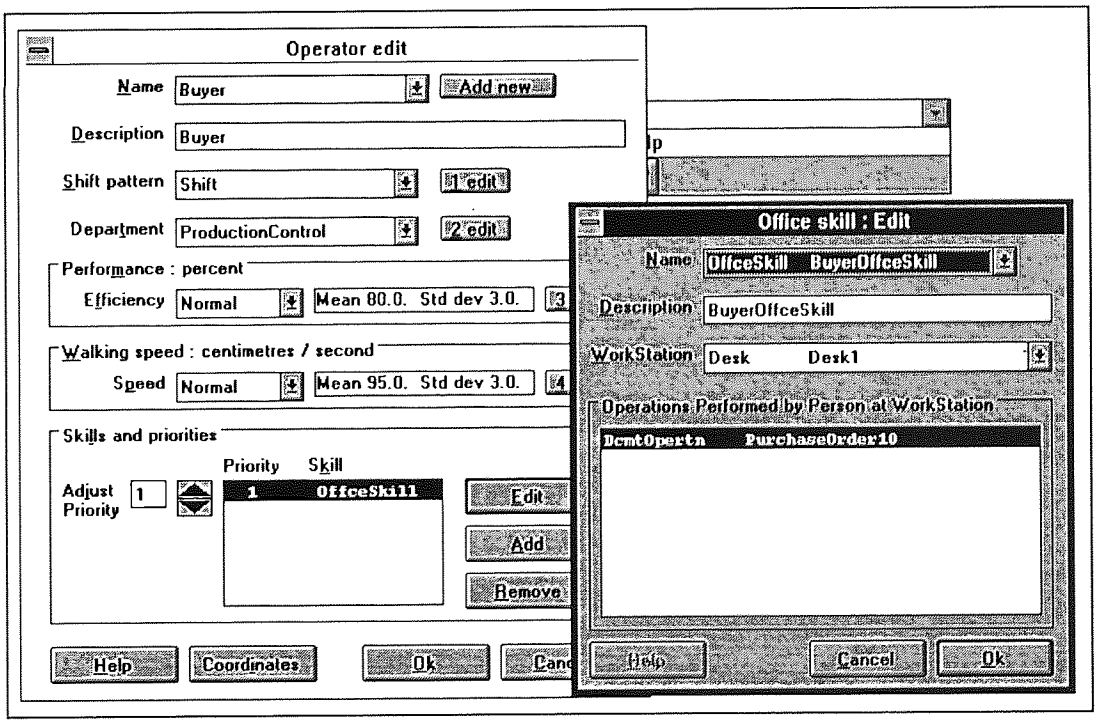


Figure 11-5 Associating a Person with an Office Skill

Notably, the object-oriented design of the *tOfficeSkill* described in preceding chapters actually sub-divided the class's functionality into three sub-classes, namely *tOfficeSkillConfiguration*, *tOfficeSkillLogic* and *tOfficeSkillProcess*. During the

implementation of WBS/Office this design feature has enabled other classes such as *tSupplier* to re-use aspects of this same document processing functionality. As Figure 11-5 demonstrates, however, from a model-builders point of view the office skill is presented as a single object, even though it is represented internally by three separate sub-classes. In other words this aspect of the design is hidden from the end user to simplify the modelling process.

In conclusion, this section has introduced some of the model-building features of WBS/Office, and emphasised the entirely data-driven approach to model construction that it offers to manufacturing system engineers. A complete listing of user-interface dialogs developed to support WBS/Office functionality is provided in Appendix 4.

11.3 Application of WBS/Office

This section presents an application of WBS/Office to demonstrate that the system fulfils the requirements that were established earlier for a modelling tool to improve the manufacturing system design process. The demonstration involves the creation of two simulation models; the first, a *factory* model, evaluates only the shop-floor aspects of the manufacturing system while the second, a *combined* model, also includes support activities, specifically the processing of works order and purchase order related documentation. For clarity both models are intentionally quite simple and do not include real company data. Full details of each model, in terms of the various types of object that each contains and the different results generated in each case, are provided in Appendix 6; a screen shot of the combined model's graphical display is shown in Figure 11-6, and emphasises the inclusion of a machine shop and support department within a single model.

In both models, works orders generate a demand for shop-floor resources. In the factory model, however, works order objects must be created by the user at model-build time and are then actioned by relevant shop-floor entities within the model as soon as the simulation begins. In other words, the factory model is typical of the kind of manufacturing system model described in earlier chapters, where the impact of any delays that might arise as each order is prepared cannot be evaluated properly in the context of manufacturing system performance.

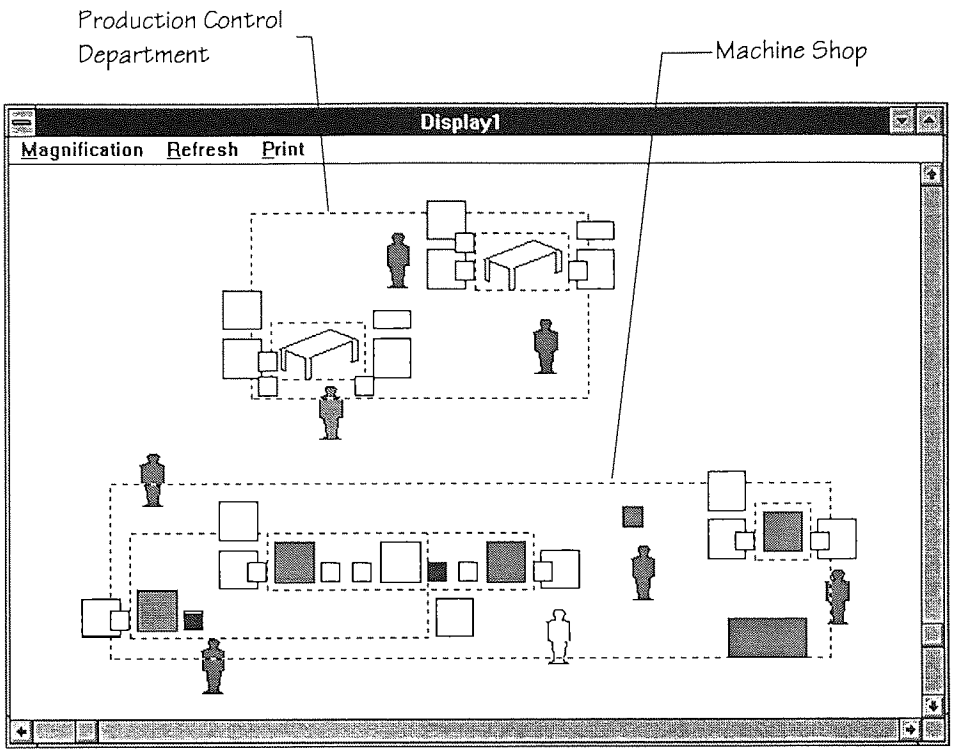


Figure 11-6 Sample Simulation Model Created using WBS/Office

In contrast, the combined model actually simulates the creation of these documents, as well as purchase order documents, by other objects that represent the personnel in production control and buying departments. Once the support functions have prepared these documents, the works orders are transferred to the manufacturing facility while purchase orders are passed on to supplier objects.

A detailed representation of a goods inwards function is not yet possible given the memory limitations imposed by *Windows 3.1* that were described earlier. Nevertheless, the scope of the combined model extends to simulating the delivery of bought-in items from suppliers to a central stores area. Deliveries are made against purchase orders generated by the buying department during the simulation, and progress of the manufacturing facility is then constrained by the availability of these bought in items. Works orders and purchase orders are generated in this model on a simple lot-for-lot basis, again because the current implementation is as yet unable to include the more sophisticated production control functionality provided by WBS/Control. Nevertheless, the combined model demonstrates the feasibility of using WBS/Office to simulate a variety of support department delays, and their impact on manufacturing system performance, as well as the ability to configure a model to reflect the information processing needs of different manufacturing organisations.

Figure 11-7, for example, shows a purchase order document, referred to in the model as *PORD/00001*, that has been generated during the simulation by a buyer, or person object. In other words, the order represents an instance of a document that would exist physically or electronically in a real business and would therefore be used to pass information between different functional areas. This particular document contains information that will eventually be received by a supplier object and then actioned appropriately by that object. In this case the supplier is being instructed to deliver a quantity of *300* of *RawMat11* to the factory. This type of representation means that both the process of generating and then responding to information affects the progress of the simulation in a way that would not be possible if documents were modelled simply as 'tokens'.

Overall, the two different types of models described here demonstrate WBS/Office's ability to simulate the dynamic and variable behaviour of entities in both shop-floor and support functions, and to also mimic the kinds of interactions that typically take place between these different aspects of a manufacturing organisation. These features were identified in preceding chapters as important requirements for evaluating manufacturing system design decisions and the system therefore fulfils the requirements of the object-oriented analysis and design stages.

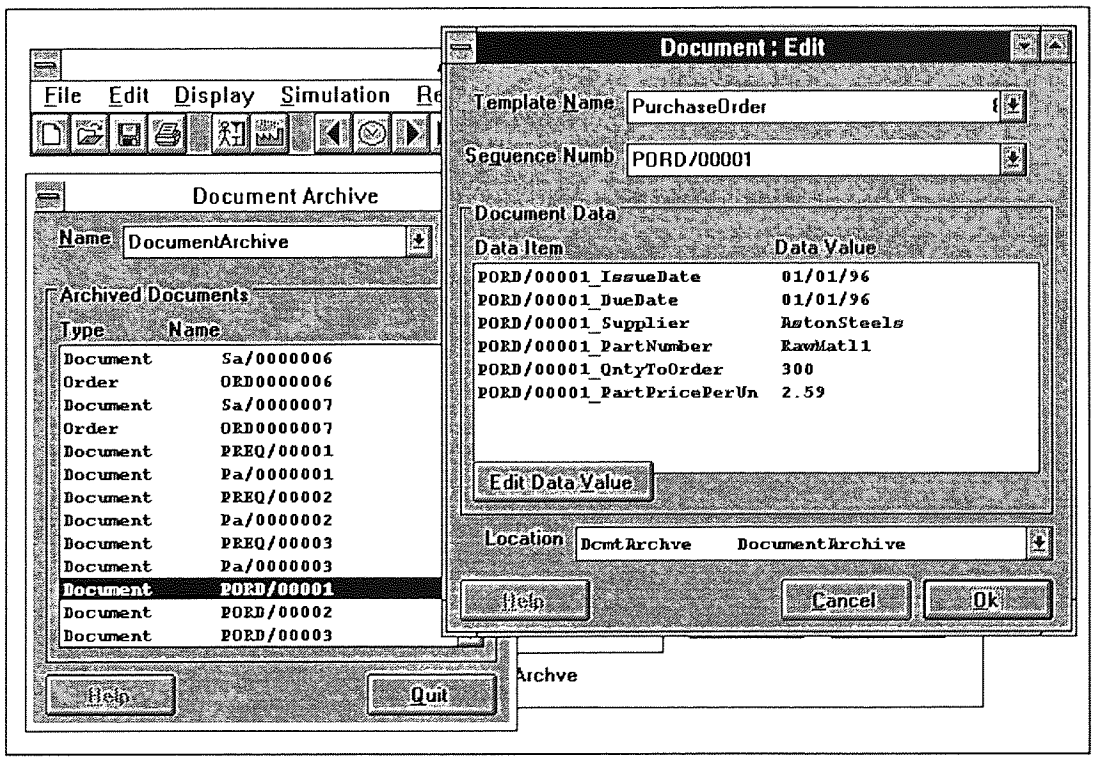


Figure 11-7 Sample Documents Objects Generated During a WBS/Office Simulation

Also, for clarity both models are intentionally quite simple and do not include real company data. Nevertheless, they serve as important examples of very contrasting evaluation methods; on the one hand, there is a conventional manufacturing model that includes only shop-floor activities, while a more comprehensive model also integrates support activities into the same representation of the manufacturing facility. Furthermore, the combined model was able to simulate delays arising in support functions that affected the performance of the manufacturing system.

The main objective of the thesis has been the provision of a better simulation tool for use in the manufacturing system design process. In this respect, these two, albeit relatively simple models succeed in demonstrating the feasibility of evaluating manufacturing system designs using WBS/Office in a way that the research has demonstrated not to be possible using other simulation tools. This application of WBS/Office therefore emphasises the significance of the work that has been conducted as part of the research project.

11.4 Summary

This chapter has described the issues involved in implementing the object-oriented design of WBS/Office that was presented earlier into a usable data-driven simulator. The amount of additional functionality that the system proposes to offer is such that it has not been possible to implement the design fully due to limitations imposed by the target operating system. A complete implementation is possible, however, by porting the existing source code to a 32-bit programming environment, although it has not been feasible to complete this conversion within the time frame imposed by the research project.

Nevertheless, the current implementation of WBS/Office enables model-builders to construct simulation models of a manufacturing system and most of its supporting functions without the need to resort to programming or to defining simulation logic. In other words, the system is completely data-driven. Furthermore, example models have been constructed using the system to simulate the interactions between a machine shop, a buying department and a production control function, and which demonstrate the feasibility of using WBS/Office to analyse manufacturing system designs in this

way. As such, WBS/Office fulfils the main objective of the thesis, and its development thereby represents a significant contribution to the issue of predicting the likely performance of a proposed manufacturing system prior to implementation.

This chapter completes the main body of the thesis. The next and final chapter presents the conclusions that can been drawn from this research project and identifies opportunities for further work.

12. Conclusions and Opportunities for Further Work

The thesis has examined the provision of a new simulator to assist in the design and analysis of production systems. This chapter presents the main conclusions that can been drawn from this research and identifies opportunities for further work.

12.1 Summary of the Manufacturing System Design Problem

The thesis has established limitations in the current manufacturing system design process. In order to continually improve the competitiveness of their organisations manufacturing engineers often design or re-design a firm's production facilities, with the expectation of making further gains in manufacturing system performance. The research has addressed specific aspects of this decision-making process such as the re-configuration of shop-floor plant and labour, the selection or design of a production control system, make-or-buy decisions and so on.

Since the manufacturing systems that result from these decisions must also interact with many other parts of an organisation, the expected performance improvements can often be hampered by constraints that arise elsewhere in the business. As a result, engineers should attempt to predict just how well a proposed design will perform when these other factors are taken into consideration. Nevertheless, the research has also established that where quantitative analysis is used to evaluate alternative manufacturing system design configurations, the analysis model invariably ignores the potential impact of support functions on a system's overall performance. A more comprehensive modelling approach is therefore required.

A study of how various business functions interact has demonstrated that to properly represent the kind of delays that give rise to support department constraints, a model should actually portray the dynamic and stochastic behaviour of entities in both the manufacturing and non-manufacturing aspects of a business. This implies that computer simulation be used to model manufacturing system design decisions but current simulation software does not provide a sufficient range of functionality to enable the behaviour of these entities to be assessed properly. The main objective of the research has therefore been the development of a new simulator that will overcome

limitations of existing software by including manufacturing and non-manufacturing activities in a single model and thereby providing a better way of evaluating manufacturing system design decisions.

12.2 The Development of a Simulation Solution

As well as being able to represent specific support department features, the new simulator has also been required to fulfil other practical considerations which have had important implications for the way the system has been designed.

12.2.1 A Data-Driven Modelling Solution

A review of different simulation methods has established discrete-event simulation as the most appropriate technique for evaluating manufacturing system designs. This approach offers detailed modelling of dynamic behaviour and uncertainty, and also provides a closer match between entities in the model and objects in the real world, which is beneficial from both a development and a model-building point of view.

However, if a modelling tool is to be used in practice to evaluate the consequences of this type of decision, then it should also allow people with only a limited knowledge of modelling techniques to quickly build models. Since a discrete-event model is likely to include a high level of detail, the approach is potentially demanding in terms of the time and level of simulation expertise required to construct models. Consequently, another important requirement of the new system is that discrete-event logic describing the behaviour of entities in a manufacturing business be pre-defined in the simulator, allowing a model-builder to simply configure these elements together to create a model of their particular organisation. In other words, an important system requirement is that it be a completely data-driven simulator in order to be practical for building models of the size and scope needed for a manufacturing system design exercise.

12.2.2 Development of an Object-Oriented Simulator

The research has reviewed the implications for developing the new simulator (named WBS/Office) as an object-oriented application. This type of approach potentially offers a more logical and safer way to construct any large software application but is particularly appropriate for developing simulation software since people naturally tend

to view the real-world in terms of objects. This means that the simulator's data constructs, or classes, can be more easily designed to represent real-world objects. In turn, this also means that the end-users are more likely to be presented with concepts and terminology that relates to the system being modelled, which will invariably simplify the model-building process. In other words, object-oriented principles offer a potentially better way of fulfilling both the functional and ease-of-use issues relating to the new simulator.

In addition, by designing WBS/Office as an object-oriented application, it has also been possible to re-use much of the functionality provided by two existing simulation class libraries, namely the Advanced Factory Simulator (AFS) and WBS/Control. These libraries already provide the ability to simulate the material processing and production control activities within a business. As a result, the development of WBS/Office has mainly involved the design of new simulation classes for representing administrative and other types of support department activity, such as stores-handling. These classes have been integrated with the existing shop-floor and control system classes to provide an unusually comprehensive discrete-event simulation tool for modelling manufacturing organisations.

A particularly novel feature of the design is the ability for users to model a firm's specific information processing requirements in order to investigate how delays incurred in these areas might hamper shop-floor performance. In fact, example models have been constructed using WBS/Office to demonstrate the significance and feasibility of including support department activities in a manufacturing system evaluation model. The system therefore fulfils the main objective of providing a new simulator to assist in the design and analysis of production systems.

12.2.3 Observations Relating to the Object-Oriented Development Process

The scope and size of WBS/Office is sufficiently large to have exposed the author, from both a theoretical and practical point of view, to a range of object-oriented software development issues during the system's design and implementation. In particular, some general observations made during the software development process, specifically in relation to object-oriented simulation, are worthy of brief discussion here.

Firstly, in general terms simulation involves the abstraction of the relevant features of a real system into a model that represents or approximates to that system for the purposes of experimentation. To this end, the purpose of WBS/Office is to allow manufacturing system engineers to easily abstract the various types of entity found in a real manufacturing business into a simulation model. As mentioned earlier, people naturally tend to view these entities in terms of objects and certainly throughout the analysis and design of WBS/Office, object-oriented modelling proved to be a particularly suitable way of logically breaking up a complex simulation problem into more manageable parts. As a result, the approach has enabled specific characteristics of support functions to be represented in the program code in the same way as they are normally viewed in the real-world, which is reflected in the types of entity that a model-builder can use to model support activities. In other words, the expected benefits of adopting an object-oriented approach were realised during the design and implementation of WBS/Office.

In practice, development of the system was also very much an interactive process with the original object model having to be continually refined during later stages of software construction. It was also observed that adopting an object-oriented development policy encourages system designers to encapsulate or 'hide' information in self-contained units or classes in a way that is more difficult to enforce if object-oriented methods are not applied. This is particularly important from the point of view of group development since individual developers should be encouraged to implement their own aspects of the system as encapsulated units or classes that can be accessed safely by other developers without causing knock-on effects elsewhere in the system. In the case of WBS/Office, for example, development of the system has involved re-use and expansion of many aspects of the existing AFS architecture. Since this architecture was constructed originally in a strictly object-oriented way, new support department classes could be added to the architecture with minimal knock-on effects in other parts of the simulator.

In addition, the ability to inherit functionality from existing classes has also ensured greater consistency between the shop-floor and support department aspects of the simulator. Nevertheless, against the many positive opportunities presented by the application of object-oriented concepts, the approach can also introduce some potential

problems. Specifically, concepts such as polymorphism (Rumbaugh et al., 1991; Booch, 1994) mean that existing code can sometimes be difficult to understand since the sequence in which particular blocks of code will be executed may not be determined by the simulator until run time. This problem is especially relevant in relation to functionality such as the state transition logic typically required by a data-driven simulator. While the overall system can be neatly designed and implemented in terms of objects, the fundamental logic that controls when and how entities interact or change state can get very complicated.

Indeed, by the very nature of the real-world problems being addressed, the internal logic of any data-driven simulator is likely to be very complicated, even when developed using object-oriented techniques. In practice, this means that powerful development and debugging tools need to be used to properly test and debug this type of application. Nevertheless, object-orientation has proved overall to be an invaluable and fundamental part of the development of WBS/Office which should be reflected in the quality of the completed system that will be made available to manufacturing system engineers.

12.3 Practical Implications and Limitationsof WBS/Office

While the current design and implementation of WBS/Office does provide manufacturing system designers with a range of important and novel model-building capabilities, the system does have some potential limitations that users ought to be aware of before attempting to use the system for analysing alternative design proposals.

For example, decision-makers ought to be aware of the types of organisation that the system can and cannot model. WBS/Office re-uses an existing library of simulation classes, namely those provided by the Advanced Factory Simulator (AFS), to represent shop-floor activities. AFS enables model-builders to represent real-world entities such as manually operated machines and material-handling devices, workstations and workcentres, as well as specialised system configurations such as Nagare flow. In other words, this aspect of the simulator is appropriate for modelling entities typically found in a batch manufacturing environment but does not include classes such as

conveyors or automated-guided-vehicles (AGVs) that are more characteristic of flow-line or continuous production systems.

Furthermore, the additional information or document-related simulation classes that WBS/Office provides are appropriate for modelling organisations where purchasing and sales-order processing activity is likely to be quite intensive. This also implies batch-manufacturing environments, and in particular, make-to-order scenarios characterised by medium volumes of a wide variety of products. As a result, WBS/Office ought to be considered primarily as a simulator for modelling firms in the make-to-order batch-manufacturing sector as opposed to other types of manufacturing organisation.

The current WBS/Office architecture also assumes that other entities will always be available to generate the 'data' to be processed by support department entities in the model. For example, the current design assumes that production control information will be generated by WBS/Control classes and that financial or cost information will be introduced into a model by, say, a commercial accounting system.

In practice, it may not be appropriate or feasible to create a detailed model of these aspects of an organisation but at present WBS/Office cannot model production control or other information systems in an aggregate or less detailed way. In this context, for example, aggregate modelling might involve a model-builder defining simple algorithms for generating information values during a simulation rather than using real systems or detailed models of those systems to generate that data.

If this capability were available it would broaden the application of the simulator as well as reducing the model-building effort. Indeed, the effort needed to build simulation models of the scope proposed in this thesis is another factor that decision-makers ought to appreciate. In practice, a lot of data is likely to be required but the current system provides no facility for importing or uploading any of this data from existing information systems or databases. Instead, the system requires that all data needed to define the manufacturing and support functions within an organisation be manually collated, inputted and then checked by the user. As a result, even though WBS/Office allows decision-makers to construct models without the need to resort to

programming, this process may involve a significant amount of time and the reliance on manual data-entry, in particular, will potentially increase the likelihood of errors being introduced into models.

In addition, although each WBS/Office class encapsulates logic that the object-oriented analysis phase identified as appropriate for modelling support department activities, the system cannot prevent invalid models from being created. For example, the system does not control or guide the model-building process to ensure that all aspects of the model are constructed properly and in the correct sequence. In practice, therefore, those people that are most knowledgeable about the real or proposed system being modelled ought to be encouraged to use the graphical display that WBS/Office provides in order to help identify errorsin the model.

A related but more general criticism of the current system is that it provides only limited support for interpreting the results of a simulation experiment. In other words, while the simulator might identify that support department activities are constraining shop-floor performance, the system is limited in its ability to help decision-makers identify which particular aspects or features of the support activity are contributing to this behaviour. The ability to perform this type of analysis would help to focus a decision-maker's attention on which aspects of their current proposal ought to be modified.

Finally, while each aspect of WBS/Office has been designed to be both adaptable and extendible for modelling a range of different manufacturing organisations, one particular aspect of the design is still arguably quite weak in this respect.

The concept of *DocumentAction* classes have been designed as a convenient link between the manufacturing and non-manufacturing aspects of the system. This feature attempts to encapsulate a form of 'intelligence' or logic that enables shop-floor entities in the model, such as *Person* objects, to take appropriate actions as a result of receiving completed documents from non-manufacturing entities, even though the structure of these documents is potentially quite unique for each model. In that sense, this feature attempts to reduce the model building effort but, in practice, it does so at the expense of limiting the scope of WBS/Office and also increases the likelihood of model

invalidation since the model-builder does not have access to the logic associated with this aspect of the model.

In hindsight, therefore, a more robust design of this aspect of the system would be to allow model-builders to associate shop-floor related 'skills' with specific document types (or *DocumentTemplate* objects) during the model-building process. By adopting this approach, the system would not have to try and automatically decipher what action ought to be performed as a result of a manufacturing entity receiving a *Document* object. Instead, model-builders would have more control of how entities in their model interact but would still be relieved of the need to explicitly define any model logic.

This section has described some of the practical implications and limitations of the current version of WBS/Office. Nevertheless, rather than identifying faults with the existing system architecture, the criticisms relate mainly to functionality that has not yet been added to this design and therefore represent opportunities for further work. The next section will discuss additional opportunities in more detail.

12.4 Opportunities for Further Work

Examples of WBS/Office having been applied to simulate the interactions between a production system and its support functions were presented earlier to demonstrate the feasibility of using such a system to improve the manufacturing system design process. By providing decision-makers with this more appropriate simulation tool for predicting the wider effects of design decisions, the main objective of the research has been fulfilled. Nevertheless, there are opportunities to develop the work described in the thesis even further; these opportunities will now be discussed in more detail.

12.4.1 Extending the Functionality of WBS/Office

The range of modelling features offered by WBS/Office is presently constrained by its target operating system, namely *Microsoft Windows 3.1*. As a 16-bit operating system, *Windows* limits the size and amount of data that can be handled by any application. In practice this has meant that the full range of simulation classes needed to represent support functions, and which have been presented in the form of an object-oriented design for WBS/Office, cannot be fully implemented as a 16-bit application.

Specifically, the current implementation of WBS/Office does not include the WBS/Control class library for modelling production control systems, or the range of classes designed to represent the non-administrative aspects of support department work, such as goods inwards handling.

Since the original implementation of WBS/Office, 32-bit operating systems have become available in the form of *Microsoft Windows 95* and *Microsoft Windows NT*. Both of these operating systems overcome the restrictions imposed by the older *Microsoft Windows 3.1* system, and so the conversion of WBS/Office to a 32-bit application presents an important opportunity for further research and development of the simulator that would enable a full implementation of the system's proposed functionality.

Furthermore, while Borland's *Object Pascal* has provided a very suitable object-oriented language for coding the core functionality of WBS/Office, it is quite limited in terms of providing developers with an easy way of creating a Windows-style graphical user-interface. For this reason, the system's user-interface has not been implemented as an object-oriented application. However, updated versions of Borland's *Object Pascal* have since become available in the form of Borland *Delphi* that would enable the existing core simulator to be re-compiled as a 32-bit application and would also provide developers with an easy way of creating both an object-oriented and Windows-style graphical user-interface. An object-oriented interface would make this aspect of the simulator easier to extend, which is relevant in the context of other potential applications of WBS/Office which will now be discussed.

12.4.2 Development of a New Manufacturing System Design Methodology

The main inspiration for developing WBS/Office has clearly been to overcome limitations of the current approach to manufacturing system design. In fact, the research has included a review of formal methodologies relating to manufacturing system design, production control system design and make-versus-buy analysis. The lack of adequate quantitative analysis tools has been identified as a shortcoming of all these formal methods.

WBS/Office therefore successfully addresses the main issue of providing decision-makers with a more appropriate simulation tool for performing quantitative analysis. Nevertheless, the potential of this new simulator is only likely to be fully realised if a new or revised methodology is produced to reflect the need for decision-makers to use the tool for predicting how manufacturing system design and related decisions might be hampered by constraints that arise elsewhere in the business. In other words, a new methodology is needed to emphasise the advantages of carrying out the kind of evaluation offered by WBS/Office as part of the design process. Such a methodology would also need to address specific limitations of current methodologies that were identified during the review presented earlier.

The development of a new methodology could also involve industrial applications of WBS/Office. In other words, WBS/Office and an associated methodology could be used to support the design and implementation of real manufacturing systems. This type of industrial application would serve at least two purposes.

Firstly, industrial applications of the simulator could actually form part of the development process for the new methodology. For example, real case studies might be used to identify the relative significance of different support department activities in terms of their impact on manufacturing system performance. The results of this type of analysis might then be presented as a set of alternative solutions or guiding principles for different aspects of the production system design problem. This might be similar to the concept of 'Design Option Guides' used in the DRAMA II methodology (Bennett & Forrester, 1993) that was reviewed earlier, but would also need to emphasise the importance of properly assessing any 'guided' solution in the context of overall business performance.

Secondly, since the final development of any software application is invariably an iterative testing and debugging process, using WBS/Office in an industrial context also presents an opportunity to further test the general robustness of the system's design and implementation. It also provides a way of gaining feedback from potential end-users of the software that can then be used to modify or extend the system's functionality to reflect specific modelling needs.

12.4.3 Development of the Whole Business Simulator

It was explained in the introductory chapter that the research described in this thesis has been completed within the context of a broader research agenda proposed by Love et al. (1992). The overall objective of this research agenda is to develop a modelling tool that satisfies the requirements for a Whole Business Simulator (WBS). In their paper, Love et al. (1992) demonstrate that product design and manufacturing system design decisions are likely to have a significant impact on the financial performance of any business. For this reason, WBS is proposed as a novel modelling tool that will predict how a business is likely to perform financially if a particular engineering decision or design were to be implemented.

The current implementation of WBS/Office represents an important aspect of the WBS concept since it allows detailed modelling of a firm's support functions, which will invariably impact on business performance. In addition, however, specific features of WBS/Office's overall design present opportunities for incorporating other important WBS concepts into models of support functions. For example, in order to predict how a business is likely to perform financially, WBS will need to generate and process financial information. The 'document data-object' concept in WBS/Office has been implemented in such a way that additional data-types, such as those representing financial information, can be added to the simulator relatively easily.

Also, the notion of 'document operations' in WBS/Office allows model-builders to configure how different types of information are to be generated during a simulation. This aspect of the simulator could therefore be extended to provide 'hooks' into real company software applications, such as financial accounting systems or computerised inventory systems to provide a closer match between the information used in the real business and the information used in the model of that business. WBS/Office could also then be used to simulate the processing of financial information and presentation of financial results which is ultimately the main objective of the Whole Business Simulator.

The generic design of the document-related classes in WBS/Office also presents other opportunities in relation to WBS. Since the system allows document data-objects to

represent not just numeric data but also references to other simulation objects, such as part numbers or suppliers, the system could actually be used to model the manufacturing system design or new product introduction processes themselves. For example, an 'engineering design' might be modelled as a document that contains a 'drawing' object, a 'bill of materials' object, and a set of approval signatures. If these additional data-types were added to WBS/Office, the system could then be used to predict the kind of delays involved in completing these aspects of the engineering design process, and their impact on manufacturing system performance and ultimately on the firm's financial performance.

12.4.4 Other Applications of WBS/Office

The research has so far related to the provision of a modelling tool for predicting how support departments might constrain manufacturing system performance. In other words, the expected outcome of the modelling process would be modifications to the manufacturing system that take account of support department constraints. Nevertheless, the same modelling functionality could also be used to improve the effectiveness and productivity of support departments themselves.

For example, a review of business process re-engineering (BPR) modelling tools was presented earlier. The main limitation of these systems from a manufacturing system design point of view was that they tend to represent documents simply as tokens. In other words, the context of the information that would be contained on these documents in a real business cannot be used to affect the progress of the simulation. In contrast, a novel feature of WBS/Office is the ability to include these information values as part of the simulation. As a result, WBS/Office could also be used as a BPR modelling tool, but to provide a more detailed model of areas likely to be affected by new business processes than modelling tools developed specifically for this purpose. Decision-makers would then be able to predict the potential gains in business performance that would be realised if alternative process designs were implemented. Alternatively, firm's proposing to implement ISO9000 or BS5750-type procedures could also use the functionality provided by WBS/Office. In this case, the simulator might be used to predict how effectively existing support department personnel would be able to cope with any additional administrative workloads that new quality procedures might introduce.

In summary, an important opportunity exists to incorporate WBS/Office into a new manufacturing system design methodology that reflects the need to predict how design decisions might be hampered by constraints that arise elsewhere in the business. Furthermore, while WBS/Office has been designed primarily to improve the manufacturing system design process, specific features of the design, including its development as an object-oriented application, mean that the system can potentially be extended for use in other areas of business decision-making. Consequently, opportunities exist to further the research described in the thesis to model other types of business problem.

12.5 Concluding Remarks

The effective design and re-design of production systems represents a very important part of a manufacturing organisation's activities. Computer simulation is intended to allow manufacturing system engineer's to test the effectiveness of their designs without disturbing the real system. However, current modelling tools restrict analysis to those aspects of a business most directly related to shop-floor activities, and by doing so, ignore the potentially significant impact of other factors that can ultimately hamper the effectiveness of any proposed design. The work described in this thesis has therefore related to the development of a new computer simulation tool that enables a more holistic evaluation of manufacturing system design decisions. An important aspect of the work has been the application of object-oriented principles to provide a simulator that relates closely to the problems it is intended to address.

References

Adiga, S., 1989,"Software Modelling of Manufacturing Systems : A Case for an Object-Oriented Programming Approach", *Annals of Operations Research*, 17 : 363-378

Anderson, E.J., 1994,*The Management of Manufacturing* Addison-Wesley.

Askin, R.G., Standridge, C.R., 1993,*Modeling and Analysis of Manufacturing Systems*, John Wiley & Sons, New York.

Baily, P., 1991,*Purchasing Systems and Records*, 3rd edition, Gower Press (published in association with the Institute of Purchasing and Supply)

Ball, P.D., 1994,*Design of an Expandable Manufacturing Simulator Through the Application of Object-Oriented Techniques*, PhD Thesis, The University of Aston in Birmingham, UK.

Ball, P.D., Boughton, N.J., Love, D.M., 1994,"Extending the Boundaries of Manufacturing Simulation", *International Journal of Manufacturing System Design*, 1(2) : 99-109

Banerjee, S.K., Kourouklis, A.P., Penman, J., 1994,"Manufacturing Planning and Control Decision Model : A Design Methodology", *International Journal of Production Economics*, 34(3) : 283-292

Banks, J., 1995,"Software for Simulation", *Proceedings of the Winter Simulation Conference* : 3-6 December : 32-38

Banks, J., Aviles, E., Mclaughlin, J.R., Yuan R.C., 1991,"The Simulator: New Member of the Simulation Family", *Interfaces*, 21(2) : 76-86

Barton, J.A., Bailey, K., Love, D.M., 1992,"A Working Demonstrator of a Whole Business Simulator", *Proceedings of the 8th National Conference for Manufacturing Research*, University of Central England, Birmingham, UK, 8-10 September : 49-53

Bennett, D.J. & Forrester, P.L., 1993,*Market-focused Production Systems : Design and Implementation*, Prentice-Hall, UK.

Bergen, S.A., 1977,"The Make or Buy Decision", *R & D Management*, 8(1) : 39-42

Bernard, P., 1989,"The Carrying Cost Paradox : How Do You Manage It?", *Industrial Engineering*, 21(11) : 40-46

Berry, B. & McLintock A., 1991,"Accountants and Financial Modelling", *OR Insight*, 4(4) : 11-14

Billesbach, T.J. & Schniederjans M.J., 1989,"Applicability of Just-In-Time Techniques in Administration", *Production and Inventory Management Journal*, 30(3) : 40-45

Blundell, P., 1994, "Difficult Decisions Simplified at NedCar", *Management Services*, 38(5) : 16-17

Booch, G., 1994, *Object-Oriented Analysis and Design with Applications*, 2nd edition, Benjamin/Cummings Publishing Company, USA.

Booth, R., 1995, "Process Mapping", *Management Accounting*, 73(3) : 32

Boughton, N.J. & Jackson, A.T., 1994, "A Methodology For Evaluating the Cost Implications of Manufacturing System Design/Redesigns using Activity-based Cost Models", internal publication, Lucas Engineering & Systems, UK.

Boughton, N.J., 1995, *Modelling Manufacturing Planning and Control Systems : the Application of Object-Oriented Principles and Discrete-Event Simulation*, PhD Thesis, Aston University, Birmingham, UK.

Bravoco, R.R. & Yadav S.B., 1985, "A Methodology to Model the Dynamic Structure of an Organisation", *Information Systems*, 10(3) : 299-317

Brewer, S.K., 1995, "Simulation : Why aren't we where we should be?", *Industrial Engineering*, 27(1) : 47-48

Bridge, K., 1990, *The Application of Computerised Modelling Techniques in Manufacturing System Design*, PhD Thesis, The University of Aston in Birmingham, UK.

Bridgeland, D. & Becker, S., 1994, "Simulation Satyagraha, a Successful Strategy for Business Process Reengineering", *Proceedings of the Winter Simulation Conference*, 1214-1220

Brimson, J.A., 1993, "Activity Product Cost", *IEEE Engineering Management Review*, 21(1) : 16-29

Buchowicz, B.S., 1991, "A Process Model of Make-vs.-Buy Decision-Making; The Case of Manufacturing Software", *IEEE Transactions on Engineering Management*, 38(1) : 24-32

Burbidge, J.L., 1979, *The Principles of Production Control*, MacDonald and Evans, Plymouth

Burhanuddin, S. & Randhawa, S.U., 1992, "A Framework for Integrating Manufacturing Process Design and Analysis", *Computers and Industrial Engineering*, 23(1-4) : 27-30

Busby, J.S. & Williams, G.M., 1993, "The Value and Limitations of Using Process Models to Describe the Manufacturing Organisation", *International Journal of Production Research*, 31(9) : 2179-2194

Byrd, J., 1978, "The Value of Queueing Theory", *Interfaces*, 8(3) : 22-26

CACI, 1992, *Reference Manual for SIMFACTORY II.5 and* SIMPROCESS, CACI Products Company, La Jolla, California, U.S.A.

Carrie, A., 1988, *Simulation of Manufacturing Systems*, John Wiley & Sons, UK.

Chaharbaghi, K. 1990, "Using Simulation to Solve Design and Operational Decisions", *International Journal of Operations & Production Management*, 10(9) : 89-105

Chase, R.B. & Garvin, D.A., 1989, "The Service Factory", *Harvard Business Review*, 67(4) : 61-69

Christy, D.P. & Kleindorfer, G.B., 1990, "Simultaneous Cost and Production Analysis of Manufacturing Systems", *Proceedings of the Winter Simulation Conference*: 582-589

Clarke, S. & Tobias A., 1995, "Corporate Modelling in the UK : a survey", *OR Insight*, 8(3) : 15-20

Collins, P., 1992, "Admin JIT - Method Study Comes of Age", *Management Services*, 36(12) : 16-19

Colquhoun, G.J., Baines, R.W., Crossley, R., 1993, "A State of the Art Review of IDEF0", *International Journal of Computer Integrated Manufacturing* 6(4) : 252-264

Cooper, R. & Kaplan, R.S., 1988a, "How Cost Accounting Distorts Product Costs", *Management Accounting (London)*, 69(10) : 20-27

Cooper, R. & Kaplan, R.S., 1988b, "Measure Costs Right : Make the Right Decisions", *Harvard Business Review*, 66(5) : 96-103

Coyle, R.G., 1977, *Management System Dynamics*, John Wiley & Sons, UK.

Dahl, O.J. & Nygaard, K. 1966, "SIMULA - an ALGOL-based Simulation Language", *Communications of the ACM*, 9(9) : 671-678

Dale, B.G. & Cunningham, M.T. 1983, "The Importance of Factors Other Than Cost Considerations in Make or Buy Decisions", *International Journal of Operations & Production Management*, 4(3) : 43-54

Dale, B.G. & Russell, D. 1983, "Production Control for Small Group Production", *OMEGA The International Journal of Management Science*, 11(2) : 175-185

Dales, M.W. & Johnson, P. (*sic.*), 1986, "The Redesign of a Manufacturing Business", *Proceedings of the IMechE Conference on U.K. Research in Advanced Manufacture*, December : C379-386

Darlington, J. 1995, "Optimising Production Resources", *Management Accounting (London)*, 73(4) : 57-60

Davenport, T.H. & Short, J.E., 1990, "The New Industrial Engineering : Information Technology and Business Process Redesign", *Sloan Management Review*, 31(4) : 11-27

Davies, M.N., 1994,"Back-office Process Management in the Financial Services : a Simulation Approach using a Model Generator", *Journal of the Operational Research Society*, 45(12) : 1363-1373

Devereux, S., Smith, P., Wood, D., 1994,"A Survey of the Use of Design Methodologies for Implementing Change in Manufacturing Companies in the United Kingdom", *International Journal of Manufacturing System Design*, 1(1) : 51-58

Dixon, R.L., 1953,"Creep", *Journal of Accounting*, July : 48-55

Doll, W.J. & Vonderembse, M.A., 1991,"The Evolution of Manufacturing Systems : Toward the Post-Industrial Enterprise", *OMEGA : International Journal of Management Science*, 19(5) : 401-411

Drury, C., 1992,*Management and Cost Accounting* Chapman & Hall, London

Dwyer, J. & Korwin, S., 1990,"Honeywell Puts Simulation to Work in Multiple Areas", *Industrial Engineering*, 22(10) : 33-36

Elder, M.D., 1995a,"Why Simul8 ?", University of Strathclyde, Department of Management Science, Working Paper Series (Theory and Practice), Number 95/15

Elder, M.D., 1995b,"Using Simulation Earlier", *International Journal of Manufacturing Systems Design*, 2(2), 139-144

Floyd, B., Walls, J., Marr, K., 1995,"Managing Spreadsheet Model Development", *Journal of Systems Management*, 46(3) : 38-43, 68

Fritz, S., Schmid, F., Wu, B., 1986,"A Survey of Current Practise and Development in Computer Aided Manufacturing System Design", *Proceedings of the International Conference on Managing Integrated Manufacturing*, Keele, UK, September : 673-689

Forrester, J.W., 1958,"Industrial Dynamics : A Major Breakthrough for Decision Makers", *Harvard Business Review*, 36(4) : 37-66

Gladwin, B. & Tumay, K., 1994,"Modelling Business Processes with Simulation Tools", *Proceedings of the Winter Simulation Conference*: 114-121

Gogg, T.J. & Mott, J.R.A., 1992,*Improve Quality and Productivity with Simulation*, JMI Consulting Group, USA

Goldratt, E. & Fox, R., 1989,*The Goal*, Gower, UK.

Graham, I., 1994,*Object-Oriented Methods*, Addison-Wesley, UK.

Hall, G., Rosenthal, J., Wade, J., 1993,"How to Make Reengineering Really Work", *Harvard Business Review*, 71(6): 119-131

Hall, M. & Stewart, B., 1994,"Internal Benchmarking & Business Process Re-engineering for Improved Productivity : A Case Study", *Management Services*, 38(11): 18-23

Hall, R.W., 1991, *Queueing Methods for Service and Manufacturing* Prentice-Hall International Editions.

Hamill, B.J. & Steele, P.M., 1973, *Work Measurement in the Office : an MTM Systems Workbook*, Gower Press Ltd., Essex

Hammer, M., 1990, "Reengineering Work: Don't Automate, Obliterate", *Harvard Business Review*, 68(4) : 104-112

Hammer, M. & Champy, J., 1993, *Reengineering the Corporation : A Manifesto for Business Transformation*, Nicholas Brearley Publishing, London

Harrell, C.R. & Tumay, K., 1991, "ProModel Tutorial", *Proceedings of the Winter Simulation Conference*: 101-105

Hayes, R.H. & Wheelwright, S.C., 1984, *Restoring our Competitive Edge : Competing Through Manufacturing*, John Wiley & Sons.

Heuttner, C.M. & Steudel, H.J., 1992, "Analysis of a Manufacturing System via Spreadsheet Analysis, Rapid Modelling, and Manufacturing Simulation", *International Journal of Production Research*, 30(7) : 1699-1714

Hill, T., 1993, *Manufacturing Strategy : The Strategic Management of the Manufacturing Function*, 2nd edition, The Macmillan Press Limited.

Hlupic, V. & Paul, R.J., 1995, "A Critical Evaluation of Four Manufacturing Simulators", *International Journal of Production Research*, 33(10) : 2757-2766

Hoefer, P., Dagher, EE., Davis, S., Donnelly, J., Madu, C., 1994, "Using Simulation to Understand the PID Approval Process at Indian Point 3, *Industrial Engineering*, 26(12) : 67-69

Howlett, D., 1996, "Financials with built-in Workflow", *PC User*, Issue 285 : 44-46

Huckvale, T. & Ould, M., 1994, "Process Modelling: Why, What and How" in *Software Assistance for Business Re-Engineering* edited by Spurr, K., Layzell, P., Jennison, L., Richards, N.; John Wiley & Sons, UK.

Hunter, M., 1993, "Finding Ways to Eliminate Transactions on the Factory Floor", *Industrial Engineering*, 25(7) : 30-34

Innes, J. & Mitchell, F., 1991, "ABC : A Survey of CIMA Members", *Management Accounting (London)*, 69(9) : 28-31

Innes, J. & Mitchell, F., 1995, "ABC : A Follow-up Survey of CIMA Members", *Management Accounting (London)*, 73(7) : 50-51

Jackman, J. & Johnson, E. 1993, "The Role of Queueing Network Models in Performance Evaluation of Manufacturing Systems", *Journal of the Operational Research Society*, 44(8) : 797-807

Jones, J., 1995,"SIMPROCESS III : Object-Oriented Business Process Simulation", *Proceedings of the Winter Simulation Conference*: 3-6 December : 548-551

Karger, D.W. & Hancock, W.M., 1982,*Advanced Work Measurement*, Industrial Press Inc., New York

Kellock, B., 1992,"The Evolution of the Cell Revolution", *Machinery and Production Engineering*, 21 February, 150 (3821) : 37-40

Ketcham, M.G., 1991,"Simulating Operations in Financial Service Systems", *OMEGA The International Journal of Management Science*, 19(1) : 17-30

Kochhar, A.K. & McGarrie, B., 1991,"Selection and EffectiveImplementation of Manufacturing Control Systems - The Need for a Structured Approach", *Proceedings of the 26th Annual European Conference of BPICS*: 241-256

Kochhar, A.K., Suri, A.K., Oldham, K., Thacker, S.M., McGarrie, B., 1992,"A Structured Methodology for the Effective Implementation of Manufacturing Control Systems", *Proceedings of the ACME Conference*, 109-114

Larsen, N.E. & Alting, L. 1993,"Criteria for Selecting a Production Control Philosophy", *Production Planning & Control*, 4(1) : 54-68

Law, A.M. & Haider, S.W., 1989,"Selecting Simulation Software for Manufacturing Applications : Practical Guidelines & Software Survey", *Industrial Engineering*, 21(5), 33-46

Law, A.M. & Kelton, W.D., 1991,*Simulation Modeling and Analysis*, McGraw-Hill, New York

Law, A.M., McComas, M.G., Vincent, S.G., 1994,"The Crucial Role of Input Modeling in Successful Simulation Studies", *Industrial Engineering*, 26(7) : 55-94

Lewis, P.A., 1994,*A Systemic Approach to the Design of Cellular Manufacturing Systems*, PhD Thesis, The University of Aston in Birmingham, UK.

Levine, L.D. & Aurand, S.S., 1994,"Evaluating Automated Work-flow Systems for Administrative Processes", *Interfaces*, 24(5) : 141-151

Lingineni, M., Caraway, B., Benjamin, P.C., Mayer, R., 1995,"A Tutorial on ProSim™ : a Knowledge-based Simulation Model Design Tool", *Proceedings of the Winter Simulation Conference*: 3-6 December : 408-412

Lorenz, C., 1993,"Time to Get Serious - Rank Xerox's Ambitious Re-engineering Programme", *Financial Times*, 25 June 1993

Love, D.M., 1980,*Aspects of the Design of a Spare Parts Provisioning System*,PhD Thesis, The University of Aston in Birmingham, UK.

Love, D.M., Barton, J.A., Cope, N., 1992,"Whole Business Simulation and Engineering Applications" *Proceedings of the 8th International Conference on Computer-Aided Production Engineering* Edinburgh, UK, August : 166-173

Love, D.M. & Barton, J.A., 1993,"Using Whole Business Simulation to Set Operational Policies in an Integrated Manufacturing System" *Proceedings of the Conference on Managing Integrated Manufacturing* Keele, UK, 22-24 September : 145-157

Lucas Engineering & Systems, 1989,*The Lucas Manufacturing Systems Engineering Handbook*, Institution of Production Engineers.

Maull, R. & Childe, S., 1993,"A Step-by-Step Guide to the Identification of an Appropriate Computer-Aided Production Management System", *Production Planning & Control*, 4(1) : 69-76

Maull, R., Hughes, D., Childe, S., Weston, N., 1990,"A Methodology for the Design and Implementation of Resilient CAPM Systems", *International Journal of Operations & Production Management*, 10(9) : 27-36

McConnell, S., 1993,*Code Complete - A Practical Handbook of Software Construction*, Microsoft Press, USA.

Miller, J.A., Sheth, A.P., Kochut, K.J., Wang, X., Murugan A., 1995,"Simulation Modeling within Workflow Technology", *Proceedings of the Winter Simulation Conference*, 3-6 December : 612-619

Miller, J.G., & Vollman T.E., 1985,"The Hidden Factory", *Harvard Business Review*, 63(5) : 142-150

Mize, J.H. & Pratt, D.B., 1991,"A Comprehensive Object-Oriented Modelling Environment for Manufacturing Systems", *Proceedings of the Joint International Conference on Factory Automation and Information Management* Society of Manufacturing Engineers, March : 250-257

Mould, G., 1990,"Spreadsheet Simulation", *Proceedings of the Society of Computer Simulation Multi-Conference on Simulation in Business and Management*, 3-6 December : 122-127

Murphy, J.C. & Braund, S.L., 1990,"Management Accounting and New Manufacturing Technology", *Management Accounting (London)*, 68(2) : 38-40

Nordgren, B., 1994,"Taylor II Manufacturing Simulation Software", *Proceedings of the Winter Simulation Conference*: 446-449

Norfolk, D., 1995,"Controlling the Flow", *Information Age*, 1(6) : 40-45

Ould, M.A., 1995,*Business Processes - Modelling and Analysis for Re-Engineering and Improvement*, John Wiley & Sons, UK.

Papadopoulos, H.T., Heavey, C., Browne, J., 1993,*Queueing Theory in Manufacturing System Analysis and Design*, Chapman & Hall, London.

Parnaby, J., 1979,"Concept of a Manufacturing System", *International Journal of Production Research*, 17(2) : 123-135

Parnaby, J., 1986,"The Design of Competitive Manufacturing Systems", *International Journal of Technology Management*, 1(3/4) : 385-396

Patching, D., 1994,"Business Process Re-engineering", *Management Services*, 38(11) : 8-11

Pavicic, M.J., 1991,"Making the Transition to an Object-Oriented Simulator", *Proceedings of the Society of Computer Simulation Multi-Conference on Object-Oriented Simulation*: California, USA : 65-71

Peppard, J. & Rowland, P., 1995,*The Essence of Business Process Re-Engineering* Prentice-Hall International (UK) Ltd..

Pfaffenberger, B. & Wall, D., 1995,*Que's Computer & Internet Dictionary* 6th edition, Que Corporation, USA.

Pidd, M., 1992a,*Computer Simulation in Management Science*, 3rd edition, John Wiley & Sons, UK.

Pidd, M., 1992b,"Guidelines for the Design of Data-Driven Generic Simulators for Specific Domains", *Simulation*, 59(4) :237-243

Pidd, M., 1995,"Object-Orientation, Discrete Simulation and the Three-Phase Approach", *Journal of the Operational Research* Society, 46(3) : 362-374

Pidd, M., 1996,"Object-Orientation and Discrete-Event Simulation", *Operational Research Society Simulation Seminar*: 23 May, Aston University, Birmingham, UK.

Plaia, A. & Carrie, A. 1995,"Application and Assessment of IDEF3 - Process Flow Description Capture Method", *International Journal of Operations & Production Management*, 15(1) : 63-73

Probert, D.R., Jones, S.W., Gregory, M.J., 1993,"The Make or Buy Decision in the Context of Manufacturing Strategy Development", *Proceedings of the Institution of Mechanical Engineers : Journal of Engineering Manufacture*, 207 : 241-250

Ptak, C.A., 1991,"MRP, MRPII, OPT, JIT and CIM - succession, evolution or necessary combination", *Production and Inventory Management Journal*, 32(2) : 7-11

Revell, R.J., 1986,*Cemach's Work Study in the Office*, 6th edition, Anbar Publications Ltd., England

Richter J.M., 1994,*Advanced Windows NT : The Developer's Guide to the Win32 Application Programming Interface*, Microsoft Press, Washington, USA.

Robinson, S. 1993,"The Application of Simulation in Manufacturing", *Integrated Manufacturing Systems*, 4(4) :18-23

Rumbaugh, J., Blaha, M., Premerlani, W., Eddy, F., Lorensen, W., 1991,*Object-Oriented Modelling and Design*, Prentice-Hall, USA.

Santosus, M., 1993,"No Pane, No Gain", *CIO Magazine*, 7(3) : 47-52

Saunders, D. & Novels, M., 1994,"Simulation of Teamworking", *Proceedings of the International Conference on Concurrent Engineering and Electronic Design Automation (CEEDA'94)* : 129-134

Seila, A.F., 1995,"Introduction to Simulation" *Proceedings of the Winter Simulation Conference*, 3-6 December : 7-15

Shannon, R.E., 1975,*Systems Simulation : The Art and Science*, Prentice-Hall

Shapiro, R.M., 1994,"Integrating BPR with Image-based Work-Flow", *Proceedings of the Winter Simulation Conference*: 1221-1228

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

Slack, N., Chambers, S., Harland, C., Harrison, A., Johnston, R., 1995,*Operations Management*, Pitman Publishing, London

Snowdon, J.L. & Ammons, J.C., 1988,"A Survey of Queueing Network Packages for the Analysis of Manufacturing Systems", *Manufacturing Review*, 1(1) : 14-25

Spencer, M.S. & Wathen, S., 1994,"Applying the Theory of Constraints' Process Management Technique to an Administrative Function at Stanley Furniture", *National Productivity Review*, 13(3) : 379-385

Spurr, K., Layzell, P., Jennison, L., Richards, N., 1994,*Software Assistance for Business Re-Engineering*, John Wiley & Sons, UK.

Suri, R. & Diehl, G.W., 1987,"Rough-Cut Modeling : An Alternative to Simulation", *CIM Review*, Winter : 25-32

Suri, R., Diehl, G.W.W., de Treville, S., Tomsicek M.J., 1995,"From CAN-Q to MPX : Evolution of Queuing Software for Manufacturing", *Interfaces*, 25(5) : 128-150

Sutton, S.G., 1991,"A New Age of Accounting", *Production and Inventory Management Journal*, 32(1) : 72-75

Szymankiewicz, J., McDonald, J., Turner, K., 1988,*Solving Business Problems by Simulation*, McGraw-Hill, London

Tatikonda, L.U. & Tatikonda, R.J., 1991,"Overhead Cost Control - Through Allocation or Elimination?", *Production and Inventory Management Journal*, 32(1) : 37-41

Taylor, D.A., 1995,*Business Engineering with Object Technology*, John Wiley & Sons, USA.

Thomas, P.V. & Davies, A. 1996,"Remodelling a Company via Systems Re-engineering", *International Journal of Operations & Production Management*, 16(7) : 14-26

Thompson, W.B., 1994,"A Tutorial for Modelling with the WITNESS Visual Interactive Simulator", *Proceedings of the Winter Simulation Conference*: 517-521

Thompson, W.B., 1995,"Introduction to WITNESS and Linking to Process Mapping Tools", *Proceedings of the Winter Simulation Conference*: 3-6 December : 462-466

Tobias, A., 1991,"O.R. Techniques for use in Redesigning Manufacturing and Associated Business Systems", *European Journal of Operational Research*, 51(2) : 168-178

Toelle, R.A. & Tersine, R.J., 1989,"Excess Inventory : Financial Asset or Liability?", *Production and Inventory Management Journal*, 30(4) : 32-35

Tucci, C.L., Lang, J.H., Tabors, R.D., Hirtley, J.L., 1991,"A Simulator of the Manufacturing of Induction Motors" *IEEE Industry Application Society Annual Meeting*, Dearborn, USA : 1353-1359

Tumay, K., 1995,"Business Process Simulation", *Proceedings of the Winter Simulation Conference*: 3-6 December : 55-60

Turner, E. & Saunders, D., 1994,"Surveying the Manufacturing and Control Environment" *Proceedings of the 4th International Conference on Factory 2000 - Advanced Factory Automation*, York, UK, 3-3 October : 613-619

Turner, F., 1994,"Business Systems Engineering", *Proceedings of the Institution of Mechanical Engineers : Journal of Engineering Manufacture*, 208 : 1-7

Underwood, L., 1994,*Intelligent Manufacturing*, Addison-Wesley and The Economist Intelligence Unit

van der Walde, E., 1991,"Computer Simulation in Manufacturing", *Production and Inventory Management Journal*, 32(2) : 80-83

Venkatesan, R., 1992,"Strategic Sourcing : To Make or Not to Make", *Harvard Business Review*, 70(6): 98-107

Wang, W., Popplewell, K., Bell R., 1993,"An Integrated Multi-View System Description Approach to Approximate Factory Modelling", *International Journal of Computer Integrated Manufacturing* 6(3) : 165-174

Ward, P.T., Berger, P.D., Miller J.G., Rosenthal S.R., 1992,"Manufacturing Process Technology and Support Staff Composition : An Empirical View of Industry Evidence", *Production and Operations Management*, 1(1) : 5-21

Wild, R., 1995,*Production and Operations Management*, Cassell Educational Limted, London

Wood, G., 1993,"Using Simulation for Process Improvement", *Management Services*, 37(10): 18-19

Wood, J., 1986,"A Systems Approach to Business Redesign" *Proceedings of the 1st Conference on Just In Time Manufacturing*

Wu, B., 1990, "WIP Cost-related Effectiveness Measure for the Application of an IBE to Simulation Analysis", *Computer-Integrated Manufacturing Systems* 3(3) : 141-149

Wu, B., 1994, *Manufacturing System Design and Analysis* Chapman & Hall, London

Yoon, K.P. & Naadimuthu, G., 1994, "A Make-or-Buy Decision Analysis Involving Imprecise Data", *International Journal of Operations & Production Management* 14(2) : 62-69

Young, R.E., Vesterager, J., Wichmann, K.E., Heide, J., 1988, "Simulation uses in CIM Development", *International Journal of Computer Integrated Manufacturing* 1(1) : 50-54

Appendix 1 : Summary Documentation of WBS/Office Classes

This section contains summary documentation for each WBS/Office class.

Class : tAbstractCompany

This class represent an abstract class used to represent whole business objects at an aggregate level. This class will never be instantiated itself but will be inherited by other more specific company classes such as tSupplier and tCustomer that will be instantiated.

Class : tAbstractCompanyManager

This class manages the functions associated with all objects of the tAbstractCompany class.

Class : tAbstractDataObject

Document data objects represent the 'components' of information contained on document template and document objects in WBS/Office. Each data object represent a value such as a name or dates. This abstract data object class contains methods common to all document data object classes. This class will never be instantiated itself but will be inherited by other more specific data object classes that will be instantiated.

Class : tAbstractDataObjectManager

This class manages the functions associated with all objects of the tAbstractDataObject class.

Class : tAbstractDocument

This class includes all the necessary functionality to let documents physically exist in the simulation., and therefore enables AFS's existing 'document' classes to be processed in the same way as the more generic WBS/Office style document. This class will never be instantiated itself but will be inherited by other more specific document classes that will be instantiated.

Class : tAbstractDocumentManager

This class manages the functions associated with all objects of the tAbstractDocument class.

Class : tAbstractDocumentOperation

This class is holds information and methods common to 'normal' and 'aggregate' document operations classes which inherit this class. This class hasn't got its own manager - management functions (setting & getting data etc.) are performed by the tDocumentRoutingManager.

Class : tAbstractDocumentOpnDataConfiguration

Document operations represents explicit sets of instruction in the form of configuration objects that define how the value for each item of data on a document will be generated during a simulation. The configuration details for each data object will itself be stored as an object. The tAbstractDocumentOpnDataConfiguration class is an abstract class that holds attributes and methods shared by the actual configuration classes. This class will never be instantiated but will be inherited by other configuration classes.

Class : tAbstractOfficeSkillAction

Document action classes provide links between models of administrative processes and other parts of the model by enabling other entities to take some course of action as a result of a receiving a certain type of document. This aspect of WBS/Office offers the most potential for future expansion to reflect specific modelling needs but model-builder will not even be aware that this class exists. This abstract 'action' class contains attributes and methods shared by other action classes. This class will never be instantiated.

Class : tAbstractTransportHandling

Transportation functionality is represented in WBS/Office as a set of sub-classes or transportation classes. These provide the basic mechanisms required for modelling any type of transport, whether it be documents or production materials, and can be incorporated into skill classes or other 'intelligent' classes. The 'transport handling' classes enable a person to physically add and remove items from a storage area or transport device.

This abstract version of the 'transport handling' mechanism contains attributes and methods shared by other transport handling classes. This class will never be instantiated.

Class : tAbstractTransportLogic

Transportation functionality is represented in WBS/Office as a set of sub-classes or transportation classes. These provide the basic mechanisms required for modelling any type of transport, whether it be documents or production materials, and can be incorporated into skill classes or other 'intelligent' classes. The 'transport logic'

classes represent the 'intelligent' part of the transport mechanism skill and enables the person or object using this functionality to determine what to do next in terms of where to go and what to transport.

This abstract version of the 'transport logic' mechanism contains attributes and methods shared by other transport logic classes. This class will never be instantiated.

Class : tAbstractTransportSkill

This class contains functionality that is share by skill needed to transport document and materials within the model

Class : tAbstractTransportSkillManager

This class manages the functions associated with all objects of the tAbstractTransportSkill class.

Class : tAbstractWBSObjectDataObject

This is an abstract data object class. This class will never be instantiated but will be inherited by other data object classes that represent data values which correspond to other objects taking part in the simulation.

Class : tAggregateDocumentOperation

The tAggregateDocumentOperation class represents a special type of document operations that allows less detailed modelling of document processing activities such as those performed by tSupplier or tCustomer objects - in practice it may not be appropriate or even possible to model document processing in other firm's to the same level of detail that they are represented in the main company being modelled

Class : tAuthorisationDataObject

This class represents 'Authorisation' data objects. These corresponds to information contained on documents that controls the progress of those document through an organisation. For example this type of data object might be used to represent an authorisation or approval for a purchase requisition.

Class : tAuthorisationDataObjectManager

This class manages the functions associated with all objects of the tAuthorisationDataObject class.

Class : tCommentDataObject

This class represents 'text' document data objects. These represent text-based information contained on documents such as miscellaneous comments. Required to enable conversion of original AFS document classes to more generic WBS/Office-style documents.

Class : tCommentDataObjectManager

This class manages the functions associated with all objects of the tCommentDataObject class.

Class : tCustomer

This class is used to represent customers of the main company being modelled. Instances of this class can 'purchase' a defined range of finished goods produced by the company and generate sales order related documentation for these goods.

Class : tCustomerManager

This class manages the functions associated with all objects of the tCustomer class.

Class : tDateDataObject

This class represents 'Date' document data objects. These represent date information contained on documents such as DueDate or LaunchDate etc.

Class : tDateDataObjectManager

This class manages the functions associated with all objects of the tDateDataObject class.

Class : tDocumentOperation

The tDocumentOperation class holds information about document operations. These are analogous to administrative procedures and define how and where a document should be processed.

Class : tDocumentOpnLookedUpDataConfiguration

This class refers to information that administrators will obtain from a database, log-book or similar look-up device.

Class : tDocumentOpnRequestedDataConfiguration

The tDocumentOpnRequestedDataConfiguration class represents information that must be formally requested from another company or another function within the

same business.

Class : tDocumentOpnTranscribedDataConfiguration

The tDocumentOpnTranscribedDataConfiguration class represents information that is copied or transcribed from one document to another.

Class : tDocumentRouting

The document routing class stores the list of document operations that are carried out to produce a document. The class allows operations to be added (at any point) and deleted. The routing will be able to be modified at will - no special pre-cautions will be made - therefore users of the routing will have to make a copy. Copies can be made on request but will not be placed on the master list to avoid confusion.

Class : tDocumentRoutingManager

This class manages the functions associated with all objects of the tDocumentRouting and tDocumentOperation classes.

Class : tDesk

This tDesk class represents a physical location where office task such as processing documents would be performed. Document storage areas (InTrays, OutTrays and PendingTrays) will be associated with each desk object.

Class : tDeskManager

This class manages the functions associated with all objects of the tDesk class.

Class : tDocument

This class represents actual document object that will be created during the course of a simulation. Each instance of this class will have a document template associated with it to define all its data components.

Class : tDocumentArchive

Document Archive class. This class provide a means of storing old documents that have served their purpose in the simulation but which may need to be retrieved at some later date (perhaps as part of results analysis).

Class : tDocumentArchiveManager

This class manages the functions associated with all objects of the

tDocumentArchive class.

Class : tDocumentManager

This class manages the functions associated with all objects of the tDocument i.e. the actual documents that will carry information during a simulation

Class : tDocumentStore

This is a office storage class. It is specifically designed to hold documents. Documents can be added / removed / queried.

Class : tDocumentStoreManager

This class manages the functions associated with all objects of the tDocumentStore class.

Class : tDocumentTemplate

The tDocumentTemplate class is a key feature of WBS/Office. This class allows model-builders to represent exactly the documents (such as purchase orders) used in a particular organisation in a model of that organisation.

Class : tDocumentTemplateManager

This class manages the functions associated with all objects of the tDocumentTemplate. class

Class : tDocumentTransportHandling

Transportation functionality is represented in WBS/Office as a set of sub-classes or transportation classes. These provide the basic mechanisms required for modelling any type of transport, whether it be documents or production materials, and can be incorporated into skill classes or other 'intelligent' classes. This document transport handling class enable a person to physically add and remove documents from a storage area, and carry them to their destination.

Class : tDocumentTransportLogic

Transportation functionality is represented in WBS/Office as a set of sub-classes or transportation classes. These provide the basic mechanisms required for modelling any type of transport, whether it be documents or production materials, and can be incorporated into skill classes or other 'intelligent' classes. This version document-specific 'transport logic' classes enables the person or object using this functionality to determine what to do next in terms of where to go and which documents to

transport.

Class : tDocumentTransportSkill

The tDocumentTransportSkill class enables person objects to transport or move documents within the model. This class inherits and adds to the functionality of the abstract transport skill class. This class is a skill type class and therefore object of this class can be placed on an operator's skill list.

Class : tDocumentTransportSkillManager

This class manages the functions associated with all objects of the tDocumentTransportSkill class.

Class : tNumericDataObject

This class represents 'numeric' document data objects. These represent numeric information contained on documents such as QntyToOrder.

Class : tNumericDataObjectManager

This class manages the functions associated with all objects of the tNumericDataObject class.

Class : tOfficeSkill

The tOfficeSkill class is a generic class that can be attributed to any tPerson object and enables that person to process documents (i.e. to perform document operations on documents). The model-builder will configure each instance of this class to reflect which particular office tasks the person is responsible for carrying out.

Class : tOfficeSkillAction_MakeParts

Document action classes provide links between models of administrative processes and other parts of the model by enabling other entities to take some course of action as a result of a receiving a certain type of document. This aspect of WBS/Office offers the most potential for future expansion to reflect specific modelling needs but model-builder will not even be aware that this class exists. The tOfficeSkillAction_MakeParts class enables tPerson or other 'intelligent' objects such as tSupplier object to generate parts as a result of receiving a document with certain types of information on it.

Class : tOfficeSkillConfiguration

The tOfficeSkillConfiguration class represents one part of the logic used by the generic tOfficeSkill class. This part of the logic maintains information about which document operation a tPerson object is responsible for carrying out.

Class : tOfficeSkillLogic

The tOfficeSkillLogic class represents one part of the logic used by the generic tOfficeSkill class. This part of the logic represents the 'intelligent' part of the skill and enables the person to interpret document operations.

Class : tOfficeSkillManager

This class manages the functions associated with all objects of the tOfficeSkill class.

Class : tOfficeSkillProcess

The tOfficeSkillProcess class represents one part of the logic used by the generic tOfficeSkill class. This part of the logic provides the functionality for actually carrying out whatever processing tasks have been identified by the other office skill component objects. classes.

Class : tTransportConfiguration (developed by P.D.Ball)

Transportation functionality is represented in WBS/Office as a set of sub-classes or transportation classes. These provide the basic mechanisms required for modelling any type of transport, whether it be documents or production materials, and can be incorporated into skill classes or other 'intelligent' classes. The transport configuration class defines where and when a person will go to find items to be transported.

Class : tStartTime tStartTimeList

The tStartTime class allows a series of times to be defined which may be used by a variety of other classes for starting tasks. For example people/skills may use a time series to set times during each day when a particular task must be carried out.

Class : tStartTimeListManager

This class manages the functions associated with all objects of the tStartTime & tStartTimeList classes.

Class : tSupplier

This class is used to represent suppliers of the main company being modelled. Instances of this class can 'sell' a defined range of raw materials or component items

required by the company and can also respond to purchase order related documentation for these items.

Class : tSupplierManager

This class manages the functions associated with all objects of the tSupplier class.

Class : tWBSObjectNameDataObject

This class inherits the tAbstractWBSObjectDataObject class and represents 'name' data objects. In other words, data object of this type will represent the names of other objects taking part in the simulation such as tPartNumber or tSupplier.

Class : tWBSObjectNameDataObjectManager

This class manages the functions associated with all objects of the tWBSObjectNameDataObject class.

Unit : OfficeGlobalTypeDefinitions

This unit provides user-defined types for global use.

Unit : OfficeModel

Object initialisation unit for WBS/Office. This unit loads up new office objects when the user creates a new model. For example, the system generates default document template objects that model-builders can then edit or remove to reflect the actual documents used in the particular firm being modelled.

Unit : OfficeStringResourceLoader

This unit provides look-up table for document data object types. It is also used to dynamically loads strings relating to 'Office' classes from a string resource file to minimise the effects of Windows 3.1 64K common data segment memory limitation.

Appendix 2 : WBS/Office Class Hierarchy

The preceding section provided a summary for each of the new simulation classes that together form WBS/Office. This section will illustrate how these classes have been implemented in the form an object hierarchy. The illustration emphasises the inheritance relationships between the various classes in the simulator.

The illustration also includes classes that originate from the original AFS class hierarchy. Some of these classes (marked *) have been include for clarity, while others (marked #) have been included because their original structure (as defined by AFS) has had to be modified in some way to accommodate new WBS/Office functionality.

Simulation Classes :

tRootAFS *
- tAbstractDataObject
  - tAbstractWBSObjectDataObject
    - tWBSObjectNameDataObject
  - tAuthorisationDataObject
  - tCommentDataObject
  - tDateDataObject
  - tNumericDataObject
- tAbstractOfficeSkillAction
  - tOfficeSkillAction_MakeParts
- tAbstractSkill *
  - tOfficeSkill
  - tAbstractTranportSkill
    - tDocumentTransportSkill
    - tMaterialTransportSkill #
- tAbstractTransportHandling
  - tBatchTransportHandling #
  - tDocumentTransportHandling
- tAbstractTransportLogic
  - tDocumentTransportLogic
  - tMaterialTransportLogic #
- tDocumentTemplate
- tAbstractDocumentOpnDataConfiguration
  - tDocumentOpnLookedUpDataConfiguration
  - tDocumentOpnRequestedDataConfiguration
  - tDocumentOpnTranscribedDataConfiguration
- tOfficeSkillLogic
- tOfficeSkillProcess
- tOperationBasic *

tOperationNumber *
tAbstractDocumentOperation
tAggregateDocumentOperation
tDocumentOperation
tRoutingBasic *
tDocumentRouting
tSimulation *
tAbstractWorkingPattern *
tStartTimeList
tPhysical *
tAbstractDocument
tDocument
tOrder #
tRawMaterialRequisition #
tWorksOrder #
tAbstractStore *
tStorage *
tDocumentArchive
tDocumentStore
tActivity *
tOperationStage *
tAbstractCompany
tCustomer
tSupplier
tWorkStation *
tDesk

Simulation Class Manager Classes :

tRootAFS *
tRootManager *
tAbstractDataObjectManager
tAuthorisationDataObjectManager
tCommentDataObjectManager
tDateDataObjectManager
tNumericDataObjectManager
tWBSObjectDataObjectManager
tAbstractSkill Manager *
tOfficeSkillManager
tAbstractTranportSkillManager
tDocumentTransportSkillManager
tMaterialTransportSkillManager #
tDocumentTemplateManager
tRoutingBasicManager *
tDocumentRoutingManager
tSimulationManager *
tPhysicalManager *
tAbstractDocumentManager

tDocumentManager
tAbstractStore *
tDocumentArchiveManager
tDocumentStorelManager
tActivityManager *
tOperationStageManager *
tAbstractCompanyManager
tCustomerManager
tSupplierManager
tWorkStationManager *
tDeskManager
tStartTimeListManager

Appendix 3 : Document Data Types

A particularly novel feature of WBS/Office's design is the ability for users to model a firm's specific information and document processing requirements. This is achieved through the application of the Document Data Object concept whereby users construct representations of the particular documents used in the company being modelled from a range of available data types. A full listing of all data-types currently available in WBS/Office is provided below. It is expected that this aspect of the simulator offers potential for expansion to fulfil other specific modelling requirements.

tDateDataObject types :

- doDueDate
- doIssueDate
- doLaunchDate
- doCompletionDate

tCommentDataObject types:

- doComment

tNumericDataObject types :

- doQntyToOrder
- doQntyCompleted
- doQntyOutstanding
- doLeadTimeDays
- doPartPricePerUnit
- doAmountDue

tAuthorisationDataObject types :

- doAuthorisation_1
- doAuthorisation_2
- doAuthorisation_3
- doInspected

doChecked

tWBSObjectNameDataObject types :

doPartNumber

doSupplier

doRawMatlPartNumber

doLocationStore

doDestinationStore

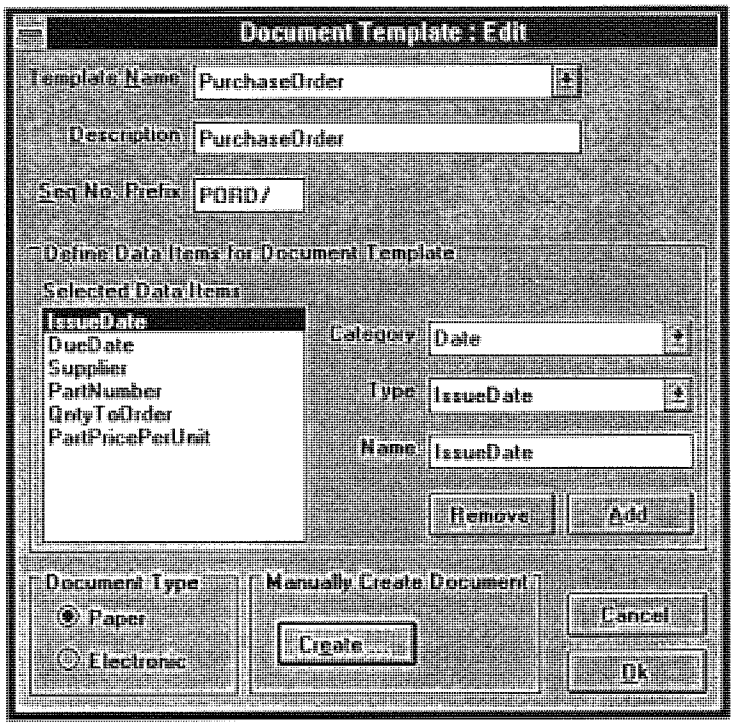
doReferenceDcmt_1

doOrder

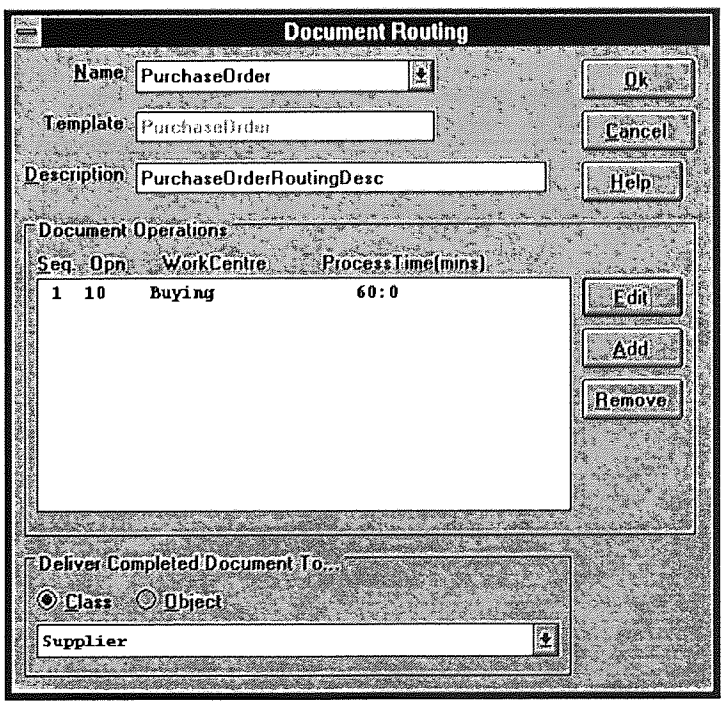
doPartRouting

Appendix 4 : Illustrated User-Interface Dialogs

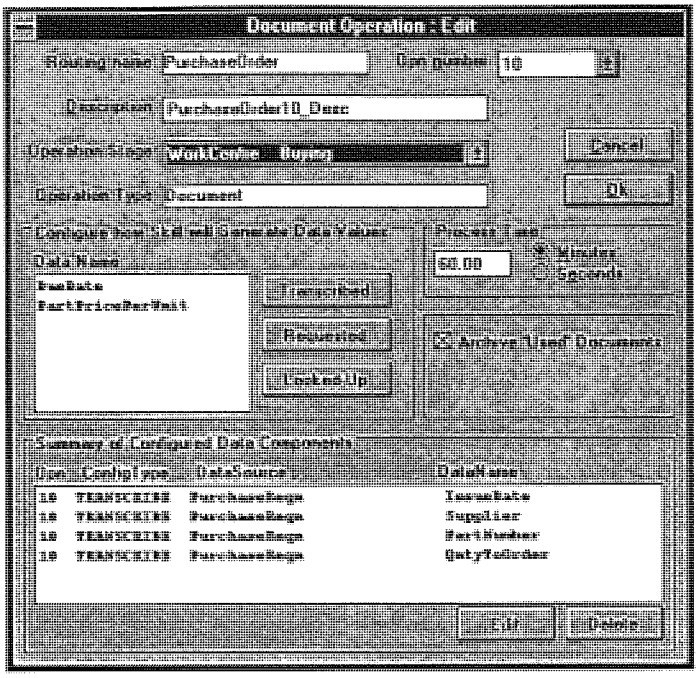
This section illustrates the user-interface dialogs developed specifically to support the new WBS/Office classes.



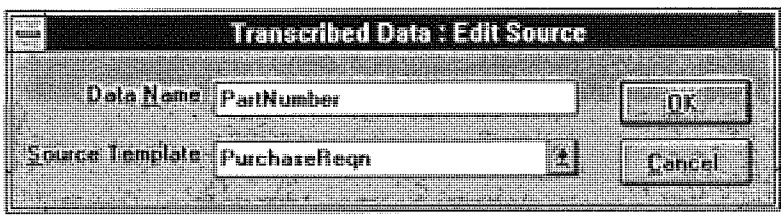
The user can describe the different types of document used in the firm being modelled as document templates (shown above).

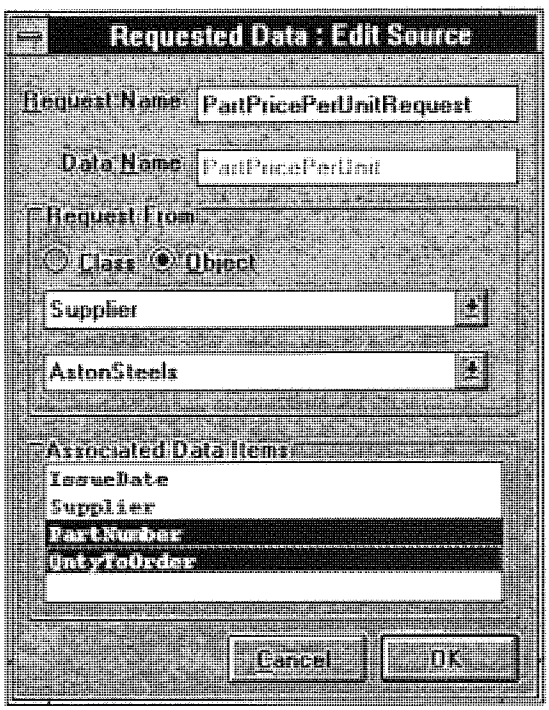


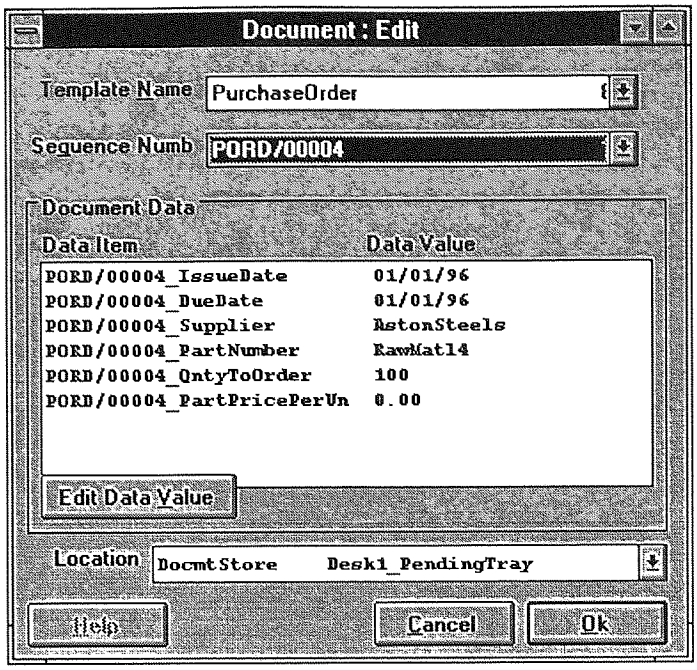
The processing requirements for each document are defined in terms of a document routing.



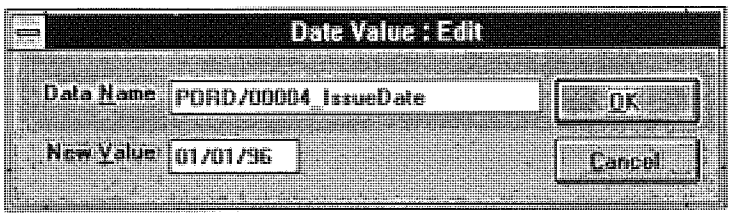
A document routing is a collection of document operations. Each operation describes how a particular item of information is obtained, which ultimately affects how long that document will take to process during the simulation. Potential sources of information are categorised as either being transcribed, requested or looked-up:

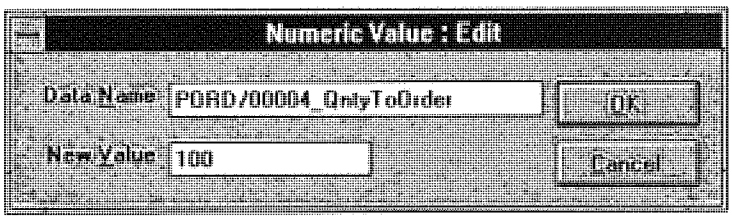


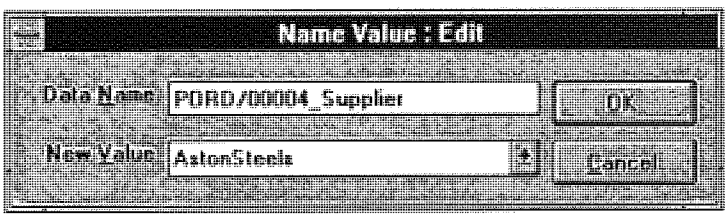


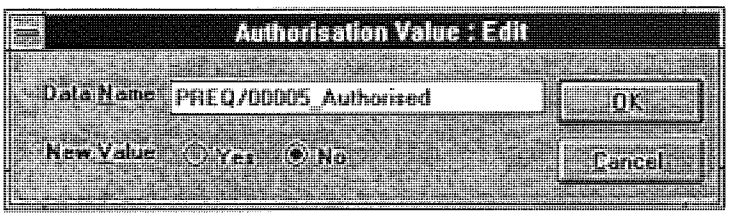


Document objects will then be created during a simulation according to their routing to convey information between different parts of the model. Model-builders can also modify the contents of a document 'on-the-fly' to affect progress of the simulation.

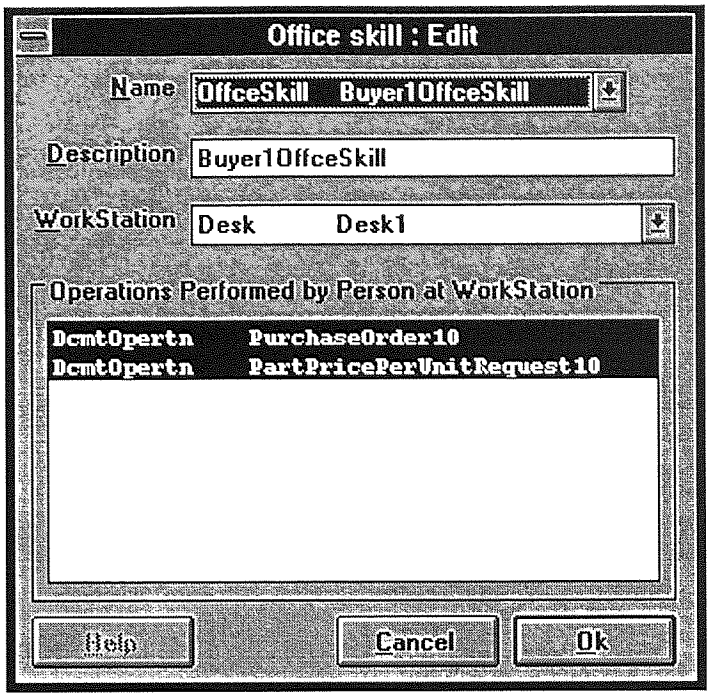




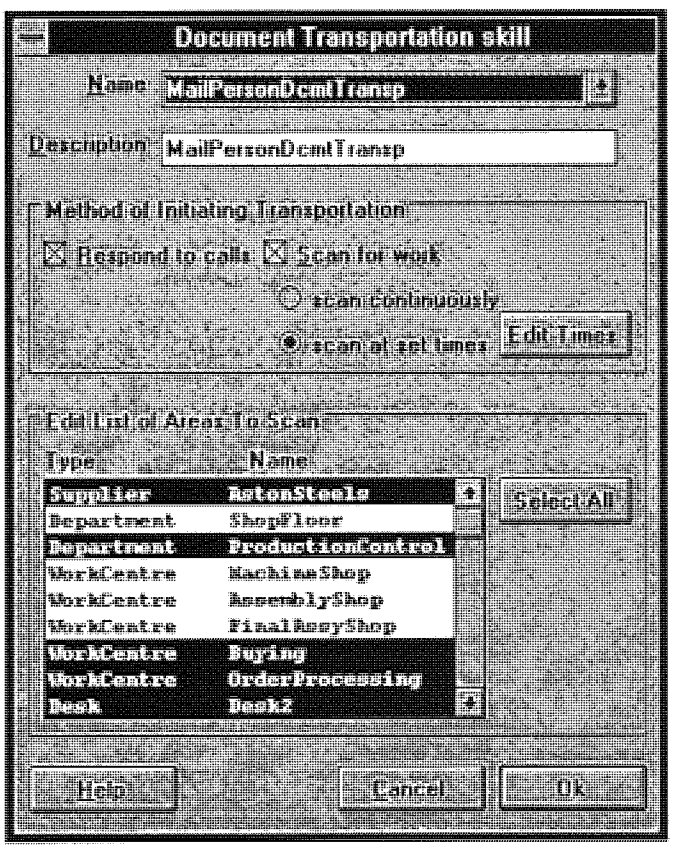




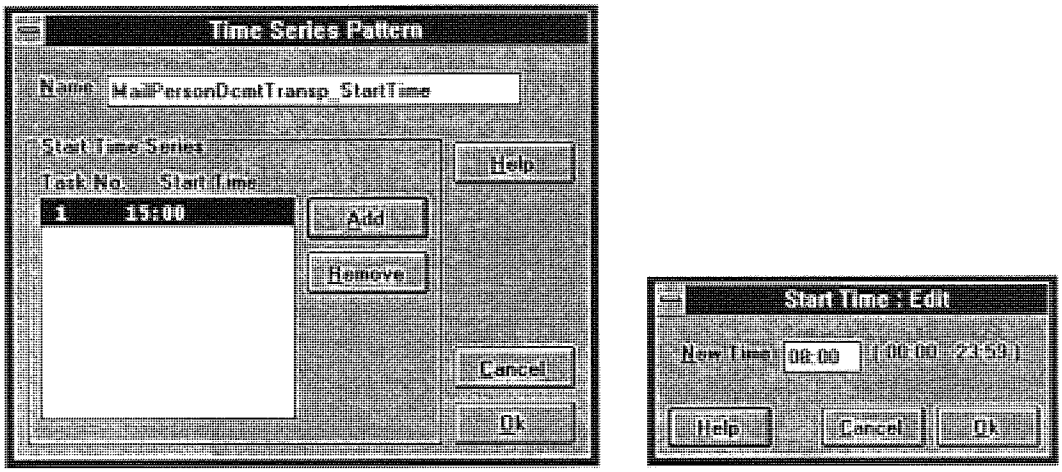
Representations of people are used to process and move documents around the model of a business. This is achieved by attributing different 'skills' to person objects:



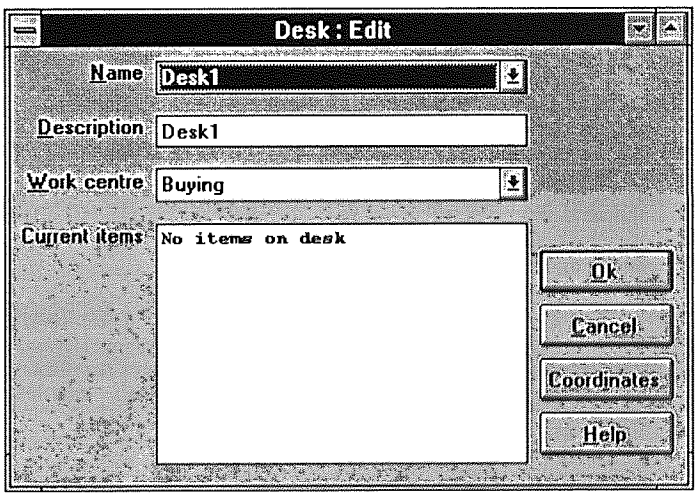
The Office Skill (shown above) enables individual person objects to process certain types of documents.



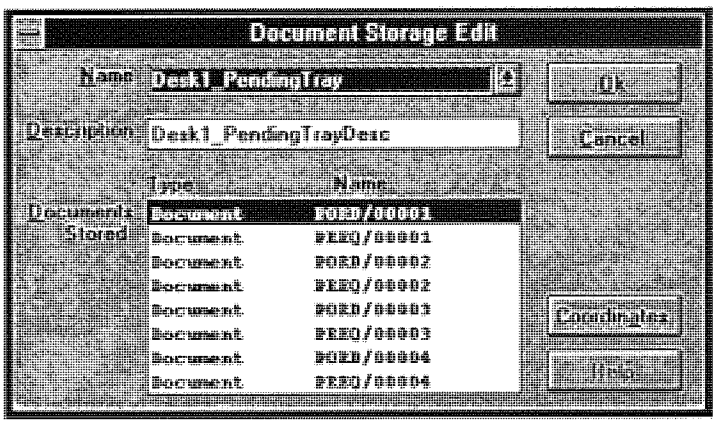
The Document Transport skill enables person objects to move documents around the business. In this way, the model-builder can define set times of the day when a 'mail-person' will collect 'post' from various departments and take each item to its appropriate destination:



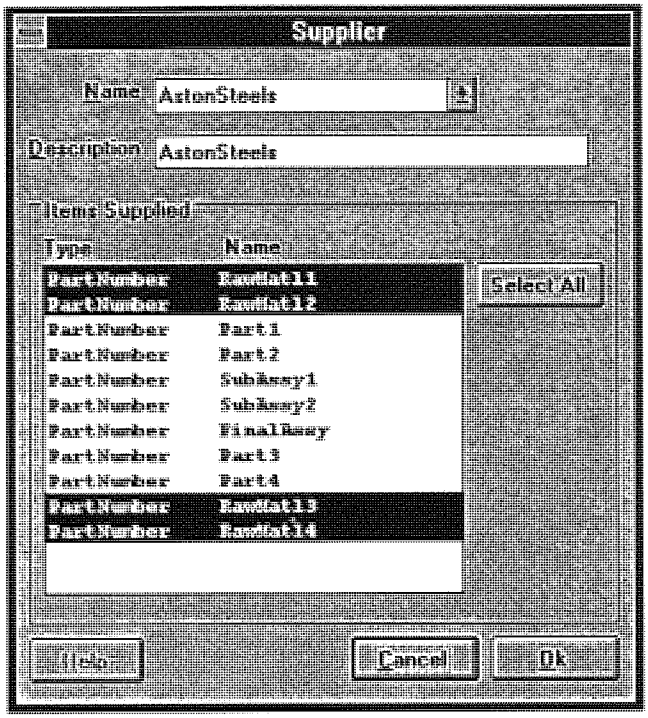
People attributed the office skill will process documents at desks,



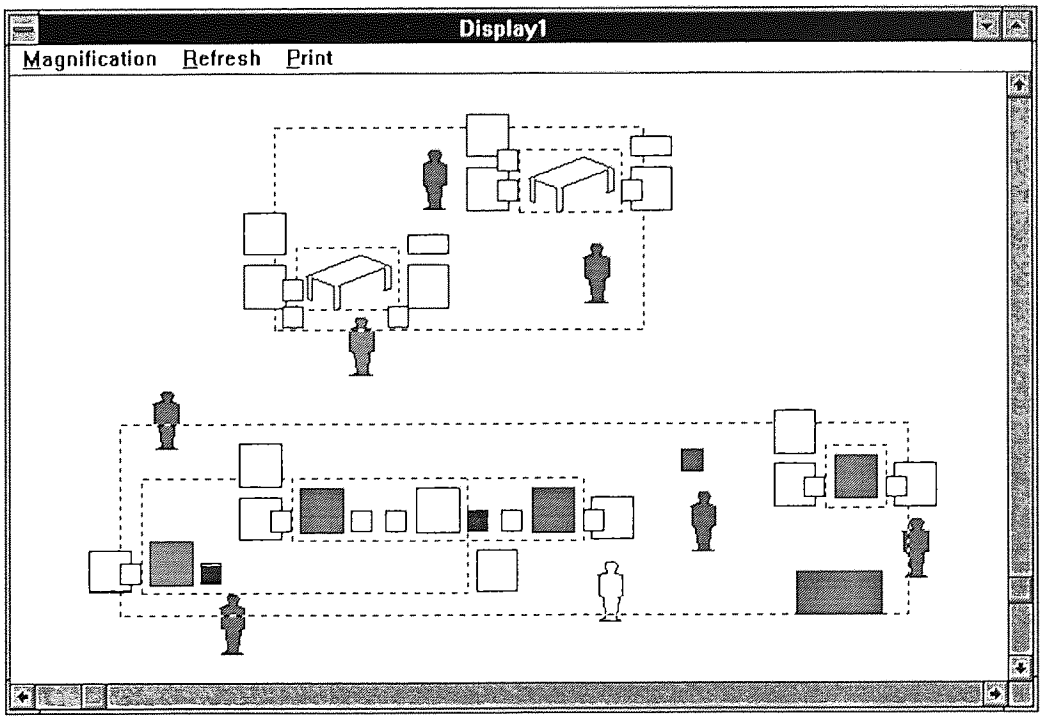
and keep completed or part-completed documents in storage areas, typically 'in-trays', or document archives.



Suppliers or other 'intelligent' objects can also process documents, and take some appropriate action as a consequence of receiving certain types of document, such as delivering parts to stores:



WBS/Office re-uses an existing graphical simulation library that forms part of the Advanced Factory Simulator (AFS). A modeler can therefore follow the progression of a simulation experiment by watching the interactions of the different entities in the model:



By re-using the functionality of AFS, WBS/Office also includes many other dialogs for defining the material processing aspects of a manufacturing system. Furthermore, when a full implementation of the system's design is completed, WBS/Office will also include production control system dialogs derived from WBS/Control. The reader is referred to Ball (1994) and Boughton (1995) for details of this aspects of the user-interface.

Appendix 5 : Hardware and Software Requirements

A5.1 Requirements for the Development of WBS/Office

The following table details the commercial software used to implement the object-oriented design of WBS/Office:

Application	**Development Software**
WBS/Office core simulator	Borland Pascal with Objects, version 7.0
WBS/Office user-interface	Microsoft Visual Basic, version 3.0, professional edition

An important feature of WBS/Office's object-oriented design is the potential for developers to expand the system at some point in the future to reflect other specific modelling needs. In addition to the commercial development software described above, the following hardware specification is recommended as the minimum need for further development of the WBS/Office class library:

Hardware Type	IBM-Compatible Personal Computer (PC)
Processor	Intel 486DX2 66Mhz
Memory	8 Mb RAM
Available Hard Disk Space	30 Mb
Operating System	Microsoft Windows, version 3.1 or Microsoft Windows 95

A5.1 Requirements for Using WBS/Office

The following table details all of the main programs and locations needed to run WBS/Office. The listing assumes that the simulator is being installed on Drive C of a user's computer:

File Name	Description	File Location
afscore.exe	core simulator	c:\wbsoffc\system
afsgraph.exe	graphical display	c:\wbsoffc\system
afsintf.exe	user-interface	c:\wbsoffc\system
icondll.dll	dynamic-link library used by graphical display	c:\wbsoffc\system
config.txt	object message configuration file	c:\wbsoffc\system
afshelp.hlp	user help system	c:\wbsoffc\help

The following table details all of additional software components and programs also need to run WBS/Office. This files must be located in the computer's 'windows' directory, typically 'c:\windows\system' or 'c:\win95\system':

Related to	Description	Required Files
Graphics Server, version 2.2	required for graphical display	gsw.exe, gswdll.dll, graph.vbx
Microsoft Visual Basic, version 3.0, professional edition	basic interpreter and additional visual component for user interface	vbrun300.ll, cmdialog.vbx, mmmasked.vbx, spin.vbx, grid.vbx, gauge.vbx
Borland Pascal with Objects, version 7.0	dynamic-link library used by the DDE client-server mechanism that enables the 3 main parts of WBS/Office to interact	ddeml.dll

The minimum hardware requirements for constructing and experimenting with WBS/Office simulation models is as follows:

Hardware Type	IBM-Compatible Personal Computer (PC)
Processor	Intel 486DX2 66Mhz
Memory	8 Mb RAM
Available Hard Disk Space	10 Mb
Operating System	Microsoft Windows, version 3.1 or Microsoft Windows 95

Appendix 6 : Simulation Models Constructed Using WBS/Office

This section describes an example application of WBS/Office. The demonstration involves the creation of two simulation models; the first, a *factory* model, evaluates only the shop-floor aspects of the manufacturing system while the second, a *combined* model, also includes support activities. Specifically, the support activity model will be used to simulate the processing of works order and purchase order related documentation. For clarity both models have intentionally been kept quite simple and include synthetic rather than real company data. Nevertheless, they serve as interesting and very important examples of contrasting approaches to manufacturing system analysis. The discussion that follows will relate mainly to features of the combined model. Results will then be presented for comparison.

The list below summarises the different types of object included in the two models before simulating. In other words, the list only includes static objects or entities created by the model-builder; other objects were dynamically generated by WBS/Office during the simulation but will be discussed later. The static objects marked '*' in the list are unique to the combined model and represent the additional functionality provided by WBS/Office for modelling administrative activities. All other objects shown are common to both models and represent the entities involved in simulating material processing activities.

```
WBS/Office : Model data.  Date: 27/09/96. Time: 22:00.
-------------------------------------------------------------------------
Model Name : Demonstration Model
Model Description : Model created using synthetic data to demonstrate
the functionality of WBS/Office.  The model includes a machine shop,
a production control department

Class: Department
-----------------------------------------------
{object/entity name}            {object/entity description}
ShopFloor                       manufacturing system
ProductionControl *             support function

Class: WorkCentre
-----------------------------------------------
MachineShop                     manufactures components
AssemblyShop                    manufactures sub assemblies
FinalAssyShop                   manufactures finished goods
Buying *                        generates purchase orders
OrderProcessing *               generates works orders

Class: Supplier
-----------------------------------------------
AstonSteels *                   supplier for all raw materials
```

Class: MachManual

MachManual1	Manually Operated Machine No.1
MachManual2	Manually Operated Machine No.2

Class: AssyManual

AssyManual1	Manual Assembly Machine No.1
AssyManual2	Manual Assembly Machine No.2
AssyManual3	Manual Assembly Machine No.3

Class: Desk

Desk2 *	location for administrative work
Desk1 *	location for administrative work

Class: HandTruck

HandTruck	HandTruck for moving shop-floor items

Class: Stores

CentralStores	central storage area for all production items

Class: BatchStore

MachineShop_InStore	local part storage
MachineShop_OutStore	local part storage
AssemblyShop_InStore	local part storage
AssemblyShop_OutStore	local part storage
FinalAssyShop_InStore	local part storage
FinalAssyShop_OutStore	local part storage
Buying_InStore	local part storage
Buying_OutStore	local part storage
OrderProcessing_InStore	local part storage
OrderProcessing_OutStore	local part storage
MachManual1_In	local part storage
MachManual1_Out	local part storage
MachManual2_In	local part storage
MachManual2_Out	local part storage
AssyManual1_In	local part storage
AssyManual1_Out	local part storage
AssyManual2_In	local part storage
AssyManual2_Out	local part storage
AssyManual3_In	local part storage
AssyManual3_Out	local part storage
AstonSteels_InBatches	local part storage
AstonSteels_OutBatches	local part storage

Class: DocmtStore

ShopFloor_InTray *	local document storage
ShopFloor_OutTray *	local document storage
ShopFloor_PendingTray *	local document storage
ProductionControl_InTray *	local document storage
ProductionControl_OutTray *	local document storage
ProductionControl_Pending *	local document storage
AssemblyShop_DcmtIn *	local document storage
FinalAssyShop_DcmtIn *	local document storage
Buying_DcmtIn *	local document storage
Buying_DcmtOut *	local document storage
OrderProcessing_DcmtIn *	local document storage
OrderProcessing_DcmtOut *	local document storage
Desk1_InTray *	local document storage
Desk1_OutTray *	local document storage
Desk1_PendingTray *	local document storage

```
Desk2_InTray *                  local document storage
Desk2_OutTray *                 local document storage
Desk2_PendingTray *             local document storage
AstonSteels_InDcmts *           local document storage
AstonSteels_OutDcmts *          local document storage
AstonSteels_PendingDcmts *      local document storage

Class: DcmtArchve
-----------------------------------------------
DocumentArchive *               Maintains archive of 'used' documents

Class: Operator
-----------------------------------------------
MachOperator1                   Manual Machine Operator No.1
MachOperator2                   Manual Machine Operator No.2
AssyOperator1                   Assembly Machine Operator No.1
AssyOperator2                   Assembly Machine Operator No.2
AssyOperator3                   Assembly Machine Operator No.3
MatlHandler                     Material Handler
Buyer1 *                        processes purchase orders
ProdnController *               processes works orders
MailPerson *                    moves documents within model

Class: CellLeader
-----------------------------------------------
CellLeader                      manages shop-floor activities

Class: Shifts
-----------------------------------------------
Shift                           default operator shift : 5 days/wk

Class: PartNumber
-----------------------------------------------
RawMat11                        Raw Material for Part1 - purchased
RawMat12                        Raw Material for Part2 - purchased
RawMat13                        Raw Material for Part3 - purchased
RawMat14                        Raw Material for Part4 - purchased
Part1                           Component - made in house
Part2                           Component - made in house
Part3                           Component - made in house
Part4                           Component - made in house
SubAssy1                        assembled in house
SubAssy2                        assembled in house
FinalAssy                       finished product - assembled in house

Class: DcmtTemplt
-----------------------------------------------
PurchaseReqn *                  request for bought in items
PurchaseOrder *                 order for bought in items
RawMaterialReqn *               request items from stores
WorksOrder *                    order to manufacture items
Order *                         Order
SalesOrder *                    customer orders
PartPricePerUnitRequest *       formal request sent to supplier

Class: DcmtRoutng
-----------------------------------------------
PurchaseReqn *                  processing details for document
PurchaseOrder *                 processing details for document
RawMaterialReqn *               processing details for document
WorksOrder *                    processing details for document
Order *                         processing details for document
SalesOrder *                    processing details for document
PartPricePerUnitRequest *       processing details for document

Class: Document
-----------------------------------------------
PREQ/00001 *                    instance of purchase requisition
```

```
PREQ/00002 *                    instance of purchase requisition
PREQ/00003 *                    instance of purchase requisition
PREQ/00004 *                    instance of purchase requisition
Sa/0000001 *                    instance of customer order
Sa/0000002 *                    instance of customer order
Sa/0000003 *                    instance of customer order
Sa/0000004 *                    instance of customer order
Sa/0000005 *                    instance of customer order
Sa/0000006 *                    instance of customer order
Sa/0000007 *                    instance of customer order

Class: OrderSystm
-----------------------------------------------
OrderSystem                     monitors works order status

Class: WIPTracker
-----------------------------------------------
WIPTracker                      monitors WIP batches
```

Since WBS/Office does not yet include production control system functionality, documents that might normally be generated by this type of entity have had to be created manually as part of the model-building process for the combined model, Consequently, the object listing include several instances of the 'document' class. Specifically, the document objects correspond to customer orders and purchase requisitions, and thereby represent the initial workload placed on support functions in this model. Table A6.1 illustrates the procedures, or 'document routings', defined as part of the model for generating purchase orders during the simulation. These purchase order objects are ultimately required to obtain raw materials from the supplier object.

The illustration shows that in this particular model, a buyer is triggered to create a purchase order on receiving a purchase requisition. In fact, the purchase requisition provides most of the information that the *buyer* needs to generate the order (this information is 'transcribed' onto the order) but the *buyer* must make a formal request to the supplier to get the relevant price per unit for the ordered parts.

Figure A6.1 graphically 'walks-through' this aspect of the model, and demonstrates the various interactions and delays involved. A similar type of procedure is also included in the model that enables the *ProdnController* object to generate shop-floor works orders based on customer order objects.

During the simulation, the combination of delays that these different interactions represent affects the availability of production orders and parts in the manufacturing system, and therefore impacts on the performance of this aspect of the combined

model. Figures A6.2 and A6.3, for example, illustrate selected results produced by simulating the factory model and the combined model respectively for a period of one week.

PurchaseRequisition, created by model-builder in this case		
Data Object	Data Source	Performed By
IssueDate	not applicable, user generated	model-builder
Supplier	not applicable, user generated	model-builder
PartNumber	not applicable, user generated	model-builder
QntyToOrder	not applicable, user generated	model-builder
PurchaseOrder, Dcmt Opn 10, Process Time = 60 minutes		
Data Object	Data Source	Performed By
IssueDate	TRANSCRIBED from PurchaseRequisition	Buyer
Supplier	TRANSCRIBED from PurchaseRequisition	Buyer
PartNumber	TRANSCRIBED from PurchaseRequisition	Buyer
QntyToOrder	TRANSCRIBED from PurchaseRequisition	Buyer
PartPricePerUnit	REQUESTED from Supplier	Buyer
PartPricePerUnitRequest, Dcmt Opn 10, Process Time = 30 minutes		
Data Object	Data Source	Performed By
PartNumber	TRANSCRIBED from PurchaseRequisition	Buyer
QntyToOrder	TRANSCRIBED from PurchaseRequisition	Buyer
PartPricePerUnitRequest, Dcmt Opn 20, Process Time = 60 minutes		
PartPricePerUnit	LOOKED-UP	Supplier

Table A6.1 Document Routings Involved in the Purchase Order Process

In both cases, the simulation begins on '1/1/96', and clearly in the first model this implies that production orders were issued or 'created' immediately for the shop-floor (that is, on '1/1/96') which could then begin production. In contrast, the combined model actually simulated the creation of these works orders from customer order information, and so the works order documents which were not actually prepared until part way through the simulation period. This constrained the start of production. The modelling of delays involved in the purchase order process (that is, the ordering raw materials from the supplier) also contributed to the shop-floor model not being able to commence production until '5/1/96'.

In effect, support department constraints included in the combined model were able to hamper the final yield of the manufacturing system over the course of the simulation period. The results of the combined model therefore demonstrate the feasibility of evaluating manufacturing system designs in WBS/Office in a way that the research has established not to be possible using other simulation methods.

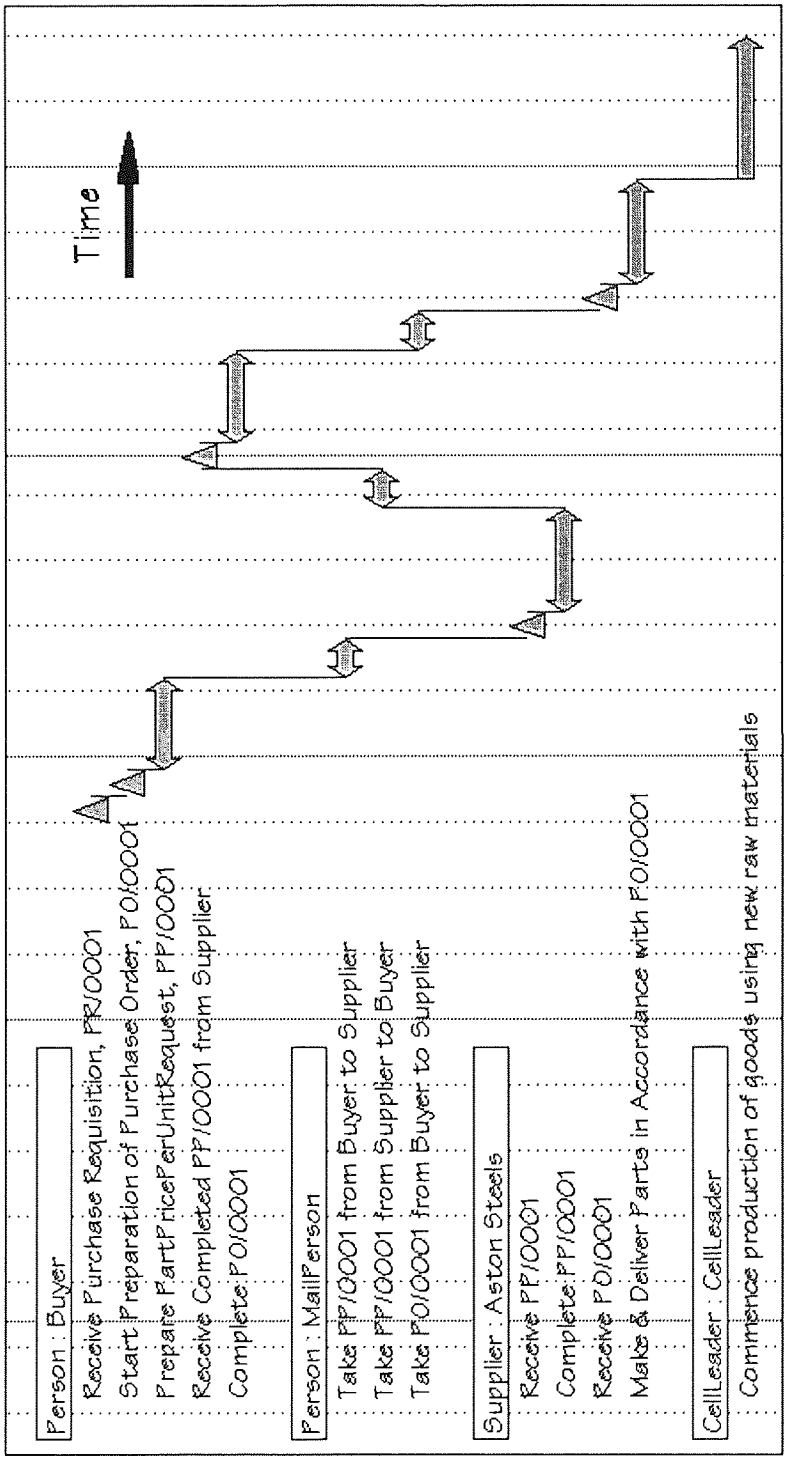


Figure A6.1 'Walk-through' of Purchase Order Process in Combined Model (not drawn to scale)

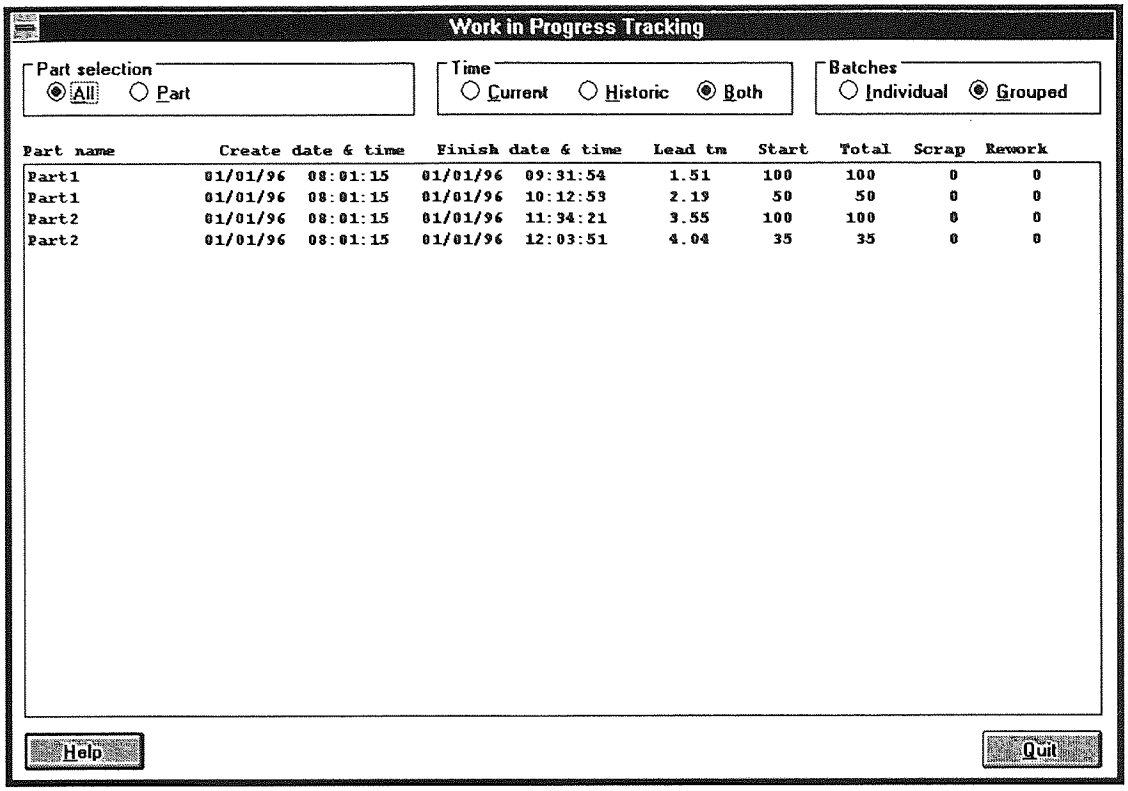


Figure A6.2 Selected Results from the Factory Model (based on 1 week simulation)

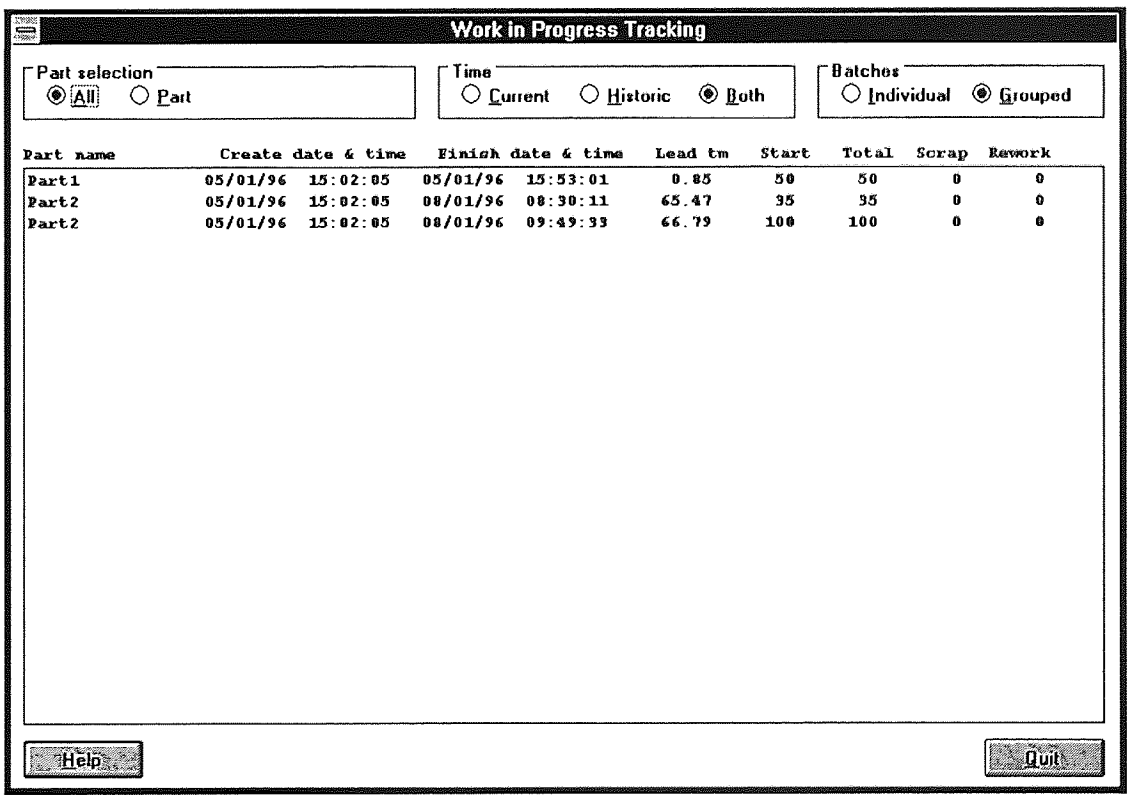


Figure A6.3 Selected Results from the Combined Model (based on 1 week simulation)