Loughborough University

University Library

Author/Filing Title ....DANI, S...........

....

Class Mark ....

Please note that fines are charged on ALL overdue items.

REUSABILITY IN MANUFACTURING, SUPPORTED BY VALUE NET AND PATTERNS APPROACHES

By
Shilpa Dani

A Doctoral Thesis

Submitted in partial fulfilment of the requirements for the award of

Doctor of Philosophy of Loughborough University

2004

© **Shilpa Dani**(2004)

Loughborough University Library

Date	
Class	
Acc No.	043109981

ABSTRACT

The concept of manufacturing and the need or desire to create artefacts or products is very, very old, yet it is still an essential component of all modern economies. Indeed, manufacturing is one of the few ways that wealth is created. The creation or identification of good quality, sustainable product designs is fundamental to the success of any manufacturing enterprise. Increasingly, there is also a requirement for the manufacturing system which will be used to manufacture the product, to be designed (or redesigned) in parallel with the product design. Many different types of manufacturing knowledge and information will contribute to these designs. A key question therefore for manufacturing companies to address is how to make the very best use of their existing, valuable, knowledge resources.

Reuse is not a new strategy, reusing an existing design in a new application is an obvious way of reducing effort and risk - not just in the design activity itself, but also in downstream functions. When changes must be made within the manufacturing system, the application of an existing design or parts of an existing successful design may help to reduce the resources needed in the initial design and analysis stages. Reuse of designs, or parts of designs, can also help to avoid the errors and uncertainty that accompany all human activities and design or development in particular. In addition, reuse helps to maximise the familiarity of production staff with the selected design and helps clients to maintain consistent ways of using and maintaining the result. However, many companies are not achieving their full potential for reuse, some have had disappointing results from their reuse efforts, and many others are still avoiding reuse altogether.

The research reported in this thesis examines ways of reusing existing manufacturing knowledge of many types, particularly in the area of manufacturing systems design. The successes and failures of reported reuse programmes are examined, and lessons learnt from their experiences. This research is therefore focused on identifying solutions that address both technical and non-technical requirements simultaneously, to determine ways to facilitate and increase the reuse of manufacturing knowledge in manufacturing system design.

The solution proposed by this research, is a new design for a Reuse-Driven Manufacturing System Design Process, which adopts a combined approach of a reuse value net and reuse process patterns. To explore and test the proposed solution, a prototype support tool and environment, which includes a specialist application called the Reuse Agent, has been designed, implemented and tested. The prototype tool that has been developed can help the organisation to identify the problems or obstacles that currently exist in their reuse implementation process and the tool can also propose possible solutions to overcome the identified problems. The proposed Reuse-Driven Manufacturing System Design Process and prototype support tool are demonstrated through a case study example and conclusions and recommendations for future work are presented.

***Key words**: Manufacturing system design, Reuse, The value net, Process patterns.*

DEDICATION

To my husband Samir and

my daughter Harshita for

their Love and Support

ACKNOWLEDGEMENT

I would like to thank Wolfson School of Mechanical and Manufacturing, Loughborough University for giving me the opportunity and financial support for pursuing this research.

Many people have directly or indirectly supported this research and I would like to thank all of them.

Most of all I would like to thank two individuals – Dr. Jenny Harding and Samir Dani. I couldn't have done this without the both of you. Dr. Jenny Harding, for your professional guidance, encouragement, understanding and friendship. I have really enjoyed working with you. Samir, thank you for all your help, love and trust over these years. Thank you for the hours of discussion and advise.

I would like to thank Dr. Young, Prof. Burns, Prof. Case, and Dr. Newman for their guidance and support.

I would like to specially thank my family in India, my parents and my in-laws. For their support over these years.

Thank you very much David, Adrian, Irena, Shahbaz, Brenda, Srinivas, you have been good friends and made the lab a second home.

I would also like to thank my friends in Loughborough and Leicester for their support friendship and good food and with whom my family and I have shared so many enjoyable times.

TABLE OF CONTENTS

List of Figures

Chapter 1
INTRODUCTION

1. 1 Introduction

Today, manufacturing enterprises are under continuous competitive pressure in the market place. They must shorten delivery times whilst trying to increase product quality; they must reduce order through-put times whilst providing wider ranges of products; they must offer competitive product prices whilst accommodating the introduction of new technologies and continuously strive for improved quality and after sales services. Manufacturing enterprises must therefore aim to make the best possible use of all their available resources, and it is therefore essential that they reuse and exploit their existing knowledge and expertise as fully and effectively as possible.

Manufacturing systems for producing goods are very complex discrete event dynamic systems (DEDS) making their design and control a challenge for many companies. Manufacturing system design or redesign generally includes some elements of modelling and simulation, so that the current systems and processes can be better understood. Models and simulations also enable substitute or new systems and processes, to be evaluated before expensive modifications are implemented (Harding & Popplewell, 1998) (Harding et al, 1999). Building models of all types is an expensive process as it takes considerable time and effort to fully analyse and understand the systems and processes that form an essential part of the operation of an enterprise. Yet, this knowledge must be gained before appropriate enterprise or simulation models may be built. Models must also be appropriate to meet the objectives of the project and tasks that they have been built to support. It can therefore be difficult to reuse existing models for other projects, particularly if different personnel are working on the projects. Unfortunately, a large (and expensive) amount of time is generally needed to re-examine all the possible existing models, and modify them as required, hence models are often used for one project and then rejected (and wasted) as new models are built for the next project. This is a waste of information and resources, as some elements of the existing models may well be appropriate for future use. When changes must be made within the manufacturing system, the application of an existing design or parts of an existing successful design may help to reduce the resources needed in the initial design and analysis stages. Reuse of designs, or parts of designs, can also help to avoid the errors and uncertainty that accompany all human activities and design or development in particular. In addition, reuse helps to maximise the familiarity of production staff with the selected design and helps clients to maintain consistent ways of using and maintaining the result (Busby, 1999).

"The aim of this research is therefore to determine ways to facilitate and increase the reuse of manufacturing knowledge in manufacturing system design."

One of the main objectives of this research is therefore the design of an efficient reuse driven manufacturing design process. The research reported in this thesis therefore examines ways of reusing existing manufacturing knowledge and this may exist in many forms, for example as information, models or software. As this is a fairly new area of research, the longer established area of software reuse was initially studied to identify similarities and lessons, which might be beneficial to the design of an efficient reuse driven manufacturing system design process.

Existing case study reports of reuse programs have been analysed to build the theory, which forms a basis for the proposed solutions, that include a value net driven manufacturing knowledge reuse process and associated support tool. The proposed process and tool are main contributions of this research, and this thesis shows that they address the problem areas and failure factors that have been identified in earlier reuse programs.

In the past, reuse programs have commonly assumed that technical solutions would overcome barriers to effective reuse. Reuse is often approached as an independent collection of tools and techniques, and the technical and non-technical aspects are commonly examined separately. However, recent retrospectives of reuse programs show that organisational factors can greatly affect the implementation of a reuse program. Various factors and players are involved in the reuse process. A variety of technical, managerial, social, and legal factors must be recognised and addressed in order to achieve an efficient reuse process.

The identification of these factors and players is not sufficient in itself to arrive at a decision about the requirements of an efficient reuse process. To make such a decision, the factors need to be depicted in a fashion, which makes it easier to analyse the reuse process. This research proposes that all the factors affecting reuse are interdependent and therefore should be studied simultaneously and documented in a structured, clear manner. The concept of value nets, based on game theory has been used to capture the different factors involved in the reuse driven system design process.

The proposed reuse value net is a map of the various factors and players identified in the reuse process, and the interactions occurring between the players. This thesis will show that the proposed reuse value net is a valuable tool for understanding the interdependencies and points of leverage that can be used to influence the players and advance or improve the reuse process. The reuse value net therefore enables the current status of reuse within an organisation to be established, and areas of improvement to be identified.

A strength of the proposed knowledge reuse process and associated support tool is that the reuse value net is applied in combination with a process pattern approach. In brief, once the areas of improvement have been identified, the tool provides practical assistance to managers and reuse workers, so that they can expand and improve their application of reuse. This is achieved through the identification and use of process patterns that are appropriate for both their existing business processes, and the level of reuse that currently exists within the organisation. The combined application of value nets and process patterns is described and discussed in the Theory Building section of this thesis and then demonstrated in the Implementation section (See fig. 1.1).

1.2 Structure of thesis

The thesis is arranged into eight chapters including this introduction. An overview of each chapter is listed as follows:

Chapter 2 states the scope of the research. It identifies the research objectives and places these in the context of the research problem being addressed. A description of the research approach is also discussed.

Chapter 3 provides an extensive literature review of areas related to this research work. The main topics covered are manufacturing system design tools and techniques, information and knowledge reuse and intelligent systems.

Chapter 4 explains the theory building stage of the research. Discussion and analysis of the secondary data collected from the literature for this purpose is also provided. The experimental set up is introduced in this chapter, but it is then discussed and explained in more detail in the chapter 5.

Chapter 5 describes the prototype implementation environment used to evaluate the proposed solution.

Chapter 6 is a case study example.

Finally, key issues of the research work are concluded in **Chapter 7.** The achievements of the study are addressed, and limitations of the work are also discussed. Recommendations for possible further research are also presented.

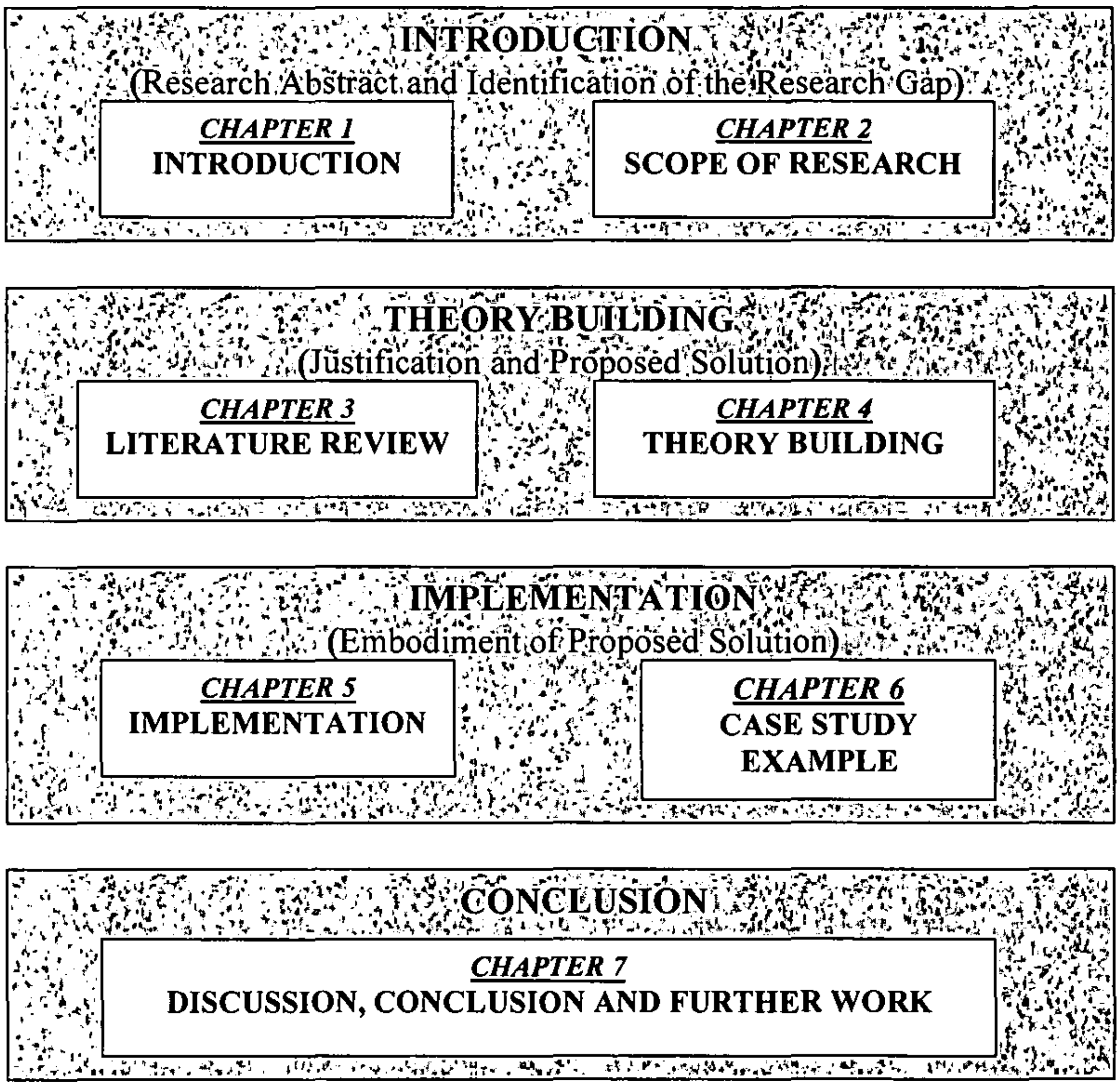


Figure 1.1 - Structure of the thesis

Chapter 2
SCOPE OF RESEARCH

2.1 Introduction

This chapter provides the research background and explains the identification of the research problem. It also lists the aims and objectives of the research and briefly explains the adopted approach.

2.2 Research problem

The research reported in this thesis examines ways of facilitating and increasing the reuse of manufacturing knowledge that may exist as information, models or software and which can be utilised during manufacturing system design.

2.2.1 Background

An enterprise generates wealth by operating a business, which produces products or services for a market (Yu et al, 2000). In today's business context the imperatives of cost efficiency and customer responsiveness have driven firms to aggressively pursue two common business strategies - globalisation (i.e. the worldwide distribution of production and associated facilities) and time-based competition. Both these strategies have dramatically transformed the way in which business activities are organized and carried out. Globalisation is motivated by pressures like cost-effectiveness, access to new markets and economies of scale (Bhatnagar & Viswanathan, 2000). It has led to the emergence of borderless organizations with globally distributed suppliers, production and related facilities.

Manufacturing is one of the few ways that wealth is created. The concept of manufacturing and the need or desire to create artefacts or products is very, very old, yet it is still an essential component of all modern economies. A manufacturing system is a set of machines, transportation elements, computers, storage buffers, and other items that are used together to produce products. People are also part of the system. Manufacturing system design is the choice and arrangement of machines, buffers, transport system, computer and communication system, operators, repair personnel, and so forth. Increasingly manufacturing process and systems need to also be designed and installed in parallel with product design and implementation. Because of the complexity of many manufacturing systems, the amount of information required to make good decisions is immense. A great deal of effort therefore needs to be expended to develop an information infrastructure that will provide the appropriate data as it is required.

The provision of a common source of information, through data models, is one of the fundamental areas where software systems can support design processes (McKay et al., 1996). A manufacturing model is used to describe available manufacturing processes, resources and strategies (Molina, 1995a). By acting as a single source of information on available manufacturing capabilities and status it helps to promote a common understanding of the manufacturing enterprise without placing undue restrictions on the autonomy and heterogeneity of anyone who wishes to use this information. Sometimes the information required to develop a new system partially exists in the design of a previous product in some form. The designing process can thus be accelerated if new designs can use part or all of the information from previous designs. For manufacturing system design potentially reusable knowledge may exist in the form of complete systems or complete process, complete models or parts of models, one or more sub-processes forming part of more complex model, models of resources, models of controls (e.g. kanban), information about requirements, specifications, codes, documentation etc. From now on, any of these potentially reusable pieces of knowledge will be referred to as "reusable components". Identifying reusable components can help the developer to design the new system in a shorter time, hence reducing the product development time.

2.2.2 Research gap

Reuse is not a new strategy, reusing an existing design in a new application is an obvious way of reducing effort and risk - not just in the design activity itself, but also in downstream functions. The potential of reuse has been identified by many researchers and research groups with differing motivations (Sivaloganathan & Shahin, 1999). Many companies trying to implement reuse, are not achieving their full potential for reuse, some have had disappointing results from their reuse efforts, but many others are still avoiding reuse altogether (McCLure, 1997). However, the problem is not the lack of technologies to support reuse. The problem arises when organizations approach reuse as an independent collection of tools and techniques, or when an organization focuses purely on the technical issues of reuse without adequately addressing the non-technical issues (Fafchamps, 1994). Too often in the past, organizations have treated reuse solely as a technical problem and neglected the welter of other critical factors, including organisational factors. More recently, many of these issues have been discussed in the literature and at several workshops and conferences (Frakes and Isoda, 1994).

This research is therefore focused on identifying solutions that address both technical and non-technical requirements simultaneously.

In order to achieve high levels of reuse and so gain the anticipated benefits, persuasion processes, organization, and management changes are required. Funding is needed to pay for the ongoing maintenance of components and people need to be trained. Experience in the realm of software reuse shows that reuse of code alone is not sufficient to produce large impact. Since the cost of the coding phase of a project is typically less than **40** percent of the whole, ways to effectively reuse other software work products (such as designs, tests, or documents) are needed to fully exploit the reuse opportunity (Griss, 1993). This point is also relevant for reusable components in manufacturing systems. Each reuse organization will encounter varying opportunities and issues peculiar to its situation. For a reuse program to be effective, the specific inhibitors likely to affect its takes up and success must be identified and overcome in a timely way.

There are many inhibitors to starting and running an effective reuse program. A study of reuse practices in different industries identified in current literature has made it strikingly clear that the impediments to improving reuse are predominantly non-technical and socioeconomic. These include many cultural, organizational, business, or process factors, which can be overcome only through a combination of management, education, process, policy, and incentive initiatives. The different factors involved in a reuse process have been identified and studied separately by different research groups. However, the identification of the factors is not sufficient in itself to arrive at a decision about the requirements of an efficient reuse process. All the factors involved in the reuse process play different roles, but they are related and are all interdependent and hence should be studied simultaneously to achieve an efficient reuse process. Study of their interdependencies provides knowledge of how a particular factor affects the reuse process and what possible solutions may exist to overcome the problem identified.

2.2.3 Research questions

The research reported in this thesis attempts to answer the following questions:

- How can reuse be implemented effectively in a manufacturing system design process?
- What are the factors involved in implementing and operating a reuse process for manufacturing system design?
- What is the role of each factor, and how are the interdependencies between different factors involved?
- Can the reuse process be made more efficient if the value of one or more factors is modified?
- How can a reuse process be made more efficient in order to achieve the maximum potential benefits from reuse?

2.3 Aims and objectives of the research

The main aim of this research (as stated in section 1.1) is "to determine ways to facilitate and increase the reuse of manufacturing knowledge in manufacturing system design". To satisfy this aim, work has been done to fulfill the following objectives:

1. Explore the current practices of information reuse in organizations.
2. Identify similarities and lessons that can be learnt from the longer established area of software reuse.

3. Identify the different factors involved in the reuse process and study the positive or negative impact of these factors on the reuse process.
4. Design an efficient reuse driven manufacturing design process (exploiting the results of objectives 2 and 3).
5. Design an experimental software environment (to support objective 4).
6. Test and validate the research done using case study example in the experimental environment.

2.4 Research approach

A research methodology has been designed to fulfil the objectives of the research. The methodology has two phases.

1. A Theory Building Stage
2. An Experimentation Stage

2.4.1 <u>Theory Building Stage</u>- Initially, current practices and requirements of reuse projects are examined. The theory building stage then leads to the identification of a conceptual framework for an efficient reuse process. The concepts presented in the following references have strongly influenced this research and have therefore been used to design the conceptual framework.

i. Case study material from reuse projects – (see Section 4.2)
ii. Factors Affecting Reuse and existing reuse techniques – identified By McClure Carma in her book "Software reuse techniques" (1997)
iii. The Value Net - identified by Nalebuff and Brandenburg in their book Co-opetition (1996)
iv. Process Patterns- identified by Scott Ambler (1998)

The following activities were undertaken during the theory building stage:

a. ***Examination of reuse processes, and factors associated with reuse.***
b. ***Examination of existing tools and techniques to support reuse.***
c. ***Identification of Factors and Players and their influence on the reuse process.***

d. ***Identification of Points of Leverage to improve Reuse.***

e. ***Examination of Relationships between promoters and inhibitors of reuse.***

2.4.2 Experimentation Stage- the second phase of the research methodology was the experimental prototyping. During this stage, the research theories were tested through a case study example. The design, implementation and testing of a prototype reuse support tool was considered to be the most appropriate method of demonstrating this research. One of the reasons for using this type of experiments is that it is valuable for testing theories that relate not to immediate practical concerns, but rather to the accrual of knowledge that will be of long-term benefit to the field. (Benbasat, 1989)

According to Kerlinger (1986), laboratory experiments serve the following purposes:

- they allow for the testing of predictions derived from theory or inferences drawn from other studies by providing a means for studying relationships under controlled, unconfounded conditions; and
- they can be used to build theoretical systems.

Chapter 3
LITERATURE REVIEW

3.1 Introduction

This chapter has two purposes, firstly to review relevant and up to date literature related to manufacturing systems, the information and knowledge used in modern day manufacturing systems and the associated information technology (IT) systems for manufacturing. Its second purpose is to review the use of knowledge based systems and the application of artificial intelligence (AI) tools and techniques in manufacturing systems, to provide a background for the proposed reuse support tool. Since the aim of this research is to facilitate and potentially accelerate manufacturing system design, through the reuse of manufacturing information, knowledge and software, it is important to consider existing approaches to manufacturing system design as these will inevitably influence the types of reusable components that may exist, or which are required. Hence, section 3.2 examines several aspects of manufacturing systems, and provides an overview of manufacturing technology development, manufacturing system design, common support tools and techniques used in their design. AI applications inherently capture and exploit knowledge from the application domain or of the expertise demonstrated by practitioners within the particular domain. Such knowledge may be potentially reusable. Therefore, intelligent systems, knowledge based systems and agent based systems have all been considered and are also discussed in this section, along with examples of the application of different AI techniques to different areas of manufacturing.

If information and knowledge are to be reused, they must be available in forms that can be shared between users. Section 3.3 therefore considers different types of information models, which have been used in manufacturing contexts. The concurrent engineering approach is also considered, as this has a substantial requirement for information sharing. Information and knowledge modelling are important to this research, both in the context of system design and the structuring of information and knowledge for reuse. Examples of different types of modelling and methodologies for system design and modelling are therefore also reviewed in this section.

Section 3.4 examines enterprise integration. This is an important subject area as an aim of enterprise integration is to ensure that each unit has access to relevant and necessary information. Hence in integrated environments, information needs to be accessible, able to be shared, and therefore reusable to at least some extent.

Reuse is a relatively new topic in manufacturing. However, literature has been identified in the areas of engineering design information reuse, and more general design reuse, and these are reviewed in sections 3.5. Finally, the need for systematic reuse in manufacturing and potential reusable components in manufacturing system design is discussed in section 3.6.

3.2 Manufacturing System Engineering

The terms systems, systems thinking and systems approach are being used with increasing frequency, whilst the concepts they describe are becoming central to the thinking and approach adopted by many different disciplines (Mason-Jones et. al., 1998). Systems thinking involve being systemic or thinking of entities, situations, problems as a complex of interacting parts, which can be divided up into specific systems and sub-systems (Mason-Jones et. al., 1998). The identification of systems is followed by an examination of the relationships between systems/subsystems, flows of influence, materials, energy etc., that occur both within and between systems (Open Systems Group, 1987).

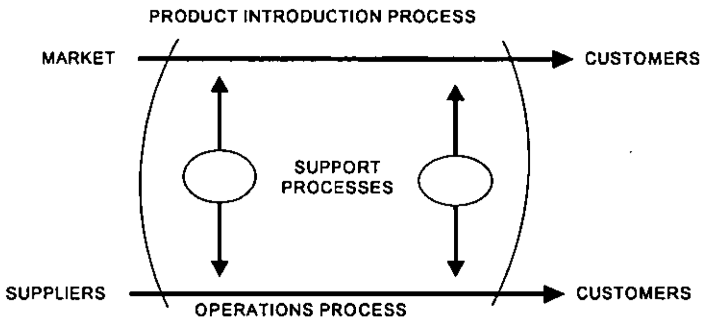
Systems concepts can be readily applied to business organisations and manufacturing companies in particular. A manufacturing company can be considered as a system that converts inputs flows (such as, materials, money, manpower, energy) into output flows (such as, finished goods, profits, waste) (Parnaby, 1979). Such a simple model can be further refined. The focal point of the system is a value-adding process that converts materials into products.

Lucas plc have been strong advocates of a manufacturing systems perspective and systems approach to the redesign of business organisations (Mason-Jones et. al., 1998).

Central to the Lucas philosophy is

- a manufacturing systems model based on organisational processes;
- a manufacturing systems redesign methodology based on the application of a systems engineering approach.

From the Lucas perspective the ideal manufacturing system is designed around two core processes, manufacturing operations and product introduction processes with



support processes, as shown in figure 3.1.

Figure. 3.1 Lucas Model of organisational design (Mason-Johns, et. al., 1998)

In order to redesign manufacturing systems for world-class performance, it is imperative that a systems approach to redesign is adopted, as pioneered by Toyota and emulated by world class companies ever since (Mason-Jones et. al., 1998). All manufacturing systems should be an integrated whole (Burbidge, 1996). Every sub-system of the whole is required to process certain inputs and produce certain outputs. To obtain desired outputs the correct inputs must be available. When all sub-systems are placed together, the input-output interactions must be compatible (Parnaby, 1979). Even if the new design is installed as a sequence of sub-systems over a number of years each piece will match the final system at its interfaces and the final design will interface with the multi-variable needs of the business environment (Parnaby, 1991).

Business unit redesign is undertaken using a systems engineering approach to implement sets of best practice methodologies for the achievement and maintenance of competitiveness and customer satisfaction (Parnaby, 1994). In Lucas terminology systems engineering is described as the:

> *science of designing complex systems, by the efficient use of resources in the form of men, materials, machines and money, so that individual sub-systems making up the overall system can be designed, fitted together, checked and operated so as to achieve the overall objective of the system in the most efficient way (Anon, 1989).*

More information about the Lucas approach to redesign can be found in Anon, 1989; Parnaby, 1991; Parnaby, 1994.

The above systems perspective of a manufacturing system can be traced back to over 40 years when Burbidge (1984, 1996) postulated his Law of Gestalt for manufacturing systems: "The whole is not the sum of its parts" and that "a set of sub-optimum solutions can never produce a true optimum solution".

3.2.1 An Overview of Manufacturing Developments

Recent advances in manufacturing and information technologies present promising new strategic alternatives for designing a new manufacturing information system (Tan and Uijttenbrock, 1997; Coates, 2000). The growing complexity of industrial manufacturing and the need for higher efficiency, greater flexibility, better product quality, and lower costs have changed the face of manufacturing practice. The quest for lower operating costs and improved manufacturing efficiency during the 1990s forced a large number of manufacturing firms to embark on advanced manufacturing technologies (AMTs) projects of various types (Udo, & Ehie, 1996). The dramatic developments in AMT at various organizational levels can be attributed to numerous benefits that improve the competitive position of the adopting companies. AMT impact not just manufacturing, but on all the business operations, giving new challenges to a firm's ability to manage both manufacturing and information technologies.

Modern manufacturing technology is interdisciplinary in nature and allows the application of different knowledge from other scientific fields such as manufacturing, computer science, management, marketing and control systems. By successfully extending the integrated system concepts beyond the ordinary manufacturing functions, manufacturing firms are able to achieve an optimal balance of product standardization and manufacturing flexibility. Information technology is a key ingredient in this emerging recipe for achieving competitive advantage through manufacturing (Coates, 2000; King and Teo, 1997; Lederer and Sethi, 1996; Sambamurthy *et al.*, 1993). The computer-era has changed the way that products are manufactured, enabling the implementation of progressively more advanced manufacturing techniques. Rao et al defined the first four stages of industrial automation in 1993 and these are listed in Table 3.1. Since Rao's categorisation, more recent developments have included ever increasing use of intelligent systems and AI in manufacturing.

Stage	Feature	Automation	Design	Manufacture
1	Labour-intensive	None	Individual	Manual
2	Equipment-intensive	Instruments	Group	NC, CNC
3	Information-intensive	Information	CAD	FMS
4	Knowledge-intensive	Decision	ICAD	CIMS

Table 3.1 Development of industrial automation (Rao et. al., 1993)

In the first stage of development of manufacturing industry described by Rao, the quality of the product was dependent on the skills and experience of the human operator, but in the second stage, the introduction of automatic equipment played a dominant role. The result was improved quality with more productivity. The common example used to demonstrate this is the introduction of numerical control (NC) machines. In the third stage, automation was realized at the level of data processing for groups of automatic machines (Rao. et. al., 1993). Manufacturing industry was moving towards the intelligent systems. Computers assist human experts for numerical analysis, synthesis, simulation and graphics and also provide information for them to make decisions. Fault Diagnosis and equipment maintenance depends not only on the information from sensors but also on human experts' experience. CAD/CAM technology is an example of this stage. With the rapid growth of complexity of manufacturing processes and demand for higher efficiency, better product quality, and lower costs, industrial practice approached a more advanced level of automation (Rao, et. al.,1993). AI started to appear in Rao's fourth stage, and computers helped human experts not only for decision-making as well as for data processing.

Throughout the 1990s, and into the new millennium, intelligent systems have become increasingly intelligent, and an increasing emphasis has been laid on AI in manufacturing. Since its emergence in the 1950s, AI has provided several techniques with applications in manufacturing. The application of AI in manufacturing has been the subject of extensive research in the last two decades. This surge in the application of AI in manufacturing is mainly due to the availability of powerful computers. In the early years, knowledge-based systems (KBS) attracted great attention. Recently neural networks (NN), case-based reasoning (CBR), genetic algorithms (GAs) and fuzzy logic have attracted more attention and have been successfully employed in manufacturing (Meziane, et. al., 2000). Agent based systems have also increasingly been used in particular areas of manufacturing. This progressive adoption of AI in manufacturing can be seen by the examples of the application of KBS through to agent based systems which will be described in more detail in sections 3.2.1.2 and 3.2.1.3.

The World Wide Web represents one of the most important challenges in the emerging information society. Many organisations and companies have turned to the Web to sell and promote their product. The Web is seen as the future window shopping for businesses. This new approach to business is commonly known as e-commerce. The www is also promising for distributed manufacturing.

Virtual Manufacturing (VM) is another new approach to manufacturing. It requires a robust information infrastructure that comprises rich information models for products, processes and production systems (Meziane, et. al., 2000). The decreasing costs of hardware have made virtual environments increasingly popular and they are used in many fields. Virtual Reality (VR) can be applied for most components of the manufacturing process. In design VR can support detailed and accurate design activities and provide sophisticated means of manipulating shape and form represented by the virtual models. Nakayasu *et al.* (1999) used VR for design and production of metal sheet. VM is expected to support the assessment of the manufacturability of a candidate design and to provide accurate estimates for processing times, cycle times and costs as well as product quality. Lin et al argue that VM will be able to model both the processes employed for the product's manufacture and the production process (Lin *et al.*, 1995a). VM may play a significant role in distributed manufacturing and it is expected to support distributed design. VM can provide details and information about process, production and shop floor control to be shared over networks.

The performance of computer hardware also keeps improving and this will result in more powerful applications and automations in manufacturing. In addition to providing improved knowledge and decision making support to human operators, the increased performance and lower-cost hardware will enable computers to carry out progressively more tasks that were previously carried out by humans. Hence, robots will be needed and Kopacek argues that the research in robotics will be dominated by two directions (Kopacek, 1999). Additional features such as combined force and position control, external sensors based on micro-systems, flexible and lightweight structures need to be added to robots used in classical applications. Another type of robot that need to be developed for the next generation are service robots. Service robots are characterised by the following facilities: mobility, portability, operating case, sensing, learning, judging function and adaptability (Kopacek, 1999). In an invited paper, Burdea (1999), reviewed how robotics and VR can be integrated. VR-enhanced CAD design, robot programming and plant layout simulation were considered.

The total manufacturing information system (TMIS) is also a powerful alternative which blends recent developments in manufacturing and information technology to achieve competitive advantage (Lee, 2003). TMIS implies movement toward total integration of manufacturing technologies and business strategies into an information system. It includes all the business functions today's manufacturers should have, from market analysis to quality control and management with business decision support capabilities. TMIS enables manufacturing firms to respond quickly to market changes, achieve flexibility of products and process, and manage the complexity of today's manufacturing environment. It supports and shapes an organization's competitive strategies, and provides competitive advantages to a firm. TMIS becomes more powerful as they cope with changes in technology, resources, demands and responsibilities.

The following section presents the application of different intelligent AI techniques such as Neural Networks, GA, Fuzzy Logic, etc in different areas of manufacturing. This is relevant to the current research, to demonstrate the more recent developments in industrial automation and also to show the types of manufacturing information and knowledge required and used by such systems.

3.2.1.1 Intelligent Systems in Manufacturing

The earliest substantial work in the field of Artificial Intelligence (AI) was done by the British logician and computer pioneer Alan Mathison Turing (Copeland, 2000). In 1935, at Cambridge University, Turing conceived an abstract computing machine. Turing's computing machine of 1935 is now known simply as the universal Turing machine, and all modern computers are in essence universal Turing machines.

Intelligence refers to the ability to capture and apply application domain-specific knowledge and processing to solve problems (Bigus, Bigus, 2001). Intelligence is the property, which enables the system to operate effectively when available information is inadequate (Rzeveski, 1997). The intelligence in the intelligent systems can range from hard-coded procedural or object-oriented logic to sophisticated reasoning and learning capabilities (Bigus, Bigus, 2001). Therefore, intelligent systems should have the capabilities to acquire and interpret information about that part of their environment where changes are likely to occur. From the information acquired, the intelligent system should be able to cope with the uncertainty that arises due to changes in its environment.

Typical characteristics exhibited by an intelligent system Rzevski, (1997) are: adaptability, self-maintenance, communication, autonomy, learning, self-improvement, anticipation, goal-seeking, creativity, reproduction. The intelligent system is capable of exchanging information with other systems and is capable of changing its behaviour to accommodate unpredictable changes in its environment.

The development process of intelligent systems is divided into five stages- identification, conceptualisation, formulation, implementation and evaluation.

1. At the first stage, i.e. **identification**, the issues related to the problem, participants, resources, and goals are considered.
2. At the **conceptualisation** stage, the problem is decomposed into sub-problems, knowledge acquisitions are performed, and input/output relationships are analysed.
3. The **formulation** process involves mapping the key concepts, sub-problems and information flow characteristics that were isolated in the conceptualisation stage into the formal representation. Knowledge is analysed, and the user-interface is designed.
4. The fourth stage, **implementation**, transfers the formalized knowledge into the representation framework associated with the development tools chosen for this problem.
5. The final stage, **evaluation**, involves testing and modifying the prototype intelligent system and the form of its representation that has been used to implement the system.

Components of an intelligent manufacturing system

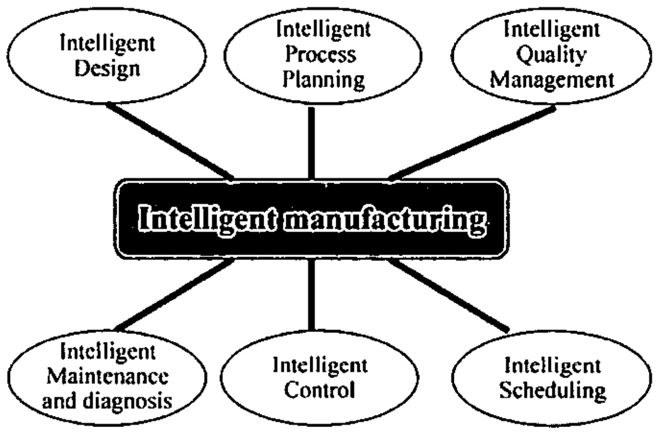

Intelligent Design
Intelligent Process Planning
Intelligent Quality Management
Intelligent manufacturing
Intelligent Maintenance and diagnosis
Intelligent Control
Intelligent Scheduling

Fig. 3.2 Intelligent Manufacturing (Meziane et. al, 2000)

Rao *et al.* (1993) decomposed intelligent manufacturing systems into the following components: intelligent design, intelligent operation, intelligent control, intelligent planning and intelligent maintenance. Meziane et. al.(2000) modify this decomposition slightly to reflect the current trends in the literature on intelligent manufacturing systems as shown in Figure 3.2. They give a brief description of each phase as follows, and look at how AI techniques are used within each component.

Intelligent design

The importance of product design is undeniable. A firm's products or services are typically the primary source and focus of contact with its customers, and the development of new designs plays a key role in establishing and maintaining a competitive position for most firms. There are many problems in design manufacturing systems. A review of the problems encountered in manufacturing systems can be found in Kouvelis (1992).

Neural Networks (NN) have been used for the flat rolling process (Gunasekera *et al.*, 1998), Rao and Gu, (1995) used a multi-layered NN to configure and alternate cell designs by considering multiple constraints and objectives, Kusiak and Lee (1996) constructed a three layer NN for designing a cellular manufacturing system that integrates several manufacturing functions. A fuzzy basis material removal optimisation approach is suggested by Ip (1998) to compensate the variation of cutting speed due to the change of gradient on the sculptured surface in machining process. Babuska *et al.* (1999) used fuzzy modelling used for the penicillin-G conversion process. Gao *et al.* (1998) used Cased-Based-Reasoning (CBR) for mechanical plan systems design. Design plans are stored as the actual cases in the CBR system. CBR is used for the design of bar linkages (Bose *et al.*, 1997) and fixture design (Sun and Chen, 1995). Zeid *et al.* (1997) have proposed a CBR approach to solve design for disassembly (DFD) problem. Chen *et al.* (1998) have developed an integrated expert system that consists of a knowledge base, a database, pattern-recognition, artificial NN and GA modules for complicated chemical reaction systems used to prepare industrial materials. The system has been used in many applications including the production of alloy steel, synthetic rubber, ceramic materials production and materials design of composite materials, high temperature superconductors, and ceramic semiconductors. A material design system has been developed, by Shivathaya and Fang (1999), utilising mathematical modelling and knowledge-based approaches.

Intelligent process planning

Intelligent process planning is a dynamic and complex activity. Process planning provides a detailed description of manufacturing capabilities and requirements for transforming a stock of raw material into a completed product (Requicha and Vandenbrande, 1988). Intelligent process planning includes CAPP and facility and location layout. Process planning is the interface between computer-aided design (CAD) and computer-aided manufacturing (CAM). CAPP is vital to achieving the ultimate goal of complete integrated factories of the future. A CAPP system contains a large amount of knowledge that includes rules about arranging machine operations and facts about the machine shop. Inventory management is also considered under this section because successful inventory management is essential for successful manufacturing and requires sophisticated methods to cope with the continuously changing environment. Literature is rich with papers about theoretical independent demand inventory modelling but practice lags behind these developments. AI can play an important role in aiding practitioners to implement such models and also to overcome the problems associated with managing large-scale inventories.

The application of perceptron-type NNs to tool-state classification during a metal-turning operation is reported by Dimla (1999). GAs have been successfully used in layout design (Suresh *et al.* 1995; Gupta *et al.* 1996; Rao *et al.* 1999), and have been shown to outperform human and KBS designs (Hamamoto *et al.,* 1999). Parallel GAs were also used to solve the layout problem with geometric constraints (Tam and Chan, 1998). Hamamoto *et al.* (1999) integrates GAs with an embedded simulation model to tackle the facility layout problem for pharmaceutical factories. Bhaskara *et al.* (1999) demonstrates the application of GAs as a global search technique for a quick identification of optimal or near optimal operation sequences in a dynamic planning environment. Fuzzy set theory effectively models facility layout and location by incorporating subjectivity in the parameters used by the models (Guiffrida and Nagi 1998). Fuzzy set theory has also been applied in inventory production and process plan selection. Inventory management requires demand forecasts as well as parameters for inventory-related costs such as carrying, replenishment, shortages and backorders (Guiffrida and Nagi, 1998). Precise estimations of these parameters are difficult. Ben-Arieh and Chopra (1997) describe a process planning system that utilises CBR. The system, CBPlan, uses a feature-based part representation as a key to the case library. Champati *et al.* (1996) also used a CBR approach for the automated sequencing in intelligent process planning. Malakooti *et al.* (1995) developed a monitoring and supervising system for machining operations using in-process regression for monitoring and adaptive feed forward artificial NNs for supervising. Ming *et al.* (1999) has combined expert systems and NNs to develop a CAPP system. Other attempts have been made to use AI in managing dependent demand inventories. Notably several applications have been published in the area of JIT including the work of Fielder *et al.* (1993), Rixen *et al.* (1995) and Ettl and Schwehm (1995). A wider discussion can be found in the review of Proudlove *et al.* (1998).

Intelligent quality management

Quality management has evolved from a focus on inspection through quality control techniques such as statistical process control and through quality assurance to current total quality management (Proudlove *et al.*, 1998). More organisations are involving customers in the early stages of design to assure quality and a market for their products. Zhang and Huang (1995) stated that there are two approaches to quality assurance: reactive quality assurance and proactive quality assurance. Reactive tools include sampling plans, lot acceptance determination, scrap or rework analysis etc. Proactive strategy requires an emphasis on physical cause-effect knowledge, risk analysis, experience and judgement.

Chinnan and Kolarik (1997) proposed the use of NNs for optimising the controllable variables of a process to achieve real-time quality control. Gill and Bector (1997) used a fuzzy linguistic approach to quantify part feature information for the part family formation problem. In early research, fuzzy logic was mainly used in acceptance sampling and statistical process control (Chakraborty, 1992; Chakraborty, 1994; Kanagawa *et al.* 1993; Wang and Chen, 1995). In more recent research, fuzzy logic is used in quality topics such as quality improvement and quality function deployment (QFD) (Gutierez and Carmona, 1995; Khoo and Ho, 1996; Yongting, 1996; Wang, 1999; Chan *et al.*, 1999). Khoo and Ho (1996) presented a framework for a fuzzy QFD system in which the customer requirements can be expressed as both linguistic and crisp variables. Malek *et al.* (1998) developed an operator support system to help and guide the operator in decision making during the control of the plastic injection moulding process.

Intelligent maintenance and diagnosis

The goal of fault diagnosis is to detect the faults and their causes early enough, so that failure of the overall system can be avoided. From the fault-detection point, faults are divided into three categories, actuator faults, component faults and sensor faults. The basic task of fault diagnosis is to detect the faults that occur, and to provide information about their size and source (Frank and Koppen-Seliger, 1997). Three steps need to be taken in fault diagnosis: signal generation, fault classification and evaluation, and fault analysis. See Frank and Koppen-Seliger (1997) for a survey on the use of AI techniques in fault diagnosis.

NNs have been used for matching stereoscopic pictures and correcting three-dimensional measurement error (Tien and Chang, 1999) extending earlier work by Su *et al.* (1995) and identifying product defects (e.g. Kim and Kumara, 1997; Wang and Huang, 1997). A feed forward NN has been used for manufacturing diagnosis by Ransing and Lewis (1997). Xia and Rao (1999) developed a dynamic CBR system that can represent system dynamics and fault-propagation. Jeon (2000) developed a hybrid intelligent maintenance optimisation system (HIMOS) for decision support which aims at overcoming the problems of IMOS. Labib *et al.* (1997) used fuzzy logic combined with a rule base to develop an "Intelligent maintenance model" which is applied to a manufacturing company to identify the most critical machines and determine appropriate maintenance action.

Intelligent scheduling

Scheduling is a resource allocation problem subject to allocation and sequencing constraints. It is an optimisation problem. The objective in optimisation is to allocate a limited amount of resources to a set of tasks such that cost functions are optimised.

Candido *et al.* (1998) used a GA to solve job scheduling problems with many constraints such as jobs with several sub-assembly levels, alternative processing plans for parts and alternative resources for operations, requirements of multiple resources to process an operation, resource calendar, batch overlap and sequence dependent set-ups. Webster *et al.* (1998) used a GA for scheduling jobs about an unrestricted common due date on a single machine. Lam *et al.* (1999) used GAs for a scheduling problem encountered in a semi-conductor manufacturing company where the time for designing products needs to be minimised. Min and Cheng (1999) have used GAs for the identical parallel machine scheduling problem for minimising the make span. Shipley *et al.* (1996) incorporates fuzzy logic, belief functions, extension principles and fuzzy probability distributions to develop a fuzzy PERT algorithm. Yu *et al.* (1999) proposed an approach to FMS scheduling with multi-criteria based on fuzzy inference. Coello and DosSantos (1999) used CBR and heuristic search for a real-time scheduling system. Szelke and Markus (1997) combined machine learning techniques and CBR to solve the shop floor scheduling problems. Kim *et al.* (1998) used an integrated approach of inductive learning and competitive NNs for developing multi-objective FMS schedulers. Lee *et al.* (1998) combined GAs and machine learning to develop a job shop scheduling system.

Intelligent control

The basic objective of control is to provide the appropriate input signal to a given physical process to yield the desired response. It is a complex process that continues to require human intelligence to ensure proper operation.

Hao *et al.* (1995) used an NN approach in real-time control of FMSs. Sung and Choung (1999) used a multilayer perceptron NN for a batch process in a wafer fabrication. Ong and Khoo (1999) used a GA for optimising the sequence of component placements onto a printed circuit board and the arrangement of component types onto feeders simultaneously. Caprihan *et al.* (1997) used a fuzzy system for the control of flexible machines operating under information delays. Suresh *et al.* (1999) proposed a pattern recognition approach based on a fuzzy ART NN for rapid scanning of families of parts having a similar sequence of operations. Filipic *et al.* (1999) combined machine learning and evolutionary optimisation in learning to control a physical device. Vishnupad (1996) used NNs and fuzzy logic for the control of manufacturing systems. Ortega and Giron-Sierra (1998) propose the use of a fuzzy logic and GAs to develop a control system for provision of spacecraft servicing to a space station which comprises the tasks of assembly, re-supply, repair and maintenance of manufactured space parts in-orbit.

3.2.1.2 Knowledge Based Systems (KBs)

The roots of knowledge-based programming lie in the field of Artificial Intelligence (AI), which may be said to be the science that tries to replicate intelligent human behavior on computers (Dym, et al., 1991). The first attempt to be widely used to equip manufacturing systems with some degree of intelligence was the use of KBS (Schreiber *et al.*, 1993). They seek to incorporate human knowledge about an application area, usually elicited from experts in the particular domain, so that the system can automatically replicate aspects of best practice. The human knowledge is represented using the IF-THEN production systems or more structured formats such as frames and semantic nets. A good system can match the performance of a human specialist. Rao et al (1993) identified expert system or KBS technology as one of the most active branches in AI research. More recently, Copeland (2000) described a KBS as a computer program dedicated to solving problems and giving advice within a specialised area of knowledge. An important feature of KBSs is that they are able to work cooperatively with their human users, enabling a degree of human-computer symbiosis (Copeland, 2000).

Several different definitions of knowledge can be found in literature (Baker, *et al.* 1997, Bender and Fish, 2000, Collin's paperback dictionary, 1995). For example, Sanchez et al. (1996) have defined knowledge as an ability to sustain co-coordinated deployment of assets and capabilities in a way that promises to help the firm to achieve its goals. Nonaka and Takeuchi, (1995) defined it as the meaningful structured accumulation of information.

A knowledge-based (expert) system (KBES) is a computer program that performs a task normally done by an expert or consultant and which, in so doing, uses captured, heuristic knowledge (Dym, *et al.*, 1991). It is a computer program that acquires the knowledge of human experts and applies it to make inferences for users with less training or experience in solving various problems (Rao, *et al.*, 1993).

There are three primary knowledge processes, which are common across knowledge-based organisations: (1)Adding value to information, (2)Generating, capturing and sharing knowledge, (3)Applying knowledge (Knowledge management approach in NGM).

The basic components of an expert system are a "knowledge base" or KB and an "inference engine"(Copeland, 2000). The information in the KB is obtained by interviewing people who are expert in the area in question. The interviewer, or "knowledge engineer", organises the information elicited from the experts into a collection of rules, typically of "if-then" structure. Rules of this type are called "production rules". The inference engine enables the expert system to draw deductions from the rules in the KB. For example, if the KB contains production rules "if x then y" and "if y then z", the inference engine is able to deduce "if x then z". The expert system might then query its user "is x true in the situation that we are considering?" and if the answer is affirmative, the system will proceed to infer z.

The basic structure of a knowledge-based system as shown in Figure 3.5.

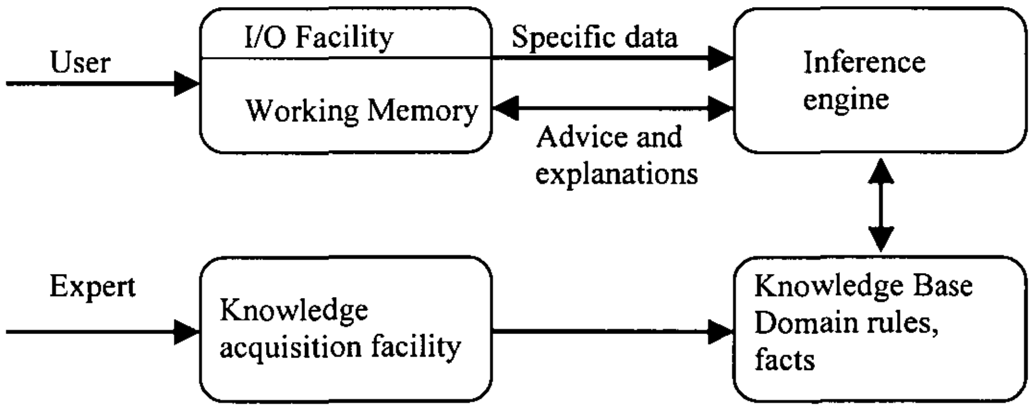


Figure 3.3 The components of a basic knowledge-based (expert) system (after [Dym 1985; Feigenbaum 1983]).

The components include **Input/output** facilities that allow the user to communicate with the system and to create and use a database for the specific case at hand; a **working memory** that contains the specific problem data and intermediate to final results produced by the system; an **inference engine** that incorporates reasoning methods, which in turn act upon the input data and the knowledge in the **knowledge base** to solve the stated problem and produce an explanation for the solution. A knowledge base that contains the basic knowledge of the domain, including facts, beliefs, and heuristics unique to the expert. In addition, the system may include a knowledge acquisition facility that allows the KBES to acquire further knowledge about the problem domain from experts or automatically, from libraries or databases.

Applications of Knowledge Based Systems in Manufacturing

KB Design - KBS were extensively used in early intelligent manufacturing systems and many of them were used in the design phase. Chon *et al.* (1993) reported the use of a KBS for centrifugal fan blade design. A number of KBS were used in the electronic engineering field. A survey of the applications developed before 1993 can be found in Rowland and Jain (1993). In the same period, KBS were also used for the design of boilers circulating fluidised beds for (Mitra *et al.*, 1993) and for designing computer network topologies (Pierre, 1993). Basu *et al.* (1995) used an expert system for the design of manufacturing cells. The trend of using KBS for manufacturing design has continued throughout the 1990s. Recently, KBS were used for concurrent engineering in metallurgy component design, materials selection, powder packing and compaction (Smith and Midha, 1999).

KB Process Planning- By 1993, a large number of expert system had been developed to assist process planning. An early survey on the use of expert systems in process planning was carried out by Alting and Zhang (1989) while a more recent survey was carried out by Kiritsis (1995). The maturity of the use of expert systems in process planning has pushed some researchers to look at the development of tools to build expert system for process planning (Eskiciogolu, 1992). Wong and Siu (1995) used an expert system for automatic process selection and sequencing. Pande and Desai (1995) used an expert system (EXTURN) for the process planning of rotationally symmetric components manufactured on single spindle automats.

KB Quality Management – A good review of expert systems used in quality control up to 1992 can be found in Kuo and Mital (1993). Deslandres and Pierreval (1995) developed SYSMIQ, a knowledge-based advisory system for quality control to assist decision-makers in selecting the best quality tools and techniques and correctly apply them on the shop floor.

KB Maintenance and diagnosis – A number of KBS were used for maintenance and fault diagnosis. A survey on the use of KBS for failure diagnosis before 1993 can be found in Rowland and Jain (1993). Arslan *et al.* (1993) reported the use of an expert system for failure diagnosis for printed circuit boards. The KBS exploits functional test data, which is output from automatic test equipment which is used to test every board subsequent to manufacture. Fujikawa and Ishii (1995) use a KBS to identify the causes of various manufacturing defects in hot forging and suggest remedies.

For maintenance, several systems have been developed to tackle specific industrial applications. For example Clark *et al.* (1992) developed a KBS to optimise the building management maintenance and Batanov *et al.* (1993) developed EXPERT-MM: a KBS for maintenance management for a large manufacturing company. A more general approach based on knowledge-based reasoning was adopted by Kobaccy *et al.* (1995) and Zhang and Jardine (1997). Kobbacy *et al.* (1995) developed IMOS, a prototype intelligent maintenance optimisation system aimed at developing and enhancement of preventive maintenance routines for large and complex industrial systems. IMOS has a rule base for selecting an appropriate model for application based on identification of maintenance data pattern. Zhang and Jardine (1997) proposed a similar smart system for data-analysis models and optimising replacement age.

3.2.1.3 Agent based systems in manufacturing

Techniques from AI have been used in Intelligent Manufacturing for more than twenty years. In the past ten years, researchers have been applying agent technology to manufacturing enterprise integration and supply chain management, manufacturing planning, scheduling and execution control, materials handling and inventory management, and developing new types of manufacturing systems such as holonic manufacturing systems.

In the mid eighties, Minsky introduced the concept of the Society of Mind, as a

> *"scheme in which each mind is made of smaller processes. These we'll call agents. Each mental agent by itself can only do some simple things that need*

no mind or thought at all. Yet, when we join these agents in societies-in certain ways-this leads to true intelligence" (Minsky, 1985).

An agent is a software object that is capable of communicating with other agents (Rzevski, 1997). "An agent is a computer system situated in some environment, and that is capable of autonomous action in this environment in order to meet its design objectives" (Jennings and Wooldridge, 1998). According to Bigus and Bigus,(2002) agents can be categorised by placing them in three-dimensional space, where an axes are *agency*, the amount of autonomy an agent has; *intelligence*, the knowledge, reasoning, and learning capabilities of the agent; and *mobility,* the ability to move between systems in a network. A review of agent theories, architectures and languages can be found in (Wooldridge and Jennings, 1995).

In distributed intelligent manufacturing systems, agents can be used to

- encapsulate existing software systems so as to resolve legacy problems and integrate manufacturing enterprises' activities such as design, planning, scheduling, simulation, execution, and product distribution, with those of their suppliers, customers and partners into an open, distributed intelligent environment via networks (Fox et al, 1993)(Barbuceanu and Fox, 1997)(Peng et al, 1998) (Shen et al, 1998a);
- represent manufacturing resources such as workers, cells, machines, tools, fixtures, AGVs, as well as products, parts and operations (Butler and Ohtsubo, 1992)(Parunak et al, 1998)(Shen and Norrie, 1998) to facilitate manufacturing resource planning, scheduling and execution control;
- model special services in manufacturing systems, such as:
    - Agent Name Server in CIIMPLEX (Peng et al, 1998) and Enterprise Mediator in MetaMorph (Maturana and Norrie, 1996)(Shen et al, 1998a) for providing registration and administration services;
    - Facilitator agents (also called Facilitators) in PACT (Cutkosky et al, 1993) and CIIMPLEX (Peng et al, 1998) and Mediator Agents (also called Mediators) in MetaMorph for facilitating communication, cooperation and coordination among other agents;

  - Database Agents (Lin and Solberg, 1992) and
  - Information Agents (Fox et al, 1993)(McEleney et al, 1998) for providing information management;
- incorporate a whole scheduler or planner into manufacturing planning and scheduling systems (Fox et al, 1993)(McEleney et al, 1998).
- perform dynamic scheduling in a flexible manufacturing system (Ouelhadj et al, 1999 and 2000).

In holonic manufacturing systems, agents are used to model holons which are software and hardware entities (Deen, 1994)(Christensen et al, 1994)(Hasegawa et al, 1994)(Biswas et al, 1995). Schedule manufacturing orders based on an holonic manufacturing system (Sousa et al, 2003). A good discussion on agent technology for holonic manufacturing systems can be found in (Bussmann, 1998).

Key issues related to agent-based cooperative systems, such as representation, ontology management, agent structure, system architecture, communications, system dynamics, overall system control, conflict resolution, legacy problems and external interfaces, have been discussed in (Shen and Barthès 1996b). Most of these issues are also applicable in agent-based manufacturing systems. (Drakos, 1996).

3.2.2 Support Tools for Manufacturing Systems Analysis and Design

3.2.2.1 Manufacturing Systems Analysis and Design (MSAD)

Manufacturing system design is a complex process involving the integration of multiple systems designed by several designers each optimising sub-systems. The life-cycle of manufacturing system design is also problematic because the size of the manufacturing system can make it incomprehensible and unmanageable by a single person. The system design should therefore be interdisciplinary in approach, asynchronous in operation, and constantly evolving (Koonce et al.,1996). This scope also leads to long development times with requirements that are not well understood in the beginning and that change during design/build cycle.

A design methodology of manufacturing systems can be defined as a set of procedures that analyses and segregates a complex manufacturing system design task into simpler manageable sub-design tasks while still maintaining their links and interdependencies(Rao & Gu, 1997). This process of segregation, analysis and generation of solutions should lead to the development of a design methodology. The methodology assumes the availability of the product designs and broadly includes the following steps which has been taken from Rao & Gu (1997):

Step 1:Requirements of manufacturing system design

Identification of requirements for manufacturing design starts with an analysis phase which includes the examination of the various factors that determine the needs of the manufacturing system. The needs of the system arise out of market analysis to determine the variation in product demand, product analysis to determine the complexity of the products; and manufacturing operations analysis to determine the operations needed to manufacture the products.

Step 2:Determination of manufacturing operations

Once product design, production volumes and forecasts are known, the next step deals with examination of the processes that should be used to manufacture the products. This is an important step as it governs the design of the entire manufacturing system.

The output from this step includes manufacturing operations, critical tooling requirements, accuracy and other technical constraints, which are essential for selection of machines.

Step 3:Selection and design of machines

Machine requirement design involves the determination of the operational requirements and machine capabilities for the system that needs to be designed.
Output of this stage is a list of machines to be purchased or designed, or which already exist.

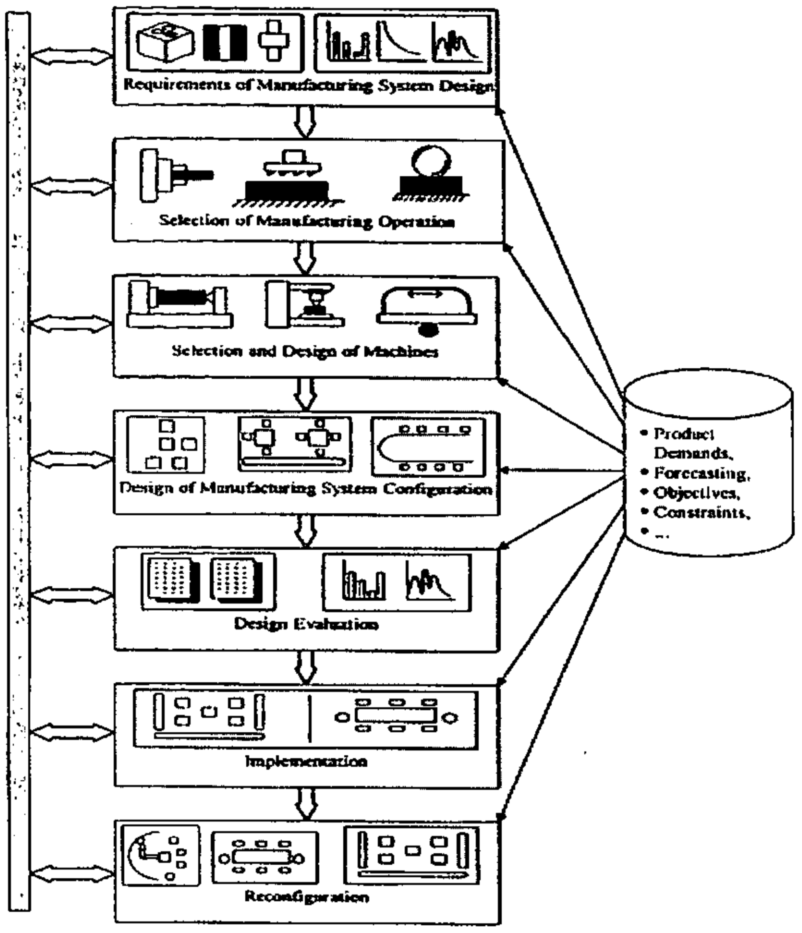

Requirements of Manufacturing System Design
Selection of Manufacturing Operation
Selection and Design of Machines
Design of Manufacturing System Configuration
Design Evaluation
Implementation
Reconfiguration
• Product Demands,
• Forecasting,
• Objectives,
• Constraints,
• ...

Fig. 3.4 Manufacturing System Design Methodology (Rao & Gu, 1997)

Step 4:Design of manufacturing system configurations

After machines are determined, the next logical step is to arrange the machines into a system, i.e. layout design. When determining a layout, in addition to the machine selection, production strategies (make to order or make to stock) and types of manufacturing systems (flexible manufacturing systems, cellular manufacturing systems or job-shop systems) and other factors are also considered. The selection of the proper type of system has a significant impact on the productivity, cost of the systems, and the flexibility for accommodating the future changes.

Step 5:Design evaluation

The layout must be evaluated and necessary changes suggested to meet the business objectives and constraints. Modelling and simulation tools are needed here to give comprehensive evaluations.

Based on the results, redesign of machines, configuration layout or material handling systems can be carried out. Essentially, the last three steps are iterative.

Step 6:Implementation

In order to establish models for design of manufacturing systems during previous design stages, some assumptions are normally made and some details have to be left out when developing models. Once the design is evaluated, good design candidates can be selected for implementation. In this stage, further analyses might be needed with more detailed information and practical constraints which cannot be considered in the previous stages. In the implementation, these details can be reviewed so that the implementation can achieve the balanced optimum results.

Step 7:System reconfiguration

This stage identifies the changes a designed system needs to undergo to accommodate the variation in the previously defined goals and objectives. It also determines certain aspects of system reconfiguration such as system life and intricate changes at the system, sub-system and machine level. It should be remembered that system design was carried out based on forecasts for a specified time period.

3.2.2.2 MSAD Methodologies

Design methodologies form an important part of manufacturing system design. Several system analysis and design methods have been used in the last two decades for analysing and designing different aspects of manufacturing systems (Al-Ahmari and Ridgway, 1998). Law and Tam, (2000), state that there are three types of approaches to system analysis and design: natural language approach, process-oriented approach and object-oriented approach.

- The use of **natural language** in system description is common. The disadvantage of this method is that it explains the system in a natural but informal language. A benefit of this approach is that it is accessible i.e. easy to understand but it has a lack of formality that leaves it open to misunderstanding.

- **The process-oriented or functional approach** is process and data oriented, emphasizing the processes that occur within a system and the data that flows as output from one process to become the input of the next process (Dori, 1996). The system is modelled as a network of interacting processes, with an established specification of how these processes can be performed (Law and Tam, 2000). Function/data methodologies can be represented by the methods like the data flow diagram method, which emphasizes process as the major theme of the analysis. At a high-level, the whole DFD can be described in a single 'Production' bubble in what is known as a 'content diagram' or 'zero-level diagram'. The internal details of the process can be exposed by 'explosion' i.e. recursively displaying lower-level DFD's of a particular process. Block diagrams, mathematical analysis, data flow diagrams (DFD), IDEF(integrated computer aided manufacturing definition) and Petri nets are some of the process-oriented approaches.
- In contrast to the process-oriented approach is **the object-oriented (OO) approach**, which puts the object at the centre of the analysis (Dori,1996). object-oriented analysis (OOA) is based on the principle that everything can be represented as an object (Booch, 1994). This includes tangible and abstract objects, activities, operations, and events. Objects have attributes and interact with each other by passing messages which invoke services or methods. The object paradigm combines behaviour and data and regards them as integrated objects. OOA encourages an analyst to concentrate on 'what' rather than 'how' and the information is organised around objects. It focuses first on identifying objects from the application domain, then on fitting procedures around them. During OOA and design there is an emphasis on finding and describing the objects - or concepts in the problem domain and defining logical objects that will ultimately be implemented in an OO programming language (Larman, 1997). One of the advantages of OO modelling identified in this process is that the subclasses can be hidden and represented by their super classes, using inheritance. The analysis can be focused on the organization of the super classes in order to design the general structure of the system. OOA and

modelling also facilitate the division of planning knowledge into logical groups and its organisation in a hierarchical way. By doing so, a better understanding of the domain problem and knowledge can be obtained. Another advantage is that when new resources are added to the manufacturing system, a new object can easily be created in the planning system without redefining its structure. The analysis and modelling of a process planning system using OO approaches also greatly improves the system reusability and portability. This approach supports the continuous reusability of partial modules of new models and the development of corresponding model libraries (Mertins et. al., 1997).

Common OOA and design techniques include: Rumbaugh's object modeling technique (OMT), Shlaer and Mellor's object-oriented system analysis, Coad and Yorden's OOA, and Booch's, Jacobsen's and UML approaches (Bahrami, 1999).

Law and Tam (2000) state that the use of OOA to create a CAPP system as the resultant object-oriented model can be used as a direct mapping during implementation. The implementation of the CAPP system is found to be nearly a one-to-one mapping from object-oriented model. In CAPP systems, computer systems are employed to code human knowledge and information in specific manufacturing domain.

The analysis and designing of the manufacturing system is a complex task , and work related to this topic has been discussed by many researchers such as Blanchard and Fabrycky (1981), Yeomans *et al.* (1985), Engelke *et al.* (1985), Talavage and Hannam (1988), Madison and Wu (1987), Lu and Subramanyam (1988), Nevins and Whitney (1989), Kusiak and Park (1990), West *et al.* (1989), Askin and Stanridge (1993) and Doumeingts *et al.* (1992). These methodologies aim to provide a generalized framework for the design of manufacturing systems. However many of the methodologies are specifically tailored for the design of cellular manufacturing systems, (Ferreira and Pradin, 1993; Ballarkur and Steudel, 1987; Gupta *et al.,* 1995; Shtub, 1989), flexible manufacturing systems (Borenstein et. al., 1999; Mak & Lu, 2000).

It is difficult to model a complete manufacturing system using a single method. Various attempts have been made to model manufacturing systems using combination of the functional and OOA approaches and the different methods available.

Doumeingts *et. al.* (1987) present a methodology for the design of advanced manufacturing systems. They make a complete survey of the conceptual models used in such methods, the various tools, and methods for each sub-system which includes decisional, informational and physical methods.

Spano *et al.* (1991) present a survey of the techniques for the design of FMSs. They point out that the design of FMSs is carried out by the independent examination of the different design phases and suggest the need for the development of tools and techniques for the integrated development of FMSs.

Law and Tam (2000) state that OOA can be used to create a Computer Aided Process Planning (CAPP) system as the resultant object-oriented model can be used as a direct mapping during implementation. The implementation of the CAPP system is found to be nearly a one-to-one mapping from the OO model.

A variety of issues related to the design of manufacturing systems can be found in many publications. For example, Noble and Tanchoco (1993) discuss in length issues related to the economic design of manufacturing systems. They also provide a comprehensive review of literature related to design methodologies and system design tools. Tompkins and Reed (1976) outline 25 requirements which are essential manufacturing system strategies that a manufacturer should address to be successful in a global marketplace. Tactical design issues have been addressed by Suh (1984) and Shingo (1988) and operational and implementation issues are discussed by Thacker (1989), Suri (1988), Wu (1992), Wysk *et al.* (1994), and Lardner (1988).

3.2.2.3 MSAD Tools

With the advances in manufacturing systems, improved analysis and design became crucial to cope with increasing levels of complexity. Computer Aided Engineering (CAE) tools appeared and offered support across a wide range of areas. According to

Molina et. al. (1995b), to realise the true value of concurrent engineering, modern CAE tools must operate in an environment which

(a) allows them to operate in parallel, and as early as possible in the design process,

(b) allows them to operate on a common source of information and

(c) provides a means by which conflicts which arise from competitive views of the evolving design can be identified.

CAE also provides computing tools for design for manufacture (DFM) (Boothroyd and Dewhurst, 1994). When a preliminary design has been computerised by means of computer aided design (CAD) software, this geometric design becomes the input for the next step in the 'product design process' namely analysis and evaluation, using further CAE tools.

For example, analytical packages, such as finite element analysis can be utilised to analyse the design mechanically. Simultaneously, computer aided manufacture (CAM) software, might be used to evaluate the component design, for manufacture.

Currently, there are CAM simulation systems designed for casting, sheet metal forming and injection moulding and CNC machining to name but a few. This software is beneficial for determining tooling requirements, time estimates, manufacturing bugs and programming (Howard, Lewis, 2003). Similarly, DFMA software might be used to evaluate the design as a whole and aid the designer in assembly and alternative manufacturing methods (Boothroyd and Dewhurst, 1994).

There are also CAE packages available for specific mechanical engineering design fields such as shaft design, bearing selection, as well as in-house customisable packages developed for company specific problems (Howard, Lewis, 2003).

A 3D design package being developed in at Department of Manufacturing and Operations Engineering, University of Limerick, Limerick, Ireland (Howard, Lewis, 2003), will expose the designer to a broader area of knowledge by reviewing a 3D component design for basic design faults and compares alternate methods of manufacture for producing the design.

GI-SIM (GRAI/IDEF-Simulation) is an example of an integrated system developed for supporting system analysis and design (Al-Ahrami, Ridgway, 1999). The method is based upon a reference model, formalism and structured approach. GIM uses three modelling methods: GRAI (to model decisional systems), MERISE (to model information systems) and IDEF0 (to model physical system). It presents a global view of the organisation in its grid, and describes different activity centres using the IDEF0 modelling technique. In this method, the lowest level of IDEF0 models can be translated into simulation tools. These features make the GI-SIM method powerful tool for analysing, designing and simulating manufacturing systems. GI-SIM is flexible and combines three important modelling concepts (conceptual, functional and simulation) to describe the manufacturing system from its global domain to its detailed specifications and includes the dynamic aspect of the system being modelled.

IMDE (Integrated Manufacturing Design Environment) was proposed as an integrated design methodology for manufacturing systems(Roa et al., 1993). IMDE enables the systematic integration of new design domains, methodologies, and technology. The IMDE architecture uses an integrating approach similar to a distributed database. Data resides locally with creating tools and a management system, which is responsible for linking entities. The IMDE is based on a practical, workable philosophy of using existing software-based design support tools and integrating new tools when necessary.

The computing tools mentioned above aid the design and prototyping of manufacturing systems. Again, these tools operate at different stages of the life cycle. Since they all create and or exploit design information and knowledge, potentially they all may make use of reusable components. Some of these tools provide data transformation or translation code to share or exchange data with other specific tools to design environment. Koonce et. al. (1996) suggest that an integration methodology be developed which takes advantage of the existing methods and models, so that they share essential information. According to Koonce et. al., in order to integrate the various design models into a unified model of the system, an architecture must allow designers to use the tools and methods best suited to their models.

Once the designing process is completed the behaviour of the manufacturing system can be exposed and investigated by using modelling techniques. Models provide very valuable abstractions of the manufacturing system, and may be built in many different ways, and refined to different levels of details, focusing the design on particular aspects of the required system. The structure of a model can be tailored to suit particular task; hence the information content of the model is dependent on the modelling approach adopted. The concept of information modelling is explored in the following section.

3.3 Applications of Information Modelling in Manufacturing

Information is the life-blood of any enterprise and without timely access to it the enterprise and its component parts cannot function effectively (Murgatroyd, *et al.* 1994). Information can be defined as knowledge of ideas, facts and processes while data are the symbols or functions which represent information for processing purposes, based on implicit or explicit interpretation rules (Doradore, Young, 1999).

An **Information Model**, is a formal description of types of ideas, facts and processes which together form a model of a portion of interest of the real world and which provides an explicit set of interpretation rules. The purpose of information modelling is to provide a representation of the information system of an enterprise at various modelling levels (Yu et. al., 1999).

Information models are a way of achieving common and structured sources of information. They provide an information repository, which is used to capture the information related to the life cycle of an artefact. To represent this information, well-defined information structures, or information data models, are required (McKay, *et al.* 1996). This element can be used to store a wide range of company information (Figure 3.5).

Software applications are responsible for supporting the life cycle functional activities involved in product development, such as design and manufacturing. Software applications share information stored in the information models, and hence it should be created based on the first element, i.e. information data model. For this reasons such software applications are also named data model driven applications (Young, *et al.* 1998) (Figure 3-5).

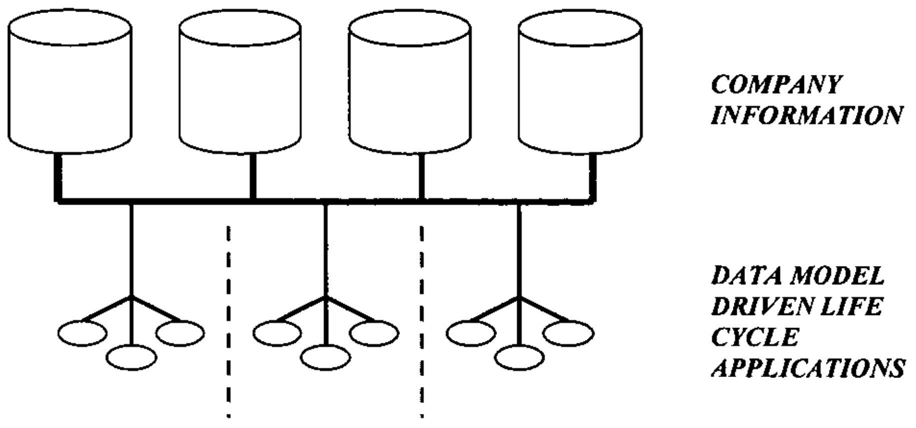


Figure 3.5 - The general information system concept (Young, *et al.* 1998)

Information modelling builds the information model, which usually consists of two main components, the *structure model* and the *process model* (Flynn, Diaz, 1996). The structure model describes the organisational and environmental elements about which information is to be recorded, commonly using the concepts of entity, attributes and relationship and showing these on an entity-relationship diagram. Whereas, the process model describes the elements concerned with processing the information, using concepts such as process, event and data flow and expressing these in terms of structure model elements.

In order to define the structure of the information models, there are basically two approaches that can be followed (Dorador & Young, 2000). The first is a top-down approach, which starts with the analysis of the enterprise level in order to obtain information about the processes, operations and objects that are involved in the process. The following steps are the definition of the information and computational levels. The other methodology is a bottom-up approach, in which the process starts with the definition of the functionality of the software, and from that point the information and enterprise levels are analysed.

In order to capture the requirements that must be contained in information models, several approaches have been proposed based on reference architectures (Dorador &

Young, 2000). Some of them have become standards, like CIM-OSA (Computer-Integrated-Manufacturing Open System Architecture) (Jorysz & Vernadat, 1990) and RM-ODP (Reference Model for Open Distributed Processing) (Toh, 1999). Other important methods are the Grai Integrated Methodology (Doumeingts, 1998) and the Prudue Enterprise Reference Architecture (PERA) (William, 1998).

Data models are formal representation of the structure of data for computer applications further, information models formally describe data so that it can be processes by computer (Schenck and Wilson, 1994). The provision of a common source of information, through data models, is one of the fundamental areas where a software system can support the design process (McKay, et. al.,1996). The necessary information to support designers can be classified into product information and manufacturing information, which can be represented in their respective product, and manufacturing models.

3.3.1 Product Modelling

In the course of its life a product is conceived, designed, manufactured, maintained and updated and eventually becomes obsolete. Throughout its life cycle, data concerning product is both generated and used as the product evolves. The design process can be defined as a series of activities by which the information about the designed object is changed from one information state to another, and hence need to solve a design problem (Dixon, 1995). Therefore, a clear and unambiguous definition and representation of the information involved in the whole product development process, is important to support design and other life-cycle applications. The concept of product modelling has been advocated by many authors. A **product model** is a representation of a product in a computer, and contains adequate information about the product to satisfy the product information needs of all the applications within a CAE system (Harding et al., 1999). A product model is a readable representation of all product related information (Kimura,1992, Krause, et. al.,1993, Anderl, 1997). A product model is a computer representation of product data that contains detailed information about a product or a family of products, so it can support the applications that are interacting in the product's life cycle (Dorador & Young, 2000).

A Product Model can be considered to be a computer representation of a product, which should hold a complete depiction of the information concerning a product. The product model therefore becomes a source of repository for information for all applications and allows information to be shared between the users and software components of a CAE system (Bugatai, 2002).

A product model was developed in the MOSES project. The aim of the research on the MOSES product model was to develop life cycle information relating to discrete products. The first STEP release was still a draft at the time the research was initiated even though some of its definition were used. However, the main structure of the product model proposed by the MOSES researchers at Leeds University used a different approach from that suggested by STEP, linking the product information to the elements of the product structure. One of the main contributions of the research undertaken was the framework of the data model that was used for multiple purposes (McKay et. al., 1996).

Product model structures

Product modelling is accepted as an important part of data exchange (STEP) and data sharing in integrated environments. **A product model** can be considered to be a computer representation of a product which should hold a complete depiction of the information concerning a product.

An example structure of product information is shown in Fig 3.6. The figure shows a product specification and product definition which are to be stored in the product model. The specification describes the requirement which the product must achieve. The product definition describes the ways in which the specification may be achieved and includes the product geometry, dimension and material and also captures the manufacturing plans, machine tool and set-up, defining operation and fixturing information.

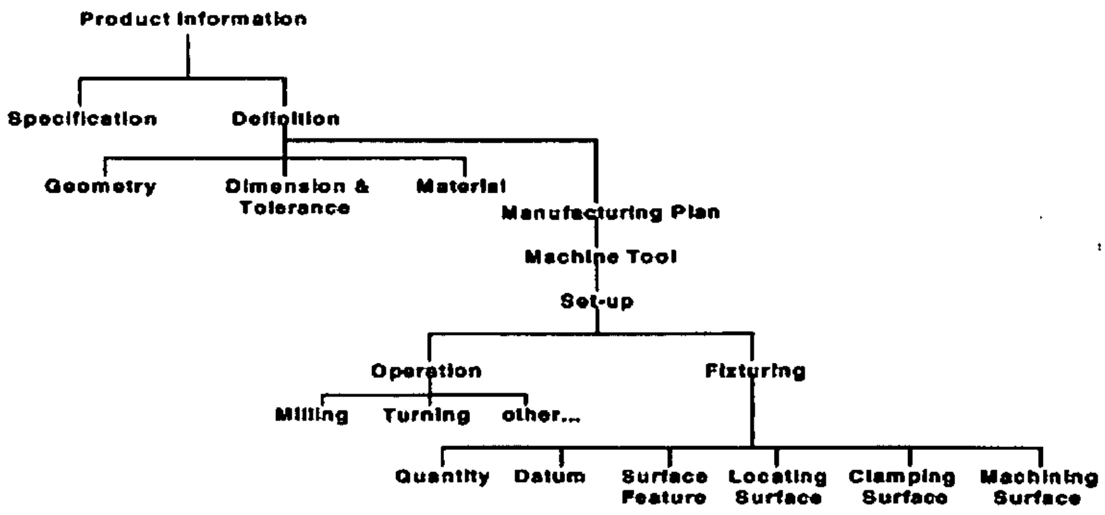


Fig. 3.6 Product data structure. (Bugtai & Young, 1998)

CAD, CAM, CAE and CIM systems with their ability to quickly generate and change product data have strained the conventional systems used to manage data (Kumar and Midha, 2001). Commercial CAD/CAM systems at different price levels are easily available for different users (Chao & Wang, 2002). Most of them have been developed for many years and have proven to be excellent tools for design and manufacturing engineers. The proliferation of CAD/CAM systems led to the need to provide an effective means by which product information can be shared and exchanged between applications and enterprises (Goh et. al., 1996). This need was recognised two decades ago (1980's) and as a result, a number of standards emerged (Vergeest, 1991). In a CAD/CAM context, there are several existing standards for data exchange, such as Initial Graphics Exchange Specification (IGES), SET, VDA-FS, EDIF, etc. (Bloor & Owen, 1991). IGES was designed as a neural format for the exchange of CAD data and has been used as the standard for geometric information by many CAD/CAM systems. IGES does not fulfil the complete requirements of representing product data. STEP, the standard for the exchange of product data, more formally known as ISO 10303, is an international standard directed at communicating the meaning of the data associated with a product. STEP defines a neutral data format for the representation and exchange of product data. The goal of this standard is to complete a system independent representation of all product related data during the product life cycle (Krause et al., 1993, Ash worth et al. 1996). It integrates geometric representation and adds additional information, such as the process models, for different stages of the product development. As such, it addresses a large portion of the information managed by many different systems. But STEP also offers a framework and supporting methods that can further aid the integration of multiple systems.

STEP includes a data definition language EXPRESS, that is also a part of the standard. All information models in STEP are defined using EXPRESS language constructs. EXPRESS is an object-like information modelling language whose purpose is to describe the characteristics of data about product that could exist in a database. It is defined.to be independent of any application that might operate on the data.

In many organizations, data are treated as the by-product of the business process. Data are stored in people's minds, filing cabinets and desktops, never to be re-utilized, only to be wastefully re-created elsewhere. As a result, the time spent by an engineer in looking for the right information far exceeds the time spent on the actual design work. There is a need to collate these data into a single source, providing complete and consistent information. This will free people to make decisions and will also improve data integrity within the organization (Obank et. al.,1995).

A product data management system is a tool that helps engineers and others manage both data and the product development process, and hence support a concurrent engineering framework in a company.

A product data management (PDM) system is a software framework that enables manufacturers to manage and control engineering information, specifically, data surrounding new product designs and engineering processes (Gascoigne, 1995). From the product perspective, it can help to organize design revisions, track versions of an evolving design concept, retrieve archived data and other product-specific information. From a process perspective, a PDM system can orchestrate procedural events such as design reviews, approvals, product releases and so on(Gascoigne, 1995).

A product data management (PDM) system is an enterprise solution that enables employees across the firm to quickly access product information (Wolfe, 1996; Kumar and Midha, 2001; Lee, 2003). PDM systems offer a wide range of functions, including engineering information management, engineering change management, and product structure management. (Gascoigne, 1995). PDM systems range from simple, off-the-shelf packages to complex, tailorable systems that can be further developed to fit a company's requirement exactly. PDM systems provide a centralized data repository that enables authorised users throughout a firm to access and update current product information, while ensuring that they follow specific procedures.

Manufacturers are able to optimise design, procurement, and manufacturing simultaneously by tightly integrating a PDM system with a component and supplier management (CSM) system or enterprise resource planning (ERP) system. Sharing information between PDM and ERP is essential for optimising design and manufacturing processes. PDM systems help in maintaining almost the entire product related information in an easily accessible form. (Kumar and Midha, 2001). PDM and ERP are the heart and soul of managing the overall product definition and production life cycles (Fulcher, 1998; Kempfer, 1998; Miller, 2000).

The application of knowledge in design automation has grown over recent years as vendors have begun to offer 'high-end computer aided design (CAD)' tools that can use knowledge linked with product geometry and data management, such as Parametric Technology's behavioural modelling, Unigraphics's UG/Wave, IBM/Ctia's Knowledgeware, and Knowledge Technologies International's ICAD(Duffy, et. al., 1998).

Moving beyond its heritage in computer aided design (CAD) and product data management (PDM) systems, PLM is evolving into a better understood, recognized, and appreciated enterprise business solution/process whose impact and influence extends across the entire enterprise out to suppliers and customers. Essentially, PLM is a business strategy that employs collaborative software solutions to create a product-centric data record for any product over its entire lifecycle, from concept to retirement. PLM encourages and coordinates interactions between a product information repository and all stakeholders, both internal and external to the enterprise, and this can provide the mechanisms for manufacturers to not only bring new and innovative products to market much faster and at a lower cost, but also effectively and efficiently support the product and customer.

IBM's PLM Express solutions consist of CATIA Version 5 the leading collaborative product design software and SMARTEAM for product data management developed by Dassault Systemes. Parametric Technology Corp.'s (PTC) Windchill is another example of PLM.

A new study by the ARC Advisory Group "PLM Software & Services Worldwide Outlook", highlights the massive commitment being undertaken by industry. It predicts that the PLM market, which totalled more than $5.6 billion in 2002, will more than double and top $14 billion by the end of 2007. This is a rise at a cumulative annual growth rate (CAGR) of 20 percent over this forecast period, a growth rate far in excess of any other major enterprise software market.

In 1995, Ford initiated the C3P program to create an integrated global CAD, CAE, CAM, PLM strategy. The C3P program demonstrated considerable success in capturing the collective intellect, i.e. the accumulated knowledge and specialized information of in-house engineering and external multi-tier supplier-based enterprise (Schnitger, 2003). He explained that a major source of a company's value is in the knowledge and expertise of the people in its extended enterprise. The C3P initiative now allows Ford to leverage that collective intellect for a competitive advantage in developing innovative products, shortening cycle times, reducing costs and improving quality.

A second model, representing manufacturing information, is also a fundamental requirement (Molina, et. al., 1995b) (Young and Bell, 1998). This manufacturing model is comprised of entities that are relevant and important for any type of manufacturing firm, namely: resources, processes and strategies. These manufacturing entities can be organised in functional levels to achieve a generic representation which can be tailored to suit different enterprises.

3.3.2 Manufacturing Modelling

A Manufacturing Model describes and captures the information about the manufacturing situation of a company in terms of its manufacturing facility and capabilities at different levels of abstraction (Molina, 1995) (Molina and Bell, 1999). A **manufacturing model** acts as a single source of information on available manufacturing capabilities and status and therefore helps to promote a common understanding of the manufacturing enterprise.

Manufacturing Modelling has at the centre of its philosophy the need to provide designers and manufacturing engineers with high quality manufacturing information, which can be easily accessed and shared (Molina, et. al. 1995b).

Three entities have been defined for manufacturing environments i.e. resource, processes and strategies, as these are relevant and important for any type of manufacturing firm. Being able to model all these three related dimensions in the Manufacturing Model allows a suitable description of the manufacturing facility in terms of its structure organisation, and capabilities, (Bugtai, 2002). The relations and interactions among them define the manufacturing environment of a company.

A de-facto standard has already emerged, by the petition of various standardisation bodies such as ISO and NBS (National Bureau of Standards), together with various European projects within ESPRIT (BSI PD 6526:1990). This de-facto standard had partitioned the factory into five hierarchical levels: Facility (Factory), Shop, Cell, Workstation and Equipment (Molina, 1995a). This hierarchical model has been used as a reference to structure the Manufacturing Model with the following four levels:

1. Factory Level
2. Shop Level
3. Cell Level
4. Station Level

where the fifth level (Equipment) is enclosed at the Station Level.

The manufacturing model describes and captures information regarding the manufacturing facility of a particular enterprise in terms of its organisation, composition and process capabilities (Bugtai, 2002). It represents the necessary information required to provide reliable manufacturing capability information for the support of life cycle activities, such as process planning, machine planning, pre-processing proving and scheduling (Molina, 1995).

Manufacturing model structures

According to Molina and Bell (1999), three entities can be regarded to be basic elements in the definition of any manufacturing environment, and these are resources, processes and strategies, (see figure 3.7). Manufacturing resources are all the physical elements within a facility that enable product manufacture, e.g. production machinery, production tools, etc. A description of the resources based on their physical properties and functional composition allows the capture of their capabilities. Being able to represent resource capability enables the support of design decisions, e.g. design for manufacture and manufacturing functions, process planning. Manufacturing processes are those processes carried out in a facility in order to produce a product. Strategies are decisions made on the use and the organisation of resources and processes. The "Manufacturing Model" as shown in figure 3.7 was structured into four levels based on a de-facto standard (i.e. factory, shop, cell, station), modelled in Booch Object-Oriented Methodology (Booch, 1994), and implemented in an object-oriented database.

Modelling of manufacturing systems, can also be achieved using simulation. Recently, Simulation has begun to be accepted as a very powerful tool for the planning, design, and control of complex production systems (Kunnathur et. al., 2004).

Discrete event simulation is one way of building up models to observe the time based (or dynamic) behaviour of a system(Ball, 1996, 2001) . During the experimental phase the models are executed (run over time) in order to generate results. The results can then be used to provide insight into a system and a basis to make decisions on.

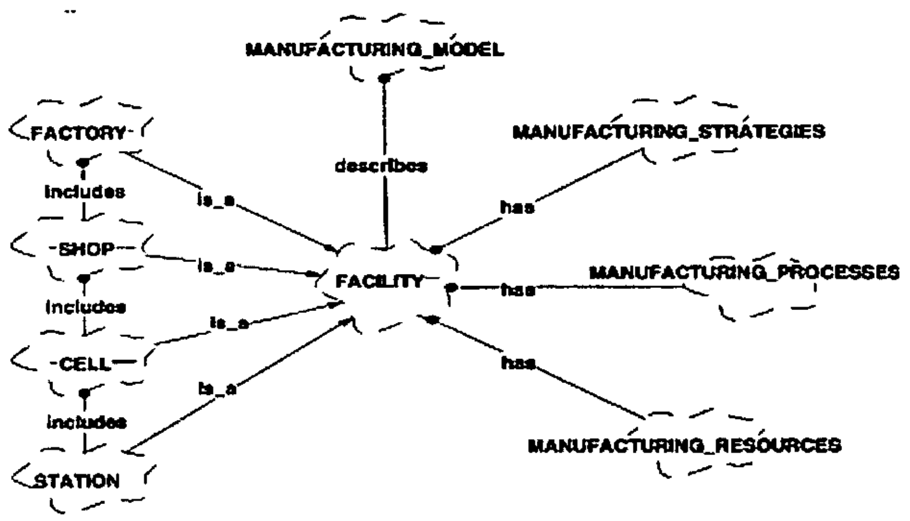


Fig. 3.7 Object-oriented representation of manufacturing model using Booch notation (Molina, 1995).

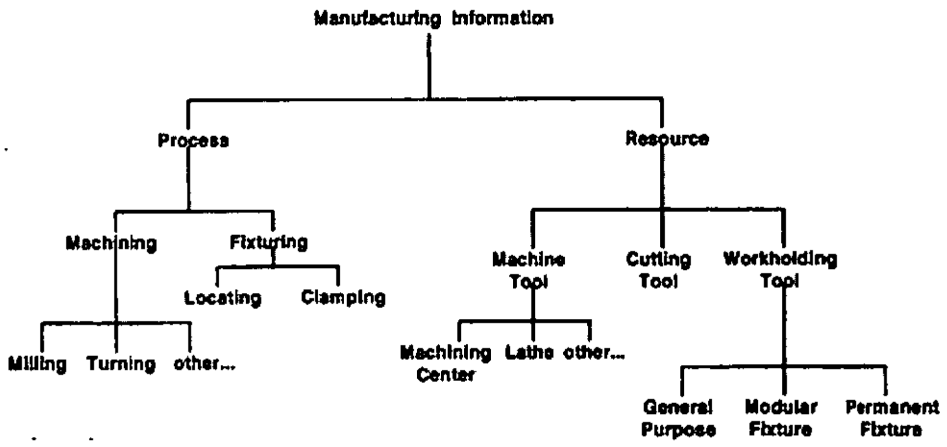


Fig. 3.8 Manufacturing data structure. (Bugtai & Young, 1998)

Simulation models can be build either by using a high level programming language or a data driven software system in which the model is specified using user-defined and default data items (e.g. machine scrap rate of 5% with default efficiency of 100%). Ball (1996, 2001) mentions that these two basic approaches can be categorised as – Languages and Simulators. *SLAM, ECSL, SIMAN* are examples of high level programming languages which allow rapid development compared with C++, Pascal, FORTRAN. It is a relatively slow approach to building models but extremely versatile. *Witness, ProModel, , TaylorII* are examples of data driven systems with little or no programming required. This approach is fast, easy but more limited in application. Some simulators are now incorporating virtual reality (VR) technology into their software. *Witness* now has a VR plug-in whilst Deneb market a package called *Quest.* A third approach can be derived from these two called Hybrid systems. *Arena* is an example of a hybrid system. It combines the flexibility of a simulation language (*SIMAN*) with the user-friendliness of a data driven system.

System modelling methodologies are discussed in more detail in the section 3.3.3.

The MANDATE (MANufacturing management DATa Exchange), ISO 15531, Industrial automation systems and integration- manufacturing management data exchange (MANDATE) includes the representation of data relating to the management of the production process and the exchange and sharing of management data within or between companies. MANDATE, in association with STEP, PLIB and other SC 4 (or non SC 4) standards, may be used in any software application that addresses manufacturing management related information such as resources management data, flow management data. The standard facilitates information exchanges between software applications such as E.R.P., manufacturing management software, maintenance management software, quotation software, etc.

MANDATE address the modelling of manufacturing management data such as:

- Resources management data (Resource model);
- Time related features (Time model);
- Flow management data in manufacturing (Flow management model).

MANDATE has been written in EXPRESS. During the development phases of the MANDATE standard, the compatibility of the standard with the ISO 10303 (STEP) standard has been the subject of a thorough analysis. (ref: http://www.iso.ch/iso/en/CatalogueDetailPage.CatalogueDetail?CSNUMBER=2814)

Three main categories of data relating to the management of manufacturing can be identified as the information related to:

1. external exchanges, e.g. with suppliers
2. the management of the resources used during the manufacturing processes
3. the management of the manufacturing flows within the plant and among the process stages.

ISO 15531 does not standardize the model of the manufacturing process. The aim of the standard is to provide standardized data models for the three types of data, above, which are usually complex, strongly time-dependent and with close relationships among them. The objective is to facilitate the integration between numerous industrial applications by means of a common, standardized tool able to represent these three sets of data that are shared and exchanged during the whole production life cycle and are in the core of the manufacturing process.

Ontologies are also increasingly being used in manufacturing. The PSL (PSL) project at the National Institute of Standards and Technology (NIST) is creating a neutral, standard language for process specification to serve as an interlingua to integrate multiple process-related applications throughout the manufacturing life cycle. This project is working/has worked closely with other efforts, such as A Language for Process Specification (ALPS) Project (Cator & Ray, 1991), the PIF (PIF) Project (Lee et. al., 1998), the Toronto Virtual Enterprise (TOVE) Project (Fox et. al , 1996), and the Enterprise ontology Project (Uschold et. al., 1998). ALPS was a research project at NIST which identified information models to facilitate process specification and to transfer this information to process control. The Process Specification Language (PSL) (Schlenoff et. al., 2003) targets process related environments such as manufacturing, construction and service industries. PSL also provides highly detailed concept definitions, but so far offers limited tool support. PSL was intended from inception to be a neutral format for the exchange (sharing) of process information.

PIF is an interchange format based upon formally defined semantic concepts, like PSL. However, unlike PSL, PIF is focused on modelling business processes and offers a single syntactical presentation, the BNF (Backus-Naur Format) specification of the Ontolingua Frame syntax. The TOVE project provides a generic, reusable data model that provides a shared terminology for the enterprise that each agent can jointly understand and use. The Enterprise Ontology project's goal is to provide "a collection of terms and definitions relevant to business enterprises to enable coping with a fast changing environment through improved business planning, greater flexibility, more effective communication and integration" (Uschold et. al., 1998). While both TOVE and the Enterprise Ontology focus on business processes, there are common semantic concepts in both these projects and the manufacturing process-focused PSL.

Ontologies in the context of system modelling are reviewed further in section 3.3.3.

3.3.3 System Modelling Methodologies

The concept of a manufacturing system was discussed in section 3.2 and now it is necessary to identify methodologies which can be used to design a manufacturing system in a systematic way. Some of the common approaches suitable for system modelling are System Analysis and Design Technique (SADT), the IDEF methods, SA/RT, the GRAI/GIM methodology, the IEM approach, the information system approaches, the object-oriented approach, Petri nets, ontologies.

SSADM (Structure System Analysis and Design Method).
(Dennis and Wixom, 2000, Kendall and Kendall, 1998)

This methodology consists of three stages: feasibility study, system analysis and system design. These are then sub-divided into problem identification, project identification, analysis of system operation and its problem, specification of requirements, selection of technical options, logical data design, logical process design and physical design. It is a comprehensive methodology for the development of data-processing and data-storage type of systems. However, it does not address the organization or resource views adequately. SSADM addresses the function view through Data Flow Diagrams (DFDs).

SADT methodology (Structured System Analysis and Design Technique)

SADT was originally developed at SofTech INC and supports the structured analysis of physical and information systems. SADT was developed as a method for detailed requirements definition, not as a CIM method or a system design method. Later on, SADT has become a full-scale methodology for problem analysis, requirements definition and functional specification applicable to many application domains. The major strength of SADT is that it is based on a structured methodology for decomposing complex system into functions, sub-functions, sub-subfunctions and so on. It is based on dual hierarchical structures, which are associated with a graphical representation, i.e. the so-called activity structure and data structure.

SADT consists of two main parts:

- a box-and-arrow diagramming language for structured analysis, and
- an analysis and design technique.

The diagramming language is based on a simple graphical notation, called the structured analysis box. This construct has a dual nature: one is called **actigram** (in the case of transformation) and is used to represent activities, the other one is called **datagram** and is used to represent data (in the case of information or document analysis).

The approach defined by SADT is divided into several phases – analysis, design, implementation, integration, testing, installation and operation. It is mainly used in the system analysis and design phases. SADT addresses both activity modelling and data modelling and is one of most applied methodologies in system design. (Wu, 1989).

IDEF

IDEF is a set of modelling methods each of which is designed for different purpose. IDEF is an acronym for ICAM Definition, (ICAM- Integrated Computer Aided Manufacturing) an initiative conducted by the U.S. Air Force in the 1970s. The aim of this project was to develop a methodology to analyse the data and activities in an enterprise, capturing the "as-is" process model and for modelling activities in an enterprise (the "as-is" state describes the actual state of the enterprise). The enterprise can develop a basis for process improvement planning and enable it to define information requirements to achieve the "to-be" state (the "to-be" state describes the desired, more integrated state targeted by the enterprise).

IDEF0 was developed based upon the SADT methodology and is a graphical modelling method. It facilitates the building of a functional model and provides a description of a system (typically a manufacturing system) in terms of an hierarchical presentation. The rectangle denotes a function (or activity) and arrows denote flow of data between activities. Input data to the left of the box is transformed and results are output from the right of the box. Control arrows to the top of the box denotes the constraints that affect performance of he activity. Mechanism arrows to the bottom of the box represent the resources that carry out the activities. To overcome the lack of information modelling, IDEF1 has been designed. It is developed for the design of integrated databases. However, its lack of any inherent means of describing inter-functional entities (Bravoco and Yadav, 1985b) led to the development of IDEF1X. IDEF1X is a graphical semantic information modelling technique which defines the logical structure of shared data in terms of entities, attributes of entities and relationships between entities (SofTech Inc., 1979)(Mackulak, 1983) (Hicks et al., 1986). The limitation of IDEF1X on dynamic modelling initiates the creation of IDEF2. IDEF2 is focused on improving the productivity of simulation modellers (Bravoco and Yadav, 1985c)(Sarkis and Li, 1994).

EXPRESS Language

The need to have a common representation of product data on which to link design and manufacture applications is now generally accepted and the International Standards Organisation have been working towards defining Standards for the Exchange of Product data through **STEP** committee for some years now. (Young, 1998). As part of this standard they have defined a language which enables the definition of data structure called **Express** (Young, 1998). More information about STEP can be found in Gu, Chan (1995).

Object-Oriented approach

The object-oriented approach is a very powerful and universal modelling tool, although based on only one modelling construct: the object. The main characteristics of the object-oriented modelling approach is encapsulation combining function modelling and information modelling into one unified paradigm. Objects are uniquely identified, have a state(i.e. a data structure) and possibly have a behaviour (i.e. a set of callable operations called methods and representing their functionality). They depict abstract or concrete things of the enterprise. The whole model is defined as a set of communicating objects. The object-oriented approach supports the continuous reusability of partial modules of new models and the development of corresponding model libraries (Mertins et. al. 1997).

UML

UML is an emerging industrial standard for object-oriented modelling. UML stands for Unified modelling language. It started as an effort by Grady Booch and Jim Rumbaugh in 1994 to combine their two popular methods, i.e. the Booch and OMT (Object Modelling Technique) methods. Later they were joined by Ivar Jacobson, the creator of the OOSE (Object-Oriented Software Engineering) method. In response to an OMG (Object Management Group, an industry standards body) request to define a standard modelling language and notation, the UML was submitted in 1997 as a candidate.

The UML is a language and notation for modelling, but it is internally not a method (Oestereich, 1999). A method needs to consider the specific framework and conditions of the application domain, the organizational environment, and much more. UML can serve as a basis for different methods, as it provides a well defined set of modelling constructs with uniform notation and semantics.

IEM (Integrated Enterprise Modeling) is largely based on object-oriented approach (Vernadat, 1996). IEM uses the object-oriented modelling technique for modelling business processes, related organisational structures and the required information systems (Mertins et. al., 1997). It provides a model for planning and optimising the processes and organisational structure within the enterprise.

The generic classes 'Product', 'Resource and 'Order' are the basis of IEM for developing models from a user's point of view. Product class represents the principle result of the entire enterprise process - the products. Resource class represents all means, including organisational units, which are necessary to carry out any activity in the enterprise. Order classes represent planning and control information.

Ontology

In recent years the **development** of ontologies, which are explicit formal specifications of the terms in the domain and relations among them (Gruber 1993), has been moving from the realm of AI laboratories to the desktops of domain experts. An ontology defines a common vocabulary to share information in a domain (Noy & McGuiness, 2000). An ontology consists of formal descriptions of entities and their properties, relationships, constraints and behaviour (Fox, et. al., 1996).

The ontologies on the Web range from large taxonomies categorizing Web sites (such as on Yahoo!) to categorizations of products for sale and their features (Noy & McGuinness, 2000). The WWW Consortium (W3C) is developing the Resource Description Framework (RDF) (Brickley and Guha 1999), a language for encoding knowledge on Web pages to make it understandable to electronic agents searching for information. The Defense Advanced Research Projects Agency (DARPA), in conjunction with the W3C, is developing DARPA Agent Markup Language (DAML) by extending RDF with more expressive constructs aimed at facilitating agent interaction on the Web (Hendler and McGuinness 2000).

The IDEF5 method provides a theoretically and empirically well-grounded method specifically designed to assist in creating, modifying, and maintaining ontologies. Standardized procedures, the ability to represent ontology information in an intuitive and natural form, and higher quality results enabled through IDEF5 application also serve to reduce the cost of these activities (http://www.idef.com/idef5.html).

In addition to the above well-known methodologies, a schematic modelling concept has been introduced to offer an advanced simulation methodology for system design. The appeal of this schematic modelling concept is its simplicity and flexibility, since many conditions can be defined in any desired way, i.e. interactions of functions performed by objects within a system (Chaharbaghi, 1991).

It is often useful to study a dynamic real-world system to learn something about its behaviour. However, it is generally necessary to use a model to study the performance of the system, since experimentation with the manufacturing system itself would be disruptive, not cost-effective or simply impossible. An appropriate model should shed some light on this question by allowing the operation of the plant to be studied as it currently exists and as it would be if the plant were expanded. A model, which is composed of entities and the relationships between the entities, can be constructed using mathematical methods, if the relationship is simple enough. Otherwise simulation can be used if the relationships are complex.

3.4 Enterprise Integration

Enterprise integration (EI) is very important topic in industrial engineering nowadays because of the growing need to improve existing industrial systems and to organise such complex systems faster, better, and in a more systematic way (Ortiz et. al., 1999). For manufacturing firms, the concept of integration is hardly novel. The Total Quality, JIT, and supply chain management movements all required improved internal and external coordination (Braganza, 2002). While these movements centred on the manufacturing function, research suggests that integration of several functions at different organisational levels can achieve above average financial and performance results. However, studies show that there are many problems associated with enterprise integration. A root problem is the fundamental assumption that all enterprise integration initiatives are equally important. Braganza (2002) challenges this assumption and argues that enterprise initiatives differ by their purpose; and he also proposes a framework for typifying enterprise integration initiatives that is based on the capabilities developed for the organisation.

EI means that each unit of the organization will have access to information relevant to its task and will understand how its actions will impact other parts of the organization thereby enabling it to choose alternatives that optimize the organization's goals. (Drakos, 1996).

Kosanke (1997) claims that the evolution in this field has gone through several phases, the initial objectives aimed at building the physical level integration; then the focus was moved to the integration of applications. The third stage of evolution was the integration of the business level and in the final stage the full enterprise operations were addressed, as shown in figure 3.9.

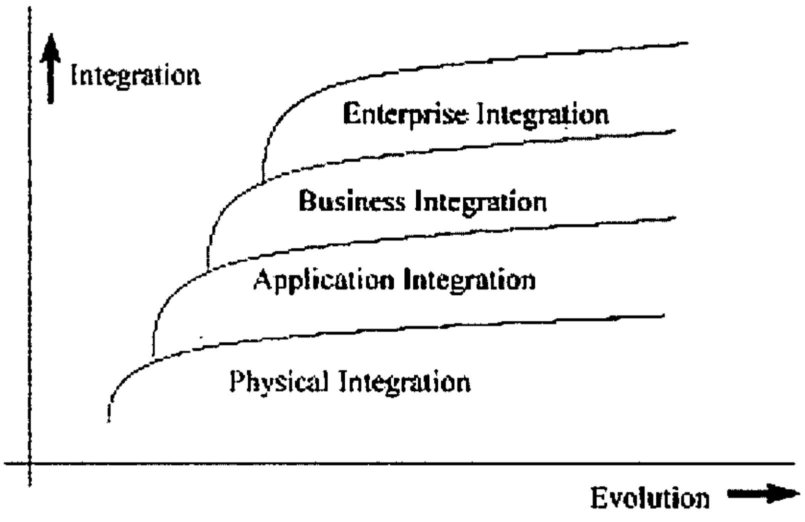


Fig. 3.9 Enterprise Integration evolution (Kosanke, 1997).

EI facilitates the material, information, decision and control flows throughout the organisation, linking functions with information, resources, applications and people. Its aim is to improve communication, cooperation and coordination in the enterprise, in order to manage the enterprise to behave as a whole and operate according to the strategy of the enterprise (Ortiz et. al., 1999).

Three key attributes emerge from the subset of the literature that focuses on EI. The first is the characteristics that define EI. The second is the scope of EI. The third theme is the range of organisational elements that would need to be co-ordinated, and hence changed in some way, as an integral part of an organisation's integration plans. Detailed explanation of these attributes can be found in Braganza, (2002).

The current need for enterprise-wide integration of business organisation can be explained by several reasons. Some of the most relevant ones are as follow (Ortiz et. al., 1999):

- The need to keep business operations aligned with strategy.
- The need to share enterprise information, (data used for decision making).
- The need to interoperate, i.e., the need for the different systems that exist in the enterprise to be able to work with each other, even across organisation boundaries (extended and virtual enterprises).
- The need to generate models and tools which let the users estimate the impact of the decisions taken in view of the globalisation of markets and the need for fast and effective response of enterprises

Various approaches have been proposed for EI (William, 1997; Esprit Consortium AMIC, 1993; Doumeingts, 1984; Jochem et. al., 1992; Bernus et. al. 1996). There are a number of reference architectures which have been developed to support enterprise integration e.g. CIMOSA (Kosanke,1995), GRAI-GIM, PERA, ARIS and ODP. However they all share some common characteristics which are (Young, Bell, 1998):

1. ***Structure:*** based on readily available and acceptable terminology, methodologies or standards.
2. ***Flexibility:*** able to be applied to a wide range of systems within its domain of applicability.
3. ***Generic:*** independent of any existing implementation.
4. ***Modular:*** open-ended in its ability to be extended in order to incorporate new concepts and technologies.

Several studies in which the different existing EI approaches are analysed can be found in Vernadat, (1997), Kosanke, (1997). Another very interesting survey is the one carried out by the IFAC/IFIP Task Force on architectures for EI (Bernus et. al., 1996). In this work, a complete study of the main EI architectures (Open System Architecture for Computer Integrated Manufacturing—CIMOSA, GRAI, Purdue Enterprise Reference Architecture—PERA) was carried out, and brought important information about the advantages, drawbacks and pending areas in order to progress towards EI.

As a result of this work, the GERAM proposal emerged (IFIP-IFAC Task Force, 1997). GERAM has evolved from a comparison grid to become a complete framework for comparing and checking the completeness of proposals and advances in the EI research field. Therefore, GERAM is not a proposal which can be directly applied (it has no constructs and no methodology of its own) in an enterprise.

PERA (William, 1992, 1994) is based on a detailed and pragmatic methodology covering the whole life cycle of an industrial project from inception to operation and even system disposal.

CIMOSA (Esprit Consortium AMIC, 1993) is an Open Systems Architecture for EI which includes a powerful language for enterprise modelling and is aimed at model-based integration (Vernadat, 1993; Zelm et. al., 1995). It provides a life cycle, but it does not cover the complete life cycle of a business entity as it starts with the Requirements Definition phase and ends with the phase of Implementation Description. The CIMOSA cube provides a consistent and adequately structured modelling framework, which is extended in GERAM into one which covers the full life cycle of enterprise entities.

3.4.1. Tools and IT for Enterprise Systems

Integrated information systems are often achieved using Enterprise Resource Planning (EPR) software. ERP systems are intended to provide standard application programmes that support the execution of activities throughout the organisation (Kennerley & Neely, 2001). In theory they enable the integration of operations, through common data processing and communications protocols. In addition to these theoretically appealing advantages, practical and immediate concerns of the late 1990s, such as the Y2K bug and the introduction of the Euro, further stimulated the market for ERP solutions. Integration is best achieved through a combination of technical and organisational process innovations (Sonde et. al., 2001). On a technical level, integration relies on the use of industry-standard common databases and common communication protocols. On the organisational process level, integration requires the simplification and streamlining of organisational processes using techniques such as business process reengineering and workflow redesign.

The last five years have seen an explosion in the implementation of enterprise resource planning (ERP) systems, such as those offered by SAP, Baan and PeopleSoft (Kennerley & Neely, 2001). Recent research indicates that the market for ERP applications will grow by 32 per cent over the next five years and that the total market will reach $66.6 billion by 2003 (AMR Research, 1999a), representing 43 per cent of the applications budgets of organisations (AMR Research, 1999b).

The origins of ERP systems can be traced back to the development of standard systems for the planning and control of manufacturing. Early systems were purely manual, relating to ordering material, hiring and firing people, following up and chasing work (expediting), until reorder point and economic order quantity (EOQ) techniques were added for the ordering of parts (Plossl, 1985). The development of materials requirements planning (MRP) in the 1960s and 1970s made it possible to plan material requirements based on future product requirements, rather than reordering based on past usage (Orlicky, 1975). The master production schedule was developed to drive MRP, relating material plans to products and options demanded by the customer (Vollmann *et al.*, 1997). The addition of shop floor control, capacity requirements planning and purchasing added a "closed loop" element to the planning system. The development of manufacturing resource planning (MRP II) further extended the planning and control activities to include production planning, business planning, financial and distribution systems within one computer system (Wallace, 1990).

ERP systems further developed the scope of standard software systems providing systems to support all business functions. Many organisations have implemented ERP systems in order to take advantage of the standardisation and integration of business systems supporting transaction execution and decision making. ERP systems have a very broad scope and their impact on the businesses in which they are implemented is potentially vast. However this scope also brings with it considerable complexity and the implementation process will often be lengthy, cumbersome and costly. According to Wortmann (1998), there has been a lack of academic attention paid to ERP systems.

Many organisations have invested significant sums of money into ERP systems, which are off-the-shelf packages that can be configured to match the needs of organisations (Markus *et al.*, 2000). At the heart of ERP packages is a database that enables organisations to structure data so that these can be shared across several applications. The database and its accompanying modelling and mining tools provide the opportunity for information to flow through the organisation (Oliver, 1999). However, the difficulties associated with implementing ERP, including the financial and time costs, can become unmanageable (Van Everdingen *et al.*, 2000). There is widespread agreement that for ERP implementation to succeed, organisations need to be integrated (Willcocks and Sykes, 2000).

SAP R/3 is an ERP system that is comprised of a collection of modules including financial accounting, cost controlling, materials management, production planning and human resources (Little & Best, 2003). Of these modules, only the financial accounting (FI) module is required for R/3 to function. The other modules provide further capabilities for the system and are integrated with FI. According to Professor Dr Henning Kagermann, SAP AG executive board member, "The SAP solution promotes integration of business processes across the entire enterprise, rather than relying on separate systems for each department or functional area, which may or may not interface with each other".

Some ERP vendors and third-party providers are beginning to bundle software with hardware. Examples are solutions by Pandesic (a now defunct joint venture of SAP and Intel), and DataWorks, which bundles the desired enterprise system (SAP, Oracle applications, or PeopleSoft) with the Microsoft platform (Horner, 1998). SAP has a product called RRR (Ready to Run R/3) preloaded and reconfigured on different hardware platform such as HP, Dell, and Sun. it is targeted for smaller business and is designed to get companies up and running more quickly.

Most hardware vendors have partnered with the major ERP providers to have specification programs and infrastructure specification that work with the ERP product. So once an organisation selects an ERP vendor, the hardware company can specify (and sell) the needed hardware to run the chosen software. HP, IBM, Dell, and Sun all have implementation services surrounding the infrastructure component selection processes.

Many benefits have been predicted if PDM and ERP systems are tightly integrated. The ability of companies to link effectively those two technologies will dictate the success of manufacturing organizations (Coates, 2000; Davenport, 2000; Sofranec, 2001).

Figure 3.10 illustrates how an information system can integrate advanced manufacturing technologies and other supporting functions, making possible a new approach to designing an ultimate manufacturing information system. It provides a shared database, a database management capability, and a communications network to link marketing, product development, design and engineering, procurement, manufacturing, and quality control functions, etc. This integration provides enhanced business capabilities which cannot be fully attained by the individual system's objectives (King and Teo, 1997; Tirpak, 2000).

Total Manufacturing Information Systems (TMIS) helps firms to build a strategic tool which functions at every stage of manufacturing operations, from gathering and analysing information for new products to developing business strategies through a totally integrated information system (Lee, 2003). An engineering manufacturing data system (EMDS) is a set of computer-based integrated applications which provide a firm with a common framework and a single access and control mechanism for all items of information, both hardware and software (Lee, 2003). It enables a firm to manage the costs incurred by the identified hidden factory. Changes to product design, inefficient purchasing and incident reports are now quantified and the implications effectively cross-referenced. This provides invaluable information, which is fed back into the process.

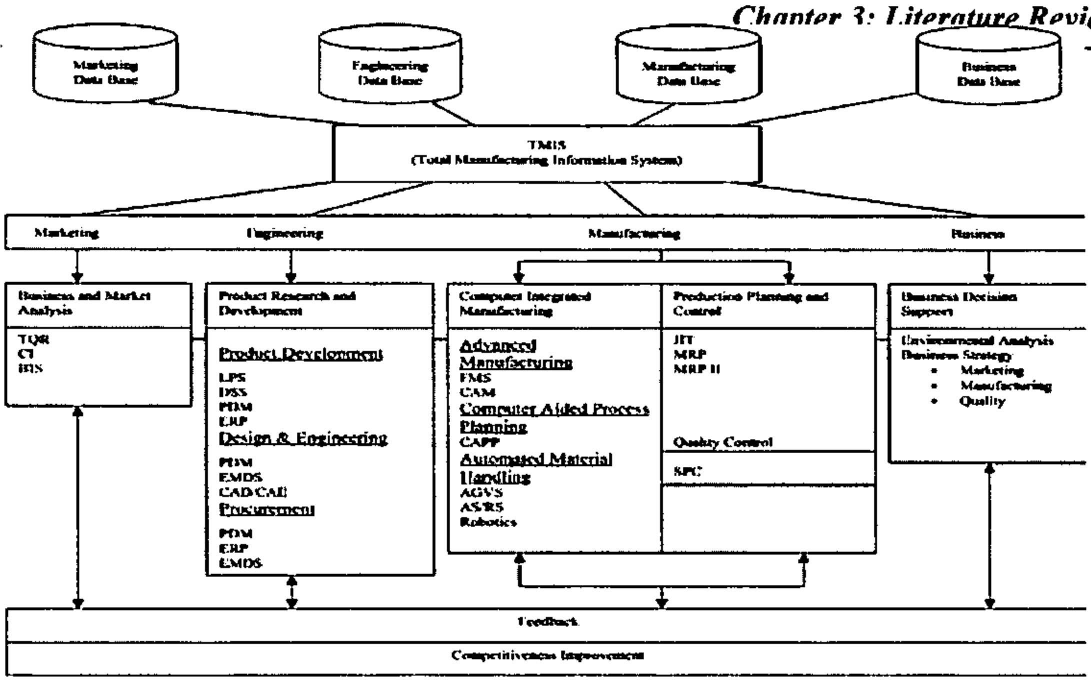


Fig. 3.10 Integration of manufacturing technologies and supporting functions (Lee, 2003)

As firms move toward TMIS, manufacturing components will be linked by both information flows and physical flows (Lee, 2003). TMIS provides a shared database, a database management capability, and a communications network to link engineering, flexible manufacturing and business decision support systems (BDSS).

In a typical TMIS, just-in-time (JIT) and manufacturing resource planning (MRP II) are two most popular tools for production planning and control that can be combined in a complementary manner (Lee, 2003). JIT can be used for short-term scheduling according to precisely timed customer delivery requirements. In contrast, MRP II is useful for longer term planning of labour availability, material procurement and capacity requirements. An integrated system, which utilizes the best attributes of JIT and MRPII can best address the changing needs of industry (Lee *et al.*, 1999; Coates, 2000). This is the reason why TMIS uses a hybrid system combination of JIT and MRP II for production planning and control.

All manufacturers must ensure that manufacturing uses only approved parts. Without PDM, workers could violate product configurations by calling out obsolete or unapproved parts for use in manufacturing. Manufacturers need a way to update automatically an MRP system's product configurations in near real time. PDM systems serve this purpose by transferring approved configuration data to the MRP system. Manufacturers are certain with PDM that they are using the right parts (Fulcher, 1998).

3.4.2 Enterprise Modelling

An **enterprise model** is one representation of a perception of an enterprise. It can be made of several sub-models including (but not limited to) process models, data models, resource models and organisation model (Lim, et. al., 1997). The content of an enterprise model is whatever the enterprise considers important for its operation. The process of building these models of an enterprise is called enterprise modelling. In other words, **enterprise modelling** is the process of building models of the whole or part of an enterprise from knowledge about the enterprise, previous models, and/or reference models as well as domain ontologies and model representation languages (Vernadat, 1996). Enterprise modelling and modelling tools can play a major role in the process of requirements definition, where the information structures, flows and transitions have to be accurately and reliably understood and documented (Toh, 1999). Enterprise models have to be able to cope with the modelling requirements brought about by rapid changes within and outside manufacturing, in order to capture the responsiveness required, for example, by new paradigms of sustainable production and development.

Enterprise models provide a data-driven and model-driven enterprise with several capabilities (Kosanke & Nell, 1999). Whether or not the integrated enterprise operates in a hierarchical, deterministic mode or in a distributed, chaotic mode, the enterprise model will provide the operator or executive, human or machine, with a map of the enterprise and some knowledge of what functions the enterprise comprises, in what state they are, and what capabilities exist at any moment to accomplish an output. If the models conform to some established framework, enterprises can seek, evaluate, set up, and go more easily toward inter-enterprise as well as intra-enterprise commerce.

An enterprise model allows more consistent modularisation so that enterprises can interchange pieces. Simulation can be used to allow evaluation of inter-operation with inter-enterprise entities and evaluation of systems with differing granularity. Enterprise models should be scalable so that a high-level model is essentially the same as a lower-level model—that is, use the same modelling constructs for all levels.

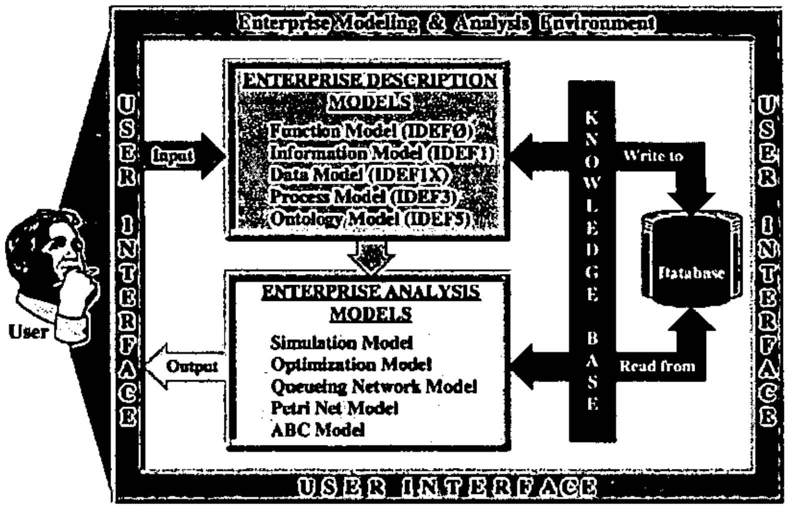


Fig. 3.11 A conceptual framework for enterprise modelling and analysis. (Dalen & Benjamin, 2003)

Innovators will continue to design enterprises by seeking optimum solutions (Kosanke & Nell, 1999), and this means that enterprise models will be continually updated and reorganised to improve processes and infrastructure. Increasing complexity of enterprise systems has stimulated the development of sophisticated methods and tools for enterprise modelling and analysis. Information systems that integrate these tools and techniques are commonly called Decision Support Systems (DSS), Enterprise Information Systems (EIS), and more recently, Business Intelligence Systems (BIS) (Turban & Aronson, 2002). Many industries have increasingly accepted enterprise modelling and analysis methods such as optimisation, simulation, Petri nets and cost analysis. Two key factors, have accelerated the use of such methods: (1) advances in information technology (increased efficiency in the collection and storage of information); and (2) significant progress in analytical and computational techniques. However, according to Lim et al, in 1997, enterprise modelling and analysis methods remain largely un-harnessed, and advances in modelling and analysis theories have yet to filter into the mainstream of managerial decision-making.

The idea behind developing enterprise modelling and integration frameworks is that a large part of the business process reengineering projects, as required by the integrated or the extended enterprise, are in fact similar and common to every type of business (Vernadat, 1996). Thus, they could be captured, standardised, and re-used instead of developing them again from scratch each time. Once standardised, the generally accepted frameworks can be supported by models, methodologies, and a range of compatible products, thus making the entire endeavour efficient in time and cost.

Another way in which enterprise modelling positively impacts analysis efforts is *reuse*. Because enterprise models are not committed to a low-level representation language (such as a particular simulation language), they provide the foundation from which a variety of analysis models can be built to satisfy various goals (Delen, 2001). Enterprise models created by a domain expert can be reused by a number of analysis method specialists to build a variety of analysis models.

Enterprise modelling has to be supported by enterprise engineering tools which implement methodologies and languages. The semantics of the modelling languages may be defined by ontologies, meta models and glossaries which are collectively called generic enterprise modelling concepts. The modelling process is enhanced by using partial models, which are reusable models of human roles, processes and technologies.

The new paradigm of enterprise organisation—extended and virtual enterprises—calls for easy collaboration between partners to exploit market opportunities with varying opportunity windows (Kosanke & Nell, 1999). Such collaborations can only be achieved if the partner contributions can be easily evaluated by means of process models which represent the relevant parts of the partner operations. However, these models have to be linked to represent the total enterprise of the new partnership and to allow performance evaluation by simulation. To provide the required model inter-operability, the models have to adhere to a common representation for both model enactment and human understandability. International standardisation is a means to provide the necessary model commonality.

An integrated approach to enterprise modelling and analysis has been proposed by Dalen & Benjamin (2003), that supports the following elements: capture of the entire enterprise model set within a single application, model integration, and extensions to the information types managed by the various model types. In order to implement this approach a commercial software application is under development called *-Model Mosaic* (Dalen & Benjamin, 2003),. The goal of the *Model Mosaic* is therefore to provide an integrated suite of enterprise modelling tools and an environment that can be easily be extended to provide an ideal setting for performing enterprise analysis.

Different enterprise modelling approaches are commonly cited in the literature as enterprise engineering, enterprise modelling, and enterprise integration modelling (Lim, et al., 1997; Gokhale et. al., 2002; Frank, 2002). Some of these research efforts have resulted in well-known enterprise modelling frameworks such as CIMOSA (Kosanke et. al., 1999), PERA (William, 1994), and ARIS (Scheer, 1999). Most of these modelling frameworks and reference architectures are developed for Computer Integrated Manufacturing (CIM) enterprises in supporting their enterprise integration efforts. In this context, enterprise modelling is defined as the process of creating abstractions of business functions and processes in order to conduct what-if scenarios for the purpose of validating various enterprise integration strategies (Ortize et. al., 1999). A key function of a reference architecture for enterprise creation, operation, and analysis is to determine, in specific and generic ways, what characteristics of an enterprise are necessary to analyse to help achieve an improved degree of enterprise integration. Once the key elements of these characteristics are logically arranged into a reference architecture, there would exist an excellent reference architecture for an enterprise model. Therefore, one could view an enterprise-reference architecture as a high-level enterprise model or a meta-model for a set of enterprise models. The elements of the reference architecture would be a framework that would indicate the key things in the enterprise that one should consider when creating, analysing, or using an enterprise model (Bernus et. al., 1995;Kosanke & Nell, 1997).

With well-designed standards about enterprise-representation models in place to provide a known environment to the developer, the risks of investing in an island of integration will be significantly reduced (Kosanke & Nell, 1999). One area where standards are important to help with the enterprise engineering and integration work is in enterprise-process representation. The ISO standards group to develop standards in this domain is TC184 SC5 WG1, *Industrial-automation systems and integration, Architecture, communications, and integration frameworks, Modelling and architecture*. WG1 is planning a family of standards that will help manufacturers, implementers, software developers, and other standards makers to create consistent environments in which the integration process can progress.

To engineer and improve the integration level of an enterprise WG1 can envision standards in four key areas: process representation, integrating infrastructure, a semantics-resolving utility, and representation of human involvement.

The first standard produced by WG1 is ISO 14258, *Concepts and rules for enterprise models* (ISO 14258, 1996). This is a high-level standard defining the nature of enterprise models with the vision that compliant models could be used to design, analyse, and eventually, operate enterprises. The rules for models are based on classic systems theory, with the assumption that an enterprise or groups of processes is basically a system and that it can be designed and analysed as such. ISO 14258 is the most general standard of the planned series from WG1.

A second standard has been developed: ISO 15704, *Requirements for enterprise-reference architectures and methodologies* (ISO 15704, 1998) (Bernus et. al., 1995). ISO 15704 defines the requirements that enterprise-reference architectures and methodologies must have to be considered complete. This standard attempts to place the concepts used in methodologies and reference architectures such as ARIS, CIMOSA, GRAI/GIM, IEM, PERA and ENV 40003 within an encompassing conceptual framework that allows the coverage and completeness of any such approach to be assessed. It draws heavily on the work of the IFAC/IFIP Task Force on Enterprise Integration and previous work from Purdue University (Specification and requirements for GERAM) (http://www.cimosa.de/Standards/IS15704.html). Potentially, all proposed reference architectures and methodologies could be characterised in GERAM so that developers of particular architectures could gain from being able to commonly refer to the capabilities of their architectures, without having to rewrite their documents to comply with GERAM. This will be useful to those trying to improve an enterprise infrastructure or its processes, and who will create an enterprise architecture of their own that is specific to a company, industry, or purpose. This standard will help guide that creation process. Previous work in CEN had developed ENV 40003 (1990) Framework for Enterprise Modelling, which is a partial implementation of these requirements.

The GERAM framework identifies a set of components that are essential for enterprise engineering and integration. The enterprise reference architecture identifies the basic concepts to be used in enterprise engineering and integration (e.g., enterprise entities, life-cycles and life histories of enterprise entities). GERAM distinguishes between the methodologies for enterprise engineering and the modelling languages that are used by the methodologies to describe and model the structure, content and behaviour of the enterprise entities in question. These languages will enable the modelling of business processes and their supporting technologies as well as the roles of the human in the enterprise operation. The resulting enterprise models represent all or part of the enterprise operations, including its manufacturing or service tasks, its organisation and management, and its control and information systems. These models can be used to guide the implementation of the operational system of the enterprise as well as to improve the ability of the enterprise to evaluate operational or organisational alternatives (for example, by simulation), and thereby enhance its current and future performance.

3.4.3 Enterprise Application Integration

Today, most organisations are using packaged software for their key business processes. Enterprise resource planning, supply chain management, customer relationship management and electronic commerce systems enable organisations to improve their focus of using information systems to support their operational and financial goals. The continued development and progression of computing and IS has led many organisations and indeed economies, to transform themselves. The usage and implementation of information technology (IT) has allowed many organisations to also institute information systems (IS) that encode and embody the business strategy completely (Davenport, 1998; Doukidis *et al.*, 1998).

The need for integration is not new but it has increased since applications moved from central processors to distributed environments and networks. In the past, many companies used electronic data interchange (EDI) technology and value added networks (VAN) to exchange their business documents in an integrated way, and piece together their supply chains (Emmelhainz, 1993). Although EDI achieves data integration, it is not adequate for enterprise and cross enterprise incorporation as it has a number of drawbacks (Choudhury, 1997; Kim and Umanath, 1999). According to Nissen (2000) EDI is a complex and invasive technology that does not achieve process integration. EDI is unable to provide the flexibility and maintainability demanded by global business with comprehensive IT infrastructures that differentiate them from their competitors.

In recent years, organisations have seen the introduction of enterprise resource planning systems (ERP) as a solution to their integration problems. Although ERP systems consist of a set of internally integrated modules that automate generic business processes, they were not designed to collaborate with other autonomous applications (Kalakota and Robinson, 2001). Enterprise systems present many limitations in integrating business and cross enterprise business processes, with Themistocleous *et al.* (2001, 2002) critically exploring these limitations.

The increasing deployment of enterprise applications alongside legacy systems has meant that companies are being compelled to adopt IS infrastructures that connect applications, data and information together. The approach that provides a solution to this is enterprise application integration (EAI). Recently, enterprise application integration (EAI) has been introduced to piece together applications to overcome the limitations of existing integration technologies and practices (e.g. EDI and ERP). EAI combines a variety of integration technologies such as message brokers, adapters and application servers, to build a centralised integration infrastructure. Such an evolution has dramatically reduced the time for integration, as developers prefer EAI packages to individual integration technologies. However, in the majority of cases, integrators need to combine integration technologies with EAI packages to develop an integrated infrastructure. The reason for this is that there is no single EAI package that provides all the functionality needed by organisations. There is therefore a need to support organisations in their attempt to select appropriate EAI technologies and packages.

Enterprise integration is considered to be of great strategic significance in the support of organisations to achieve a competitive advantage (Thermistocleous, 2004). As previously stated, traditional approaches of EDI and ERP address some, but not all of the challenges of integration. There has been a great demand by organisations to overcome integration problems and become more competitive. In this respect, enterprise application integration (EAI) has emerged to address intra and inter-organisational integration in a more flexible and maintainable way. Literature remains limited on this emerging area and there is consequently a need for further research and contribution in identifying influential factors for EAI adoption.

EAI is considered by Irani *et al.*, (2003) to be an innovation, as it is a new technology that can significantly change the way of doing business and extend the IS lifecycles. Puschmann &Alt (2004), believe that today's available EAI systems address different integration levels and support organisations in different problem areas. Therefore organisations need to have a clear understanding of their future IS architecture and balance their integration approach among different solutions.

Existing work in IS planning and evaluation had already shown the need to address any IS project in its wider organisational context (e.g. Coakes and Elliman, 1999; Clarke *et al.*, 2000). Thus, IS integration within the enterprise should also encompass more than just the technological aspects. Not only should EAI and enterprise IS include concepts of information integration in terms of integration across data, information and processes, but it should also be considered that integration across many levels of the IS organisation itself, needs to be achieved.

EAI is also a broad concept covering a wide range of EAI software products, processes and methods, and research is needed to classify the business problems and contexts that are appropriate for each of these offerings. Not only does it include the concepts of information integration in terms of integration across data, information, and processes, but it also considers integration across many levels of IS organisation . The organisational processes and the data elements need to be integrated on four levels : Data level, Application interface level, Method level and User interface level (Linthicum 2000).

Data Level EAI: Data level EAI is the process, techniques and technology of moving data between data stores. This can be described as extracting information from one database, perhaps processing that information as needed, and updating it in another database. While this sounds direct and straightforward, in updating it in a typical enterprise, it might mean drawing hundred databases and several thousands of tables. It may also include the transformation and application of business logic to the data that is being extracted.

The advantage of data-level EAI is that it is relatively inexpensive compared to the other EAI levels and their application technology.

Application Interface Level: Application level EAI refers to the leaving of interfaces exposed by custom or packaged applications. These interfaces enable many applications to be bundled together, allowing them to share business logic and information. The only limitations that developers face are the specific features and functions of the application interfaces.

This type of EAI is most applicable to packaged applications such as SAP, PeopleSoft, and Baan, which all expose interfaces into their processes and data, but do so in very different ways.

Method Level EAI : Method level EAI is the sharing of the business logic that may exist within the enterprise. For example, the method for updating a customer record may be accessed from any number of applications, and applications may access each other's methods without having to rewrite each method within the respective application.

The mechanisms to share methods among applications are numerous, including distributed objects, application servers, TP (transaction processing) monitors, frameworks, and simple applications that's the combination of two or more applications.

User Interface Level EAI : User interface level EAI is more primitive, but nonetheless necessary, approach. Using this scenario, architects and developers are able to bundle applications by using their user interfaces as a common point of integration (also known as screen scraping).

Systems that enable companies to implement inter-process communication are traditionally known as ***middleware***. Middleware is primarily concerned with data level integration, which means that these systems do not provide any functionality that enables higher levels of integration, such as object or process integration (Bernstein, 1996; Linthicum, 2001). In contrast to this, EAI systems encompass the technologies as well as the process definition to enable custom-built and/or packaged business applications to exchange semantic level information in formats and contexts that each system understands (Linthicum, 2000). This means that these products not only integrate applications on a technical layer, but also provide a communication framework that underpins the integration of information systems on a semantic and pragmatic level. EAI systems offer several types of services in addition to classical middleware - but just like middleware, none of these services alone is new (Bernstein, 1996; Linthicum, 2000; 2001; Ruh *et al.*, 2000).

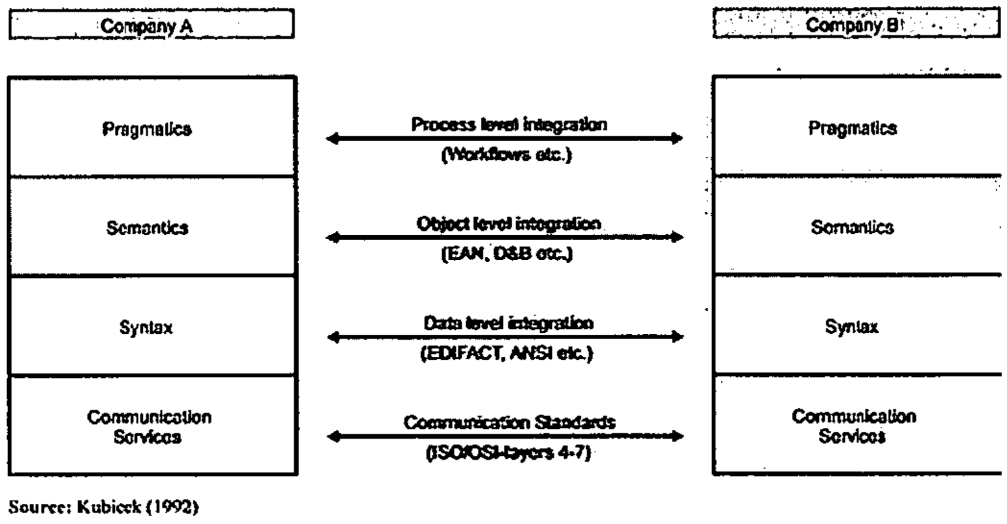


Fig. 3.12 Integration levels (Kubicek, 1992)

Application integration has been going on for years, using a broad range of connection technology (Linthicum, 2000). In the past this has been low-level play with developers working at the network protocol layer or just above, then moving to true middleware solutions such as RPC's, MOM, and transactional middleware. The next generation of middleware is here with new categories such as message brokers, application servers, distributed objects, and intelligent agents. However, expect more middleware and middleware categories to emerge as more people become interested in EAI.

The benefits that EAI provides are important as they reduce the cost of integration and the redundancy of data and functionality. Organisations that have implemented EAI solutions have reported significant benefits that support the IT/IS evaluation process (Irani and Love, 2002). Themistocleous and Irani (2001a) analysed and explained further the benefits that derive from the use of EAI technology and classified them into:

- organisational (e.g. results in more organised business processes);
- managerial (e.g. achieves return on investment);
- strategic (e.g. increases collaboration among partners);
- technical (e.g. achieves data, objects and process integration); and
- operational (e.g. reduces cost).

Evidences from case studies published by Edwards and Newing (2000) highlight that the anticipated benefits derived from EAI may be a factor influencing the adoption of EAI.

Ring and Ward-Dutton (1999) among others, support that no integration technology solves all types of integration problems, as each technology was designed to address a broad category of integration issues. It has been suggested that there is a correlation between appropriate technologies and specific integration problems. Themistocleous (2002) has gone some way to proposing and empirically validating an evaluation framework for assessing integration technologies. The framework clarifies the differences among EAI technologies and helps integrators to select an appropriate combination of integration technologies that fit their integration problem.

Published case studies on application integration such as Edwards and Newing (2000), European Committee for Standardization (2001), Puschmann and Alt (2001) and Themistocleous and Irani (2001a, b; 2002) support idea that organisations evaluate all types of benefits (e.g. managerial, technical) that EAI offers before proceeding to the introduction of this new technology. This is perhaps not surprising as IT/IS evaluation remains an important part of organisational decision making (Irani and Love, 2002).

Barriers to the introduction of EAI are similar to the barriers that exist when introducing ERP, namely it:

- promises to integrate IT infrastructures;
- introduces changes to organisations, their structure and the way of doing business;
- influences the employees tasks as well as inter-organisational relationships;
- costs a lot of money; and
- is more likely to be adopted by large organisations as an integrated technology.

Since there is a lot of failure of ERP adoption (Markus and Tanis, 1999; Markus *et al.*, 2000), organisations tend to estimate the possible impact of the adoption of application integration before proceeding to its adoption (Themistocleous, 2002). Barriers are also reported by Chwelos *et al.* (2001) and Ling (2001) as a factor that influences the adoption of EDI technology.

The advent of resource and schedule planning systems, such as enterprise resource planning (ERP), has meant that the sharing of common data and information across the enterprise is now an inherent necessity for many organisations (Eck and Marchetti, 2000). As such, it can be said that although EAI is not necessarily a novel approach to integrating IS components, it is a tool which has proved to be useful when the integration of many disparate systems needs to be carried out (Linthicum, 1999).

An analysis of current research approaches can be found in Puschmann & Alt, (2004). According to Puschmann and Alt (2004) these approaches lack in two dimensions. First, most of them have an intra-organisational integration scope. Only a few approaches consider inter-organisational integration as their focus. Second, nearly all approaches encompass only one or two integration objects concerning data, object and process integration. Most of the approaches show very technical standards for data or object integration.

3.5 Design Reuse

3.5.1 Design Reuse Concepts

Sivaloganathan and Shahin (1999) state that design reuse is aimed at maximising the use of engineering creativity and expertise in design, by reuse of successful past experience in part or in whole for new designs. The authors discuss the current research carried out in design reuse based on a classification of different categories: focused innovation, cognitive studies on design reuse, computational perspective of design reuse, use of standard components, design reuse tools and methods, design reuse systems, and issues in design reuse.

Fothergill, Arana et al. (1996) define design reuse as "the design of a specific article to satisfy a customer specification in the context of an existing history of designs of similar articles forming a product family. The assumption is that the overall functional requirements in the product family remain the same, and that re-design does not involve the synthesis of new ways of solving design problems. It does involve re-use and adaptations of existing solutions and parts, and, possibly, new combinations of sub-solutions".

In spite of all attractions offered by reuse, (Ormerod, Mariani et al. 1997; Busby 1999) explain that some problems can be identified in design practice, such as problems with encoding reuse information, problems in situating reuse within the design process, and problems in retrieving reuse information. Finger (1998) considers that functions, behaviour, form or even context can be used to retrieve a prior design, however there is no formal representation available for these attributes.

Shahin, Andrews et al. (1999) state that some relevant pieces of information, defined by the authors as data models to support design reuse are: a list of prioritised requirements, function list, annotated function and means, parts tree and feature-based model of parts and assemblies. These data models are based on the structure of the DFD (design function deployment).

However, when aiming for the computational application of design reuse into CE environments, other issues must also be considered such as the computational approach that can be applied to the retrieval process, representation of the design process and representation of the more appropriate information structures.

3.5.2 Redesign

Three main categories of design problem have been identified within the literature, on the availability of the information and knowledge during the design process (Al Hamando and Kumura 1994; Evbuomwan, Sivaloganathan et al. 1996):

i) **Original / Non-Routine design:** is characterised by ill-defined goals, development of new concepts and lack of knowledge. This kind of design can be further divided in to innovative and creative;

ii) **Redesign:** expands the boundaries of the existing design principles to adapt functions and make other changes for the new product, this includes knowledge about decompositions that may be available. However modifications may make the acquisition of new knowledge necessary. This type of redesign can be divided in adaptive and variant design;

iii) **Routine design:** is when sufficient knowledge exists about function and goal structures. Compiled plans are available for goals and sub goals, and modifications are restricted to some features;

Among the categories presented above, redesign has become the most practiced and popular kind of design, allowing the reusability of the knowledge and information from past experience and previous products. Additionally, this approach allows companies to have a maximum payback for each new development through the production of different products (variation) based on the same design concept. (Fowler 1996) defines variant design as "a technique supporting retrieval of an existing design specification for the purpose of adapting that design specification for use in the design of a new, but similar artefact". (Evbuomwan, Sivaloganathan et al. 1996) states that variant design is a kind of redesign this utilizes a proven design as a basis for generating further geometrically similar designs of differing capacities.

Fowler (1996) also states that during the redesign process designers

- Can seek inspiration for a solution from existing design solutions;
- May have a conceptual solution in mind but seek for design solutions that are conceptually similar, or
- Can have an overall idea of the structure and organisation of an appropriate satisfying artefact but can finish faster given a previously designed solution.

The above activities raise issues related to storing, manipulation and retrieving of information and past experiences to support design, and indicate that computational tools can be applied quite successfully. (Finger 1998) identifies these issues as representing, capturing, organising and retrieving the design knowledge, and addresses the issue that there is a need for knowledge and information representation of an artefact, in order to develop computational environments to support design reuse. CAD systems provide extensive support in the detailed phases of design, through features-based modelling and parametric design, however actual redesign is not supported since no reuse of information is performed (Fowler 1996; Finger 1998). Therefore, applications of computational tools to support design reuse of information have gained significant attention by industry and the research community in recent years.

3.5.3 Computational Approaches to Design Reuse

The application of computer based systems to support design reuse has increased in recent years, as can be seen in the Proceedings of the Engineering Design conference'98 Sivaloganathan and Shahin (1998).

Duffy, Smith et al. (1998) identified research work in computer based systems focused on supporting design reuse and classified them within three main computational approaches, namely: (1) indexing and information retrieval; (2) knowledge utilisation, which is further divided in case based reasoning, model based reasoning, plan reuse and customised viewpoints, and (3) exploration and adaptation. They also presented and discussed a comprehensive comparison of the reviewed systems against an existing design reuse model process.

3.5.4 Computational Support to Design Reuse

The development of a product involves several activities and steps in order to conceive, design and commercialise it. Ulrich and Eppinger (2003) present these steps as Concept Development, System-Level Design, Detail Design, Testing and Refinement and Production Ramp-up phases, where the design phases have been recognised as one of the most important activities during a product cycle life.

Dixon (1995) defines the design process as a series of activities by which the information about a designed object is changed from one information state to another, and hence a design process is needed to solve a design problem. Such a process requires different types of knowledge. This opinion is also shared by (Court, Culley et al. 1995).

Evbuomwan, Sivaloganathan et al. (1996) present a comprehensive review about design, discussing aspects such as definitions, theory and methodology, classifications, philosophies, models, methods and systems. The authors describe three main kinds of design models, namely: prescriptive, descriptive and computational. While prescriptive models tend to look at the design process from a global perspective, covering the procedural steps, e.g. (Pahl and Beitz

1996), descriptive models are concerned with designers' actions and activities during the design process, e.g. Suh's Axiomatic Approach (Suh 1995). Computational models are related to the application of computational technologies to support design (Al Hamando and Kumura 1994). The research presented in this thesis is mainly concerned with computational support for design reuse, more specifically focused on an information based approach, and therefore particular attention has been given to literature covering this area.

To provide computational support for design activities, models that represents such activities, i.e. information and process models, should be understood and built (Krause, Kimura et al. 1993; Al-Salka, Cartmell et al. 1998). Many example of different design models and design process models have been identified and considered during this research, because of their possible relevance in generating reusable components. A few of them are briefly listed here, Grabowski, Lossack et al. (1996) address two main approaches used in design systems, namely process oriented (strategies for solving design problems) and information oriented (modelling information needed for design). Gorti, Gupta et al. (1998) address the importance of modelling the design process together with the modelling of design products, i.e. artefacts, for automating the design process and representing design history. Works dealing with models to capture the design process or design rationale, in either general cases or more specifically focused on specific design phases, can be found also in (Dowlatshahi 1994; Dowlatshahi(l) 1994;Lahti,Mantyla *et. Al.* L997;Gao,Zeid *et al.* 1998).

3.5.5 Types of Reuse

Kovacs et. al. (1999) state that four basic types of software reuse can be distinguished: vertical, horizontal, diagonal and point reuse as shown in figure 3.13. In horizontal reuse the (software) components are used across different domains. In diagonal, vertical and point reuse the components are used in the area within a specific domain. In diagonal reuse the components are utilised in the same domain but in different projects. In vertical reuse the reusable elements are applied in the same projects, but in different version of software. Point reuse means that components are employed in the same version of software, where they were extracted from.

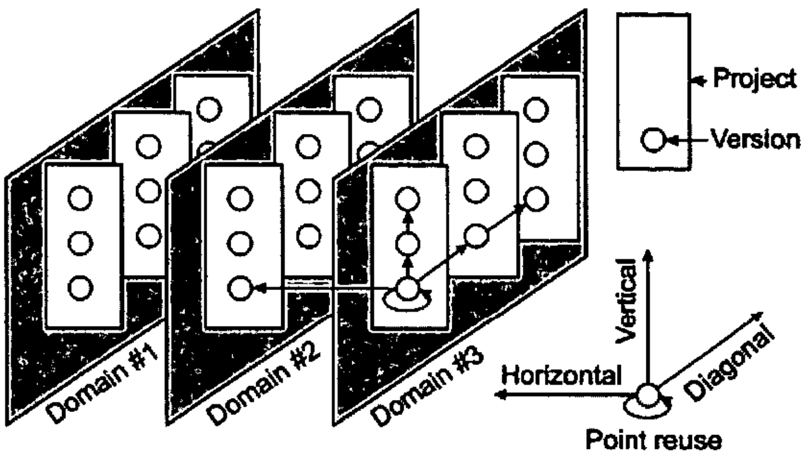


Fig. 3.13 Reuse directions. (Kovacs et. al., 1999)

The reuse types based on the reusable components found in each phase of software system design are as follows (Ambler, 1997):

- **Domain Component Reuse** – Domain component reuse refers to the identification and development of large-scale, reusable business components. Domain components provide the greatest potential for reuse because they represent large-scale, cohesive bundles of business behaviours that are common to many applications.
- **Pattern Reuse** – Pattern reuse refers to the use of publicly documented approaches to solve common problems. Pattern reuse provides a high level of reuse that you can implement across multiple languages and platforms. The disadvantage of pattern reuse is that patterns do not provide an immediate solution since code must still be written to implement the pattern.
- **Artefact Reuse** – Artefact reuse refers to the use of previously created development artefacts, e.g. use cases, standards documents, domain-specific models, procedures and guidelines, and other applications that can kick start a new project. Artefact reuse promotes consistency between projects and reduces the project management burden for each new one.
- **Framework reuse** – Framework reuse refers to the use of collections of classes that implement the basic functionality of a common technical or business domain. OO frameworks provide an important enabling technology for reusing both the

architecture and the functionality of software components. The disadvantage of framework reuse is the complexity of frameworks makes them difficult to master, requiring a lengthy learning process on the part of developers.

- **Component reuse** – Component reuse refers to the use of pre-built, fully encapsulated components in the development of an application. Component reuse offers a greater scope of reusability than either code or inheritance reuse. The main disadvantage of component reuse is that because components are small and encapsulate only one concept, a large library is required.
- **Template reuse** – Template reuse is typically a form of documentation reuse. The main advantage of documentation templates is that they increase the consistency and quality of development artefacts. The main disadvantage is that developers have a tendency to modify templates for their own use and not share their changes with co-workers.
- **Inheritance Reuse** – Inheritance reuse refers to using inheritance in an application to take advantage of behaviour implemented in existing classes. The advantage of inheritance reuse is that as the behaviour of the class has already been developed, it decreases both the development time and cost of the application. Unfortunately, the misuse of inheritance will often result in developers missing an opportunity to reuse components, which would have offered a much higher level of reuse. Secondly, novice developers will often skimp on inheritance regression testing (the running of super class test cases on a subclass), resulting in a fragile class hierarchy that is difficult to maintain and enhance.
- **Code reuse** – The main advantage of code reuse is that it reduces the amount of actual source code needed, potentially decreasing both development and maintenance costs. The disadvantages are that its scope of effect is limited to programming and it often increases the coupling within an application.

The following computer based systems support design reuse (Costa, 2000):

- DEKLARE (Design Knowledge Acquisition and Re-Design Environment), where a framework for supporting adaptive/variant design of mechanical artefacts is defined. Functional, physical and process design models are used through constraint management (Fothergill, Forster et al. 1995; Fothergill, Arana et al. 1996). However, DEKLARE can be characterised as a process based approach that is driven by a mapping of constraint definitions, and is applied to more simple and defined product structure;

- RODEO (Reuse of Design Objects), where suitable redesigned objects are retrieved from a design database to fulfill a given requirement specification. Thus the current design problem is solved by instantiation or by (small) adaptations. A feature-based model describes design objects and requirement specifications by their properties (Altmeyer, Schuermann et al. 1994; Altmeyer, Schuermann et al. 1994) However, the reuse of designed objects is focused on a CAD framework;
- DESPERATO (DESign Process Encoding & Retrieval by Agent Designated Operations), where reuse in highly innovative and creative design tasks is investigated through the use of a computer-based indexing system. Intelligent agents that conduct encoding and retrieval operations on the user's behalf, have an interface with an object oriented database (Ormerod, Mariani et al. 1997);
- DEDAL, where an intelligent guide for browsing multimedia design documents supports reuse of engineering experience. A model for retrieval and indexing of multimedia design information is used (Baya, Gevins et al. 1992). However, the reuse in DEDAL is applied to management aspects of the design documents rather than to the product design itself;
- NODES (Numerical and object based modelling system for conceptual engineering DESign), where knowledge of design solution objects and their numerical relations are modelled. NODES is an interactive modelling system, which supports building, manipulation and analysis of a model of the design artefact and provides information feedback on the model. The knowledge is obtained by accumulating solutions of problems defined within that domain (Duffy, Persidis et al. 1996).

Conventional manufacturing systems are considered to be too rigid. Sensors and artificial intelligence enable development of flexible systems capable of making decisions under conditions of uncertainty. The applications of more powerful computers have allowed the implementation of more advanced manufacturing concepts. The systematic knowledge of industrial manufacturing has created an environment that facilitates the introduction of artificial intelligence. New developments include the knowledge-based factory, holonic manufacturing (Toh, et al, 1999) and the society of agents (Rzeveski, 1997).

3.6 Potentially Reusable Information and Knowledge in Manufacturing

Information and knowledge are becoming strategic resources that are as important as more traditional resources such as raw material, energy and food (Kalpic, Bernus, 2002). Therefore, information and communication technologies can be considered today as strategic technologies, and knowledge is considered as the key capital of enterprises (Kalpic, Bernus, 2002). It is therefore very important that good use is made of these valuable strategic resources and that efforts are made to reuse information and knowledge wherever possible.

3.6.1 Need for Systematic Reuse in Manufacturing Systems

Continuous improvement in manufacturing companies often produces large amounts of information. Many of the previous sections described IT tools that enable information and knowledge to be shared either because they made use of models, which can be viewed by different people, or because the software actually actively encourages people to work together on projects, for example workflow systems. However sharing is only one simple form of reuse, and many other different types and levels of reuse exist.

System designers often meet the problem of creating new components of an application that someone probably previously has already produced. Without having effective reuse tools, usually it is more natural to create new components from scratch than to seek for useful elements in other programs and/or systems.

Working with reuse generally means that previously designed and/or implemented elements are used again (and again) in later projects with or without changes (Kovacs et. al., 1999). The effectiveness of reuse is increased when during the design and implementation phase of new (software) components it is kept in mind that these elements may be reused later on. Creating reusable components for a repository is an activity *for reuse* while the exploitation of elements from the repository is called analysis, design, making code *with reuse.*

Reuse is not a new strategy; however, the current emphasis on systematizing reuse and focusing such efforts toward narrow, well-defined application areas, or product lines is novel. This restricted focus enables reuse of large-scale components, such as software architectures and entire subsystems, that best leverage the knowledge, expertise and resources within an organization, thereby enabling greater future payoff after initial investment in software reuse.

Several studies by the European Software Institute have proven the positive effects of software reuse for software making enterprises. The quality of the software increased together with the decrease of software making costs, i.e., the productivity becomes higher and the time-to-market is shorter. At the same time flexibility increases when reuse is applied. Naturally there is an initial investment which has to pay off within a rather short period of time. As initial investment the new style of producing new software modules, i.e., the 'design for reuse', and the utilisation of reusable software, the 'design with reuse' have to be introduced on one hand, and an appropriate reuse repository has to be purchased or developed on the other hand. Initially software making costs will increase but as soon as the results can be harvested a real cost decrease and quality increase will be achieved. According to the European Software Institute, (1995a, 1995b) the additional cost of producing reusable software is about 20%, while the cost reduction by reuse is about 40% on average. The same study states that 2–5 uses (reuses) result in a payback of the investment.

The most effective reuse programs concentrate on the identification and development of a small, high-quality set of needed and useful components. In fact, as the amount of a new product made from existing components increases, improvements in costs, time, and quality will be achieved. Rine (2000), suggests that the experiences, processes, and workproducts should always be recorded for learning, and the workproducts should be designed to facilitate reuse [STARS 95].

3.6.2 Reusable Components in manufacturing systems

The potential for reuse is enormous since the majority of each new application could be assembled from reusable components if the appropriate components could be predetermined and built prior to system development. At all levels of development from requirements specifications to code, there are components that by the nature of implementing tasks and representing information on a computer must appear over and over again in an applications. Some of these similarities can be predefined and built into software tools; others can be created as reusable components which are stored in reuse libraries.

Reuse of software elements is becoming more and more important in the life-cycle of software products. However, it does not necessarily mean that only the code of a software product can be reused. One view is that reuse efforts should focus on code (Frakes & Gandel, 1990) another opinion is that all the results and resources used in a project, including human expertise, should be reused (Basili & Caldierra, 1988). There are several elements defined during the analysis, design and implementation of a simulation model, as ideas, concepts, object classes, and lines of source code created, etc., that should be reused in new applications (Kovas et. al., 1999). All the documents created during the life-cycle of a product, as ideas, methodologies, requirement specifications, design results, documentation, etc., could be reused in later projects. Application of reuse methodology and practice will reduce the effort in development of new systems.

The basic components of different FMS and FMC (Flexible Manufacturing Cell) are the same type of machine tools, robots, transfer equipment, etc. In most relevant aspects they usually only differ from each other in their quantity and working parameters. This fact itself breeds the idea of reuse of FMS and FMC elements.

The reuse of FMS components for new modelling applications has been demonstrated using the Systematic Asset Library Management System (SALMS) tool that has been demonstrated by Kovas et. al.(1999). SALMS is a central library whose retrieval facilities provide support in finding matches for all kinds of reusable assets stored in the company's software repository (Reusable Software Asset Library, RSAL), and all kinds of technical documents available in the company (Technical Library, TL).

A reusable enterprise model called the factory data model (FDM), has been developed by Yu et. al. (2000) as a part of Factory Design Process (FDP) Research Project funded by EPSRC under grant GR/L09301 (Harding et al., 1999). This model supports a multi-view approach to enterprise design. The Product Range Model (PRM) is another example of successful implementation of software reuse in manufacturing (Young, Costa).

Feature modelling has been found to be an effective method for identifying and modelling reusable objects. The feature-oriented reuse method, FORM, is extended by Lee et. al. (2000) for improving the object-oriented engineering of applications for reuse. FORM concentrates on analysing and modelling commonalities and differences in the applications of a given domain in terms of capability, operating environment, domain technology, and implementation technique features.

The analysis and design of a multifunctional virtual model of FMS has been done by Kovas et al (1999) using extended Booch methodology. It contains a generic manufacturing component library consisting of templates of manufacturing elements to serve for interactive and flexible FMS model design and configuration. The generic representation of an FMS is given in the form of a class hierarchy with top classes presenting abstract levels of manufacturing elements and with bottom classes presenting concrete manufacturing components. This representation of an FMS was developed reusing the representation of FMSs from previous projects. The reused elements contain the basic FMS concepts and the task sharing among different functional parts of the FMS model. The user of the system can build up new FMS models creating instances from the library of the ready-to-use classes by cloning them and filling out their attributes. It is also possible to create new elements by refining classes from higher abstraction levels.

According to Kovacs et. al. (1999), it is advantageous to abstract common FMS concepts and to create their representation in an abstract way. For instance, instead of designing milling, drilling, turning, and other manufacturing operations one by one, it is time-and labour-saving to design the manufacturing operation as such.

Chapter 4
THEORY BUILDING

4.1 Introduction

Chapter 3 of the thesis examined literature covering manufacturing system modelling and design and various general aspects of knowledge management and reuse. Chapter 4 continues the literature review and explores the background to the research problem in more detail. The identified research gap (from section 2.2.2) is explored with the help of previous work done in the area of information and software reuse, and particular attention is paid to practical examples and reported case studies. Analysis of the secondary data contained within the reported case study examples shows that similarities exist in the reported factors that influence the successful introduction of reuse projects and increase the ongoing levels of successful reuse. In addition, there are also identifiable similarities in factors that are detrimental to the success of reuse programs. These are termed inhibitors and they are also considered in this chapter.

Identification of these critical success factors and inhibitors leads directly to the value net driven solution proposed by this research. This is explained fully in section 4.3 (the Theory Building Stage). The proposed Reuse Value Net enables understanding of both the current status of reuse within an enterprise and ways in which it can be improved and increased. However, understanding ***what*** exists is only part of the requirement of a good reuse programme; an understanding of ***how*** to advance and improve reuse is also required. This research therefore proposes that process patterns be used in combination with the Reuse Value Net, to satisfy this second requirement. Hence this chapter also provides a review and discussion of process patterns in section 4.4.

4.2 Secondary data analysis

The purpose of this section is to establish a background for the present research and to show the need and value of this research. This section concentrates on the experiences that can be gained from reported examples and case studies of reuse programmes.

4.2.1 Literature review

The information reported in this section largely relates to reuse of software and software components. This is because systematic and intentional reuse of components in software design has a longer history and consequently far greater publication than manufacturing knowledge reuse. This research has therefore actively sought to learn from software reuse experiences wherever possible. Systematic reuse will not just "happen", it needs to be managed and become an intrinsic part of the software processes that an organization follows (Griss, 1993). The idea of systematic reuse, i.e. planned development and widespread use of software components was first proposed in 1968 by Doug Mcllroy. Since then, many attempts have been made to improve the software process by reusing software components, and these have achieved varying degrees of success.

At first glance, it might appear that all an organization has to do to implement a reuse process, is to collect well-tested software components in a library, and encourage software developers to use these components, rather than to write entirely new ones (Griss, 1993). Many companies are developing proprietary software libraries, but software reuse is not yet a major force in their corporate software development. Nevertheless, it appears that product development costs, factoring in the cost of producing, supporting, and integrating reusable software components, can decrease by a sustainable 10 to 12 percent. Quality can also improve as defect rates in delivered products can drop drastically to 10 percent of their former levels; and long-term maintenance costs can drop to 20 to 50 percent of their former values when several products share the same, high-quality components (Griss, 1993). In fact, as the amount of new product made from existing components (the "reuse level") increases, corresponding improvements in costs, time, and quality have been observed. At reuse levels of 80 percent or more, these improvements are dramatic. But, it takes more than the existence of a parts library alone to achieve these results (Griss, 1993).

As explained in section 2.2.2 (Research gap) simply announcing the existence of a library will not achieve the necessary paradigm and behaviour changes. Persuasion, training, incentives and many other management and organisational changes may be required. When confronted with their first reuse failure, and suddenly realizing the magnitude of the changes needed to make reuse succeed, organizations need to pursue an incremental improvement process, rather than to abandon the effort. Each reuse organization will encounter varying opportunities and issues peculiar to its situation. For a reuse program to be effective, the specific inhibitors likely to affect it must be identified and overcome in a timely way.

Several good books, tutorials, and review articles summarize the status of reuse practice and research. Each includes some discussion of the technical, managerial, and organisational implications, as well as important case studies (Poulin, 1997, McClure, 1997, Frakes et. al, 1991, Joos, 1994).

4.2.2 Case study data analysis

The reuse experiences described below illustrate the need to keep several factors in perspective simultaneously. Focus has typically been on technical factors, leaving aside organizational, domain, economic, and cultural issues. However, the experiences of the reported case studies show that the key to success is the integration of all these factors into well-defined reuse programs.

4.2.2.1 Reuse Success Stories

Case 1. Raytheon (Prieto-Diaz, 1991b)

A successful reuse story that has often been quoted is Raytheon's Missile Systems Division, Information Processing Systems Organization. They observed that 60% of all business application designs and code were redundant and could be standardized and reused. A reuse program was established to analyze their existing software and to exploit reuse. Over 5000 production COBOL source programs were examined and classified. Three major module classes were identified: edit, update, and report. They also discovered that most business applications fall into one of three logic structures or design templates. These logic structures were standardized and a library was created to make all classified components available for reuse. Several modules were also redesigned to fit the standard logic structures. New applications are slight variations of the standard logic structures and are built by assembling modules from the library. Programmers are trained to use the library and in the new method of recognizing when a logic structure can be reused to build a new application. Reuse is compulsory. This program has been in operation for over six years. They reported an average of 60% reused code in their new systems and a net 50% increase in productivity.

Case 2. Fujitsu. (Prieto-Diaz, 1991b)

Fujitsu's Software Development for Electronic Switching Systems (SDESS) Program has taken a simpler and more pragmatic approach to reuse. They analyzed previous Electronic Switching Systems (ESSs), catalogued and documented each in their Information Support Center (ISC). The ISC is a regular library staffed with systems analysts, software engineers, reuse experts, and ESS domain experts. Library staff roles include reference desk, cross reference support (between software and its original designers), software archives, and commercial catalogues. It is compulsory for all software projects to formally include the ISC in their development cycle. This is accomplished by including ISC staff members in all design and software reviews. Fujitsu reports a significant improvement in software development measured in percent of current projects being on time. Before the SDEES program only 20% of 300 projects were on schedule. With SDEES that number increased to 70%.

Case 3. GTE Data Services (GTEDS). (Prieto-Diaz, 1991b)

GTE Data Services' Asset Management Program (AMP) is a corporate wide program created to "develop a reuse culture" in the organization (Prieto-Diaz, 1991b). Their approach to reuse is a mix of the previous approaches with some novel features. GTEDS began analysing their existing systems to identify and select reusable assets. Assets are any software work product (designs, documents, code) that can be partially or totally reused in a new application. The collection was classified and cataloged into an automated library system. Five specialized teams were then created to maintain the library, promote reuse, and support reusers. Instead of making reuse a compulsory practice, GTEDS established a program of incentives and rewards. Programmers were paid $50 to $100 cash for each component accepted into the library and royalties were paid to program authors each time their components were reused in new projects. A "reuser of the month" award was established for those who reused the most, and the managers of projects that achieved a certain reuse level were given bonuses of budget extensions. GTEDS reported 14% reuse in the first year of the AMP. Considering GTEDS software production volume (about 700 programmers) this translates into $1.5 million in savings. Their projected figures for 1995 were 50% reuse, with an estimated $10 million in savings.

Case 4. Motorola. (Joos, 1994)

Reuse was initiated when the technical advisors requested the CEO to include software reuse in the company's goals. A Reuse Task Force was formed. The force members were experts from different departments. The reuse task force considered the need and requirements for reuse organisational issues, education and motivation, methods, technology, and implementation. The reuse task force recommended two reuse approaches: *design recovery or opportunistic reuse* and *designing reusable components or systematic reuse*. The reuse task force proposed a reuse working group to provide the development and subsequent communication of reuse guidelines and standards to software engineers and their management. A Reuse Working Group was formed with 15 engineers. The reuse working group and the CEO recommended that Motorola should implement reuse at the software-engineering level, and this was subsequently done. The reuse working group and the reuse task force determined that education was the most important activity. The working group arranged seminars to introduce basics of reuse, workshops to teach development and use of reusable components and held reuse conferences. Senior management accepted the responsibility for initiating reuse. With the help of the internal and external software experts, the senior executive program committee drafted a list of action items to be addressed by Motorola. The top management encouraged reuse throughout the organisation by a cash-reward incentive program.

Case 5: HP corporate reuse program. (Griss, 1993)

Hewlett-Packard (HP) has been engaged in software reuse since the early 1980s. Early work involved the development of instrument libraries in BASIC, the construction and use of databases to store and distribute software components, and more recently the use of Objective-C or C+ + to develop class libraries. Several of these libraries have been widely distributed within the company, and some were even distributed outside the company. Today there are many active reuse projects in HP divisions and in HP laboratories. Several HP divisions have developed and reported on more ambitious reuse programs involving common architectures, components, and libraries for families of related products in a variety of application domains. These sometimes spanned several divisions, and have included the following product areas: embedded software for instruments and peripherals; network management; and analytical, medical, and manufacturing systems.

Following an extensive survey in 1989 and 1990 of these ongoing division-sponsored reuse programs and a study of reuse at other companies, a corporate reuse program was created, aimed at making software reuse a more significant and systematic part of the software process at HP. In establishing this program, it was felt that a fairly broad, well coordinated software reuse program involving management, process, and technology was needed to make significant progress.

The program's goal was to develop, qualify, and promulgate the best practices that could be effective within HP. The program involved a core team of software reuse experts, with additional people being assigned as required on several divisional pilot projects. The core team worked on developing the following: reuse process, domain analysis, reuse assessments, economic models, coding guidelines, a reuse handbook, reuse education, and consulted on divisional reuse projects. The pilot projects combined the evaluation and refinement of proposed best practices for reuse, and the incremental introduction of new methods into divisional reuse efforts.

Case 6: Unknown SME (Biggs, 2004)

Biggs (2004) has reported software reuse efforts at a small computer manufacturer, who have a single product in public access and security domain. Their system keeps information on all the people that are currently present at a particular location and can issue and check security badges.

A software reuse program was started with the help of Durham university, in order to make the software processes more structured.

In the first step, management support was gained by giving a presentation on reuse, explaining how it could help in their company and how to best utilise it. Suggestions were then made on how to set up a reuse program in the company, along with the costs that would be associated with the setting up of this program, and a consideration of the risks involved.

Working knowledge of the company and its current working practices was gathered. The company's development process, and the viewpoint of the staff were investigated by conducting informal interviews of certain members of staff. Based on the results of the interviews, a strategy for adopting software reuse technique was suggested. An incremental approach to implementation of the reuse strategy was stressed as an incremental approach can be tried and proved on a small scale before introducing major changes in the company.

A pilot project was selected. One of the problems faced was that, in case of the tight deadlines, the development of the new system would accelerate with less regard for the reuse recommendations made. With less pressure the recommendations were reviewed, and the code written was reconsidered in order to see if it could be made more reusable. However, it was seen that the emphasis on reuse was giving the developers motivation to spend more time planning their code in advance and also go back to the code once written and restructure it in order to make it more object oriented and reusable. Following the pilot project a similar system was developed. The results showed a total reuse factor of 70% for the whole project.

4.2.3 Analysis of Reuse Literature and Case Studies

The Case Study data discussed in the previous section reports the only substantial reuse case studies that have been identified in a through literature review. As Christensen & Ron (2000) comment, studies of reuse from a practical point of view are scarce. However, the reported results and example reuse projects discussed above exhibit learning similarities in both problem areas and critical success factors within reuse projects.

4.2.3.1 The reuse inhibitors

Problems encountered in the case studies included factors that reduced or even prevented reuse from taking place. These factors are termed "inhibitors" and there are many inhibitors to starting and running an effective reuse program (Griss, 1993). The potential for reusability to improve the software development process is immense, yet a variety of factors affect its impact (Lewis, et al., 1992). These influential factors are both technical and non-technical in nature. Technical factors include development paradigm issues and tools for searching and classifying reusable components. The majority of work-performed in the area of software reuse has focussed on technical factors while non-technical issues, particularity human factors, are often ignored or dismissed.

In many reported cases, developers and users of potentially reusable components have been unnecessarily hampered because they lacked specific knowledge concerning the factors, which influence software reusability.

4.2.3.2 Factors that affected reuse

The reported examples confirm the following points:

- Many practitioners initially believed that reuse required large libraries. They therefore started their reuse programs by creating classification structures for software components, and collecting as much software (good or bad) as they could. Incentives were offered to contributors. Complex library systems were built to store, manage, and find the components. Researchers have studied alternative classification schemes, prototyped powerful browsers, and automated access systems.

- Libraries of randomly collected code generally have not been designed together, and usually only address a small number of a typical developer's needs. Components may work individually, but not work well together. Developers are reluctant to go looking for components and prefer to write their own, usually claiming (without even checking) that the available parts are wrong, inefficient, or have many defects. Incentives need to be focused on the desired goal. For example, a common incentive is to offer a "reward" for contributing to a library but not for using the library, which can easily increase library size without increasing reuse (or the library's usefulness). This creates a major information problem. People need to know what exists and how to use it (Griss, 1993).

- The software reuse at GTE has shown that without strong management, the initiatives cannot be sustained on a broader level. Issues of process and management of reuse must be defined, and questions answered such as: How can the entire cross-project software life cycle be defined whilst also trying to meet a given set of reuse goals? (Griffin, 1995).

- In almost every case of sustained reuse acceptance, a management champion, or "corporate angel," believed in the strategic importance of systematic reuse and fought for long-term commitment and investment. Individual champions and management support in general is an enabler, convincing participants to take risks, try new approaches, and trust software written by others. Management commitment to reuse can overcome many obstacles, both technical and non-technical.

- In particular, the three most crucial elements leading to a successful reuse programme are management leadership and support, organisational change, and the creation of a reuse mindset. These, plus a small, high-quality library, are the things to invest in.

- Reuse programs change the software development process. Programmers have to check what can be reused before coding and designers have to check previous designs. It is also strongly suggested that customers should identify their requirements to take advantage of existing designs. (This clearly may have implications for reuse of any type of manufacturing design knowledge).

- In the experiences of the different organisations it is clear that in order to achieve an efficient reuse process, many organizational, management and technical issues had to be addressed. Some of these factors were anticipated, whilst many others were discovered along the way.

- The lessons learnt by the organizations in the reuse program highlight not only the success factors but the inhibitors as well. It is important to identify these risks early, and plan to handle them systematically. Once the domain inhibitors have been identified, solutions can be proposed and tried. It is therefore important to have knowledge of all the factors in the reuse process from an- early stage of the reuse process.

4.2.4 Conclusion from the literature review

From the above discussion the following conclusions can be drawn:

1. For the reuse process to be successful the technical as well as non-technical problems should be addressed. It is therefore important to identify and study all the factors involved in the reuse process simultaneously.

2. It is important to identify risks early, to plan to handle them systematically, and to propose and try solutions. It is therefore important to have knowledge of all the factors in the reuse process from an early stage. This can be achieved by studying the relationships and interdependencies between different reuse factors.

3. From reuse experiences such as those discussed in this chapter, it is apparent that implementing a reuse discipline entails more than creating and using reuse libraries. People need to know what exists and how to use it (Griss, 93). The practice of reuse should be formalised by including support for reuse in software development methods, tools, training, incentives, and measurements. Unless reuse is made explicit and formalized, an organization will not be able to repeatedly exploit reuse opportunities in multiple projects. Without a reuse-driven development process to guide them, system developers will not achieve sufficient sharing and reuse.

4.2.5 The Reuse Factors

The following are examples of obstacles and possible resolutions that have been found in the case studies and reuse examples.

Management. Management commitment is essential because reuse programs demand changes in the way software is developed. Management must provide the necessary company resources required to start, evolve, and operate a reuse program and also make available necessary key information for assessing the reuse potential of the organization. Without the long-term support of senior management and their willingness to make up-front investments, most reuse programs cannot succeed. Such commitment allows projects to work together and balances the short-term needs of individual projects with the longer-term needs of the product portfolio.

The solution may be to pick a suite of projects that already have a supportive senior manager. Other members of management that are affected by the programme, must be persuaded of its importance, and of appropriate time frame expectations, and of the need for commitment to the programme. This might be achieved by providing them with case studies, cost-benefit analysis, or return-on-investment calculations. Contacts with senior managers who have applied successful reuse programs in their own organizations may also influence the general management.

Culture. People may not know how to make use of reuse effectively, or may be biased against it through a lack of trust, a “not-invented-here,’ syndrome, or fear of loss of creativity and independence.

The solution may be to introduce incentives, training, and management backing. Publicize success stories. Try several alternative methods of eradicating the “not-invented-here” syndrome. Build confidence in libraries and support teams.

Organization. Various kinds of institutional barriers make it hard to change financial policies, contracting models, and other legal policies. Evaluation policies such as rewarding individual work and productivity more highly than group work, or complex rules for exchanging and cross-charging for software between divisions or groups can be significant impediments.

The solution may be to create and empower a corporate-wide body (or several, linked, sector- or group-wide bodies) to advocate reuse and make it succeed by changing the reward and funding mechanisms. Establish groups to define and support reusable work products.

Economics. The funding profile for reuse projects is quite different from conventional software projects. Typically, several years of up-front investment are needed before payoff is realized. Managers are reluctant to make this long-term investment without some guarantee of success.

The solution may be to develop return-on-investment (ROI) models. Describe success stories. Treat reusable work products as assets, requiring appropriate design, maintenance, and enhancement.

Legal issues. In some situations, incorrect contracting mechanisms actively discourage reuse. The lack of contracting mechanisms makes it hard to create agreements that can be trusted or enforced. Increasing the use of third-party software increases the importance of this issue.

The solution may be to develop (new) contracts, maintenance agreements, and royalty systems. Negotiate for rights to contracted components.

Technical aspects. Arbitrary software work products are typically not very reusable and are often hard to find.

The solution may be to provide guidelines and standards for building, testing, and documenting reusable work products, together with an enforcement mechanism. Sometimes a library with a classification structure, retrieval mechanisms, and certification procedures may be required. Architectural guidelines, documented frameworks, and reuse reviews can help ensure that components are designed to fit. Introduce reuse-oriented inspections to ensure quality and correct component usage.

4.3 The theory building stage

In today's business context enterprises are under continuous competitive pressure in the market place. However, all businesses cannot be considered to be competitors, there are businesses that complement too. There are also products or services, which provide complementary rather than competing products and services. The Complementors are those who cooperate to capture market share (Nalebuff, Brandenburger, 1996). There are four entities affecting a business that should be considered when developing a business strategy i.e., the suppliers, customers, competitors and the complementors. Nalebuff and Brandenburger introduced a schematic map of a business called 'The Value Net', to examine the relationships between these four entities.

4.3.1 The Value Net

The concept of a value net is based on Game Theory, where business is considered as a game. The Value Net is a map of the game of business. The Value Net describes all the players and analyses the elements of competition and co-operation between them. In other words, the value net locates all the various players relative to one another and identifies the interdependencies between them. The concept of the value net is applied here to the reuse process, to support the simultaneous consideration of the multiple factors that either promote or inhibit reuse.

In its simplest state, a value net is visually represented by a diamond, with a company in the centre. The value net can be graphically portrayed as the interactions between four players, i.e. customers, suppliers, competitors and complementors and the company. In the Value Net, along the vertical dimension are the company's *customers* and *suppliers*. Resources such as raw materials and labour flow from the suppliers to the company, and products and services flow from the company to its customers. Along the horizontal dimension of the Value Net are the company's *competitors* and *complementors*. Complementors are third parties who can lend added value to a firm by supplementing its products and services. For example, a computer manufacturer views a software company as a complementor, because customers place greater value on a computer with an operating system installed than on either the computer or the software alone. The Value Net is a high level view of the key relationships that drive any company's ability to succeed sustainably (Rabbino, 98). Fig. 4.1 shows a schematic map of the whole game or The Value Net, as presented by Nalebuff and Brandenburger, (1996).

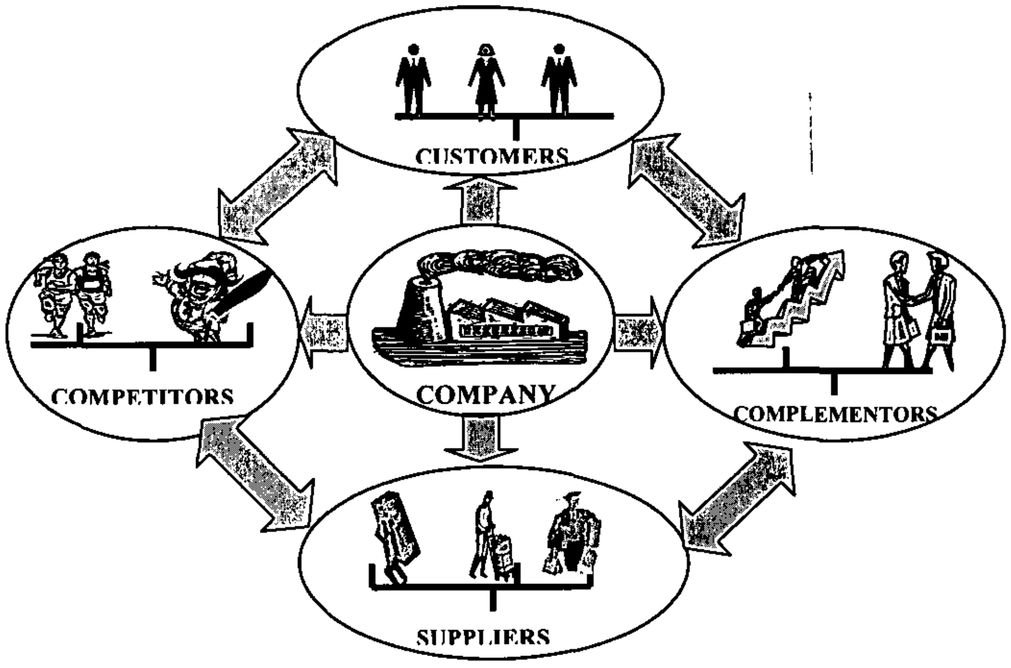


Fig. 4.1 The Value Net (adopted from Nalebuff, Brandenburger, 1996)

4.3.2 Applying value net theory to the reuse driven design process

The value net above is initially drawn from the company's point of view. But the concept extends beyond this single viewpoint, as there are customers' customers, suppliers' suppliers, and so on. The value net can either be extended to represent these extended values or separate value nets can be drawn for suppliers, customers, etc. In the context of the present research different reuse factors must be grouped because of their important interdependencies. The roles of the players, customers, suppliers, complementors and competitors will now be examined, to form a reuse value net.

4.3.2.1 The suppliers and customers

In the Value Net suppliers and the customers appear on the vertical dimension, as the customers play symmetric roles with the suppliers. They are equal partners in creating value. Experts and Designers fulfil these roles in the Reuse Value Net. Reuse is about the production and consumption of reuse (Fafchamps, 1994). A producer is a creator of reusable work products, and the consumer is someone who uses them to create other software (Lim, 1994). In the Reuse Value Net, system experts are the 'suppliers' who 'supply' or provide their expert knowledge for reuse. Management also plays a role as a supplier as it provides the funding, resources and additional support environments that are required to enable the reuse process. Designers have 2 roles, both as a 'supplier' and as a 'customer'. The designers may design a component, which is subsequently available for reuse. Therefore, in this context, a designer is an expert, who uses his skills to produce a reusable product (hence designers are suppliers). A designer may also reuse a component designed by another designer or take advantage of expert knowledge on reuse (hence designers are also customers). The Organisation is a customer as it wishes to gain the financial and operational benefits that can result from efficient reuse.

The value net shown in figure 4.2 summarises this range of customers and suppliers.

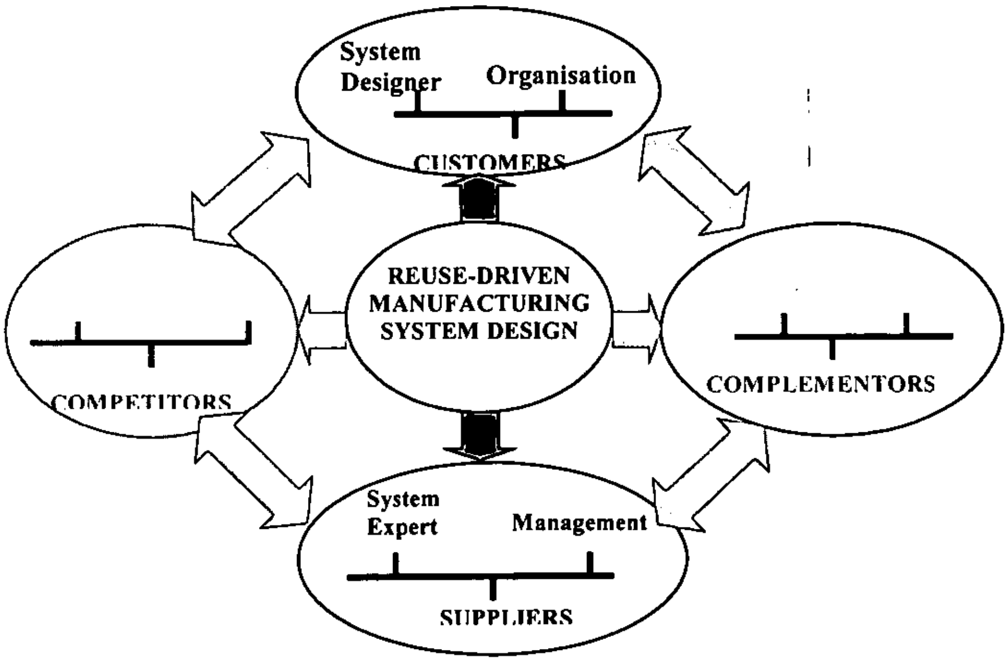


Fig. 4.2 The suppliers and customers in the Reuse Value Net

4.3.2.2 The competitors and complementors

On the horizontal dimension of the value net, there is another symmetry, i.e. the competitors and the complementors. *A competitor provides the competition to a product and to meet and overcome this competition the quality of the product must be constantly improved* (Nalebuff, Brandenburger, 1996). The traditional design methods, which do not promote reuse, are the competitors to the reuse-driven process. One of the reasons for the designers using design processes, which do not involve reuse, is the Not-Invented-Here (NIH) Syndrome (Ambler, 1998). These designers prefer the traditional design methods, which do not promote reuse, and hence are the competitors to the reuse-driven process. Hence, NIH competes against reuse. On the other hand, the members of the organization will be better motivated to practice reuse when there are reuse libraries/catalogues where they can easily find the reusable components. Hence the reuse libraries are the complementors in the Reuse Value Net. However, it is not sufficient to have something to reuse, since designers will not reuse existing components if they feel it is quicker or easier to "design from scratch", than to determine whether an existing component is appropriate for their application. The reuse libraries should therefore be complemented by the knowledge of how to reuse the available components, and by techniques to simplify the reuse of these components, see figure 4.3.

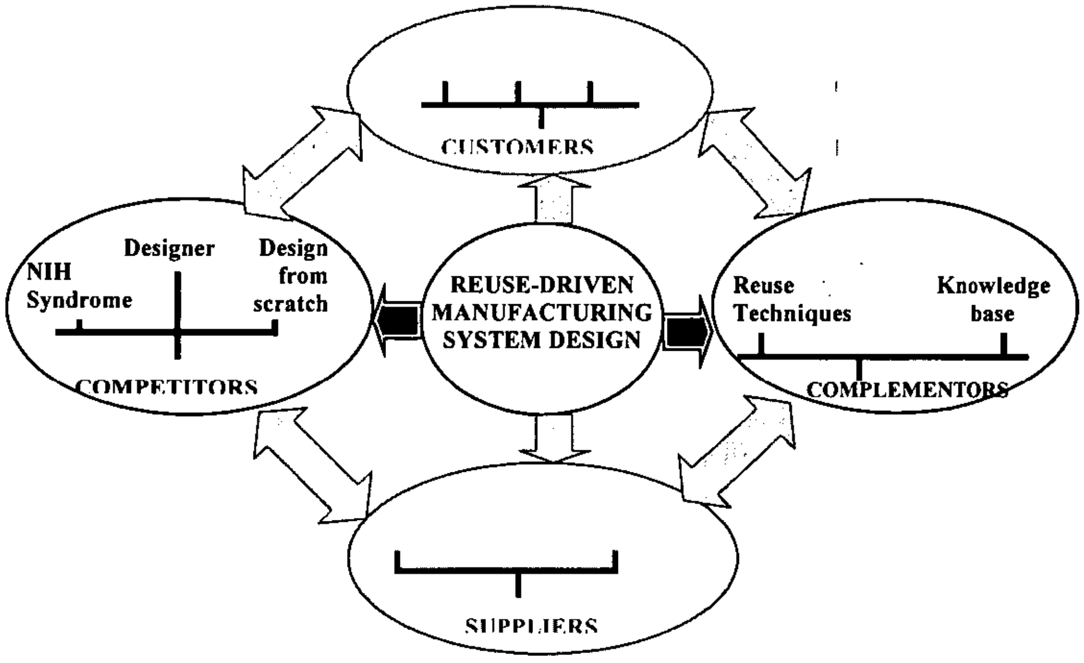


Fig. 4.3 The complementors and competitors in the Reuse Value Net

4.3.2.3 The Reuse Value Net

The previous discussions and figures show that the factors affecting reuse can be mapped using a Reuse Value Net that is based on the concept of a value net as described in section 4.3.1. The completed Value Net for Reuse-Driven Manufacturing System Design can be drawn as shown in figure 4.4.

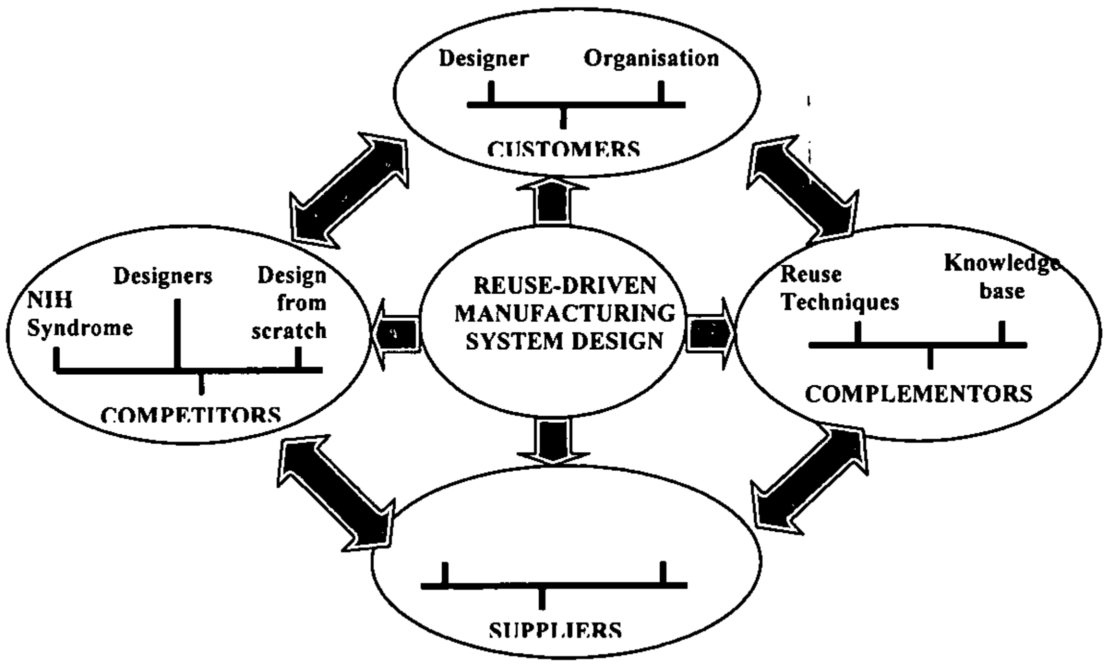


Fig. 4.4 Reuse Value Net

The Reuse Value Net in figure 4.4 shows that experts, designers and management are the *suppliers* of knowledge and funding to the reuse process. Similarly, the designers and organization are the *customers* as the designers are reusing the expert knowledge and the organization is getting benefit from savings that are made as a result of the reuse process. The competitors to the reuse process are alternative design methods, such as design from scratch, and preconceptions or existing prejudices of personnel involved in the process, such as NIH syndrome as this thinking restricts the designer, preventing him from taking advantage of reusable elements. Finally, the reuse library, knowledge bases, and various reuse tools and techniques are complementors as the reuse library provides the reusable components. The knowledge bases give the knowledge of what to reuse and the reuse techniques provide the knowledge of how to both reuse existing components and extend the library as existing components are adapted or new components are created.

The above explanation, describes the relationships between the four players participating in the reuse process. However, these players should not only be examined individually, as they are related, e.g., the designers (potential customers) may be adversely influenced by Not-Invented-Syndrome (competitor). To win this customer from the competitor and convert him over to reuse, training (which is a complementor) can be given to the designers about the advantages of the reuse process; for example, the designers can be educated about how to reuse the reusable components. Designers will also be better motivated if a reuse library (complementor) is well organised, and the components are easy to find and reuse, and if it is straightforward to determine which reusable components meet the requirements of a particular specification. Furthermore, management can create competition among the designers, or motivate the designers initially by rewarding their efforts, both in making use of reusable components and in making new components easier to reuse. The relationships of each player with other players in the value net need to be studied in detail. This information can then be used to find the points of leverage. The points of leverage may be different for different companies. The reuse value net provides all the necessary information of the relevant players and their inter-relationships to implement a reuse process, and hence can be used as a good analysis tool.

4.4 Introduction of the process patterns.

Implementing reuse entails more than creating and using libraries of reusable assets. It requires formalizing the practice of reuse by integrating reuse processes and related reuse deliverables into system life cycle processes (McClure, 1997). The reuse value net explained above identifies the "what", i.e. what should be done to achieve efficient reuse and the "who", i.e. the factors and actors involved in the reuse process. However, to achieve efficient reuse, the "when" and "how" of practicing reuse must also be known. In the current research, process patterns have been identified as a means to answer these questions. The process patterns are used to guide the designer about how the reuse process works. Process patterns identified by Scott Ambler

(1998) provide a valuable insight into the reuse process. The guidelines suggested by McClure Carma for incorporating reuse in the software development process have been used to identify the different factors believed to be involved in the reuse process generally, so that they may be applied to manufacturing knowledge reuse.

Defining patterns is difficult, because they are not bound by prescriptive formal definitions. Rather, process patterns are validated when and if there is consensus about the existence of particular patterns in a range of existing software.

James Coplien (1996) put it this way:
I could tell you how to make a dress by specifying the route of a scissors through a piece of cloth in terms of angles and lengths of cut. Or, I could give you a pattern. Reading the specification, you would have no idea what was being built or if you had built the right thing when you were finished. The pattern foreshadows the product: it is the rule for making the thing, but it is also, in many respects, the thing itself.

An alternative way of describing a process pattern is:- "A pattern is both a process and a thing; both a description of a thing which is alive, and a description of the process which will generate that thing" (Alexander, 1979). In other words, a pattern does more than just showcase a good system's characteristics; it teaches us how to build such systems (Winn, Calder, 2002).

A process is defined as a series of actions in which one or more inputs are used to produce one or more outputs. Defining a pattern is a little more difficult. Alexander (1979) hints at the definition of a pattern by pointing out that the same broad features keep recurring over and over again, although in their detailed appearance these broad features are never the same. Alexander shows that although every building is unique, each may be created by following a collection of general patterns. In other words, a pattern is a general solution to a common problem or issue, one from which a specific solution may be derived.

Process Patterns history

Patterns provide a mechanism for rendering design advice in a reference format (Fowler, 2003). According to Fowler, there are two vital components: the how and the when. The how part is obvious— how do you implement the pattern? The when part often gets lost.
"One of the most useful things I do when trying to understanding a pattern, one I'm either writing or reading, is ask, "When would I not use this pattern?" Design is all about choices and tradeoffs; consequently, there usually isn't one design approach that's always the right one to use. Any pattern should discuss alternatives and when to use them rather than the pattern you're considering. "(Fowler, 2003).

According to Fowler (2003) patterns are half-baked. This means that they always have to be personalised or specialised to adapt them for use in a particular environment. Indeed, the experience of implementing a pattern for a particular purpose is one of the best ways to learn about it. Process patterns are also useful for training purposes, since, an expert in a team can use written patterns to help educate other team members (Fowler, 2003).

Coplien (1995), in his paper "A Generative Development-Process Pattern Language," hints at a definition for the term "process pattern" in his statement that "the patterns of activity within an organization (and hence within its project) are called a process."

A *process pattern* is a pattern which describes a proven, successful approach and/or series of actions for developing software. An important feature of a process pattern is that it describes what you should do but not the exact details of how you should do it (Ambler, 1998). The scope of a single process pattern may range from a high-level view of how applications are developed to a more-detailed view of a specific portion of the software process. Ambler (1998), defined process patterns and classified them in three types, as follow:-

> **Process pattern.** A process pattern is the patterns of activity within an organization (and hence within its project) are called a process.

Task process patterns. This type of process pattern depicts the detailed steps to perform a specific task.

Stage process patterns. This type of process pattern depicts the steps, which are often performed iteratively, of a single project stage. A project stage is a higher-level form of process pattern, one that is often composed of several task process patterns.

Phase process patterns. This type of process pattern depicts the interactions between the stage process patterns for a single project phase, such as the Initiate and Delivery phases. Phase process patterns are performed in serial order, made up of stage process patterns which are performed iteratively.

Process patterns can also be considered as templates, and according to Mahemoff and Johnston (1998), the key attributes of a pattern are:-

- Name: A name to identify the pattern.
- Context: The situation(s) where the pattern is relevant.
- Forces: Any existing forces that may constrain or suggest alternative solutions. When these forces are in tension with one another, the problem is harder to solve and a compromise may be necessary.
- Solution: A solution which resolves, as far as possible, the various forces. The solution is really what the pattern is, yet the problem is a vital part of the context.

The patterns concept suggests itself as a logical way of looking at tasks, as well as imparting information about how they may be supported (Mahemoff and Johnston, 1998).

Based on the above definitions and characteristics, process patterns are considered to be a useful method for planning and directing the necessary steps for advancing a reuse process that has been specialised to be appropriate to the needs of a particular enterprise. Process patterns, have therefore been chosen as the method for communicating requirements and activities that could be recommended by the reuse support tool, based on the results of the reuse value net analysis.

4.5 Summary.

Through the range of reviewed case studies and examples of reuse, a set of critical success factors and inhibitors to reuse have been identified. This chapter has also shown that a combination of value net and process pattern approaches can provide the necessary range of functionality to initiate, plan, implement and promote reuse processes. A Reuse Value Net has also been presented and described.

Chapter 5

DESIGN AND IMPLEMENTATION

5.1 Introduction

This chapter describes the experiment conducted to explore the requirements and design of the prototype support tool and environment. The proposed reuse support tool includes a specialist application called the Reuse Agent, which can,

1. acquire knowledge of the current level of reuse practised in the company;
2. analyse the current status of reuse and generate a relevant Reuse Value Net, personalised for the company;
3. recommend methods of increasing and promoting the implementation of reuse within the company, through the use of appropriate process patterns.

The representation of the Reuse Agent and the extent to which it can provide support to reuse-driven manufacturing system design processes are also considered. Section 5.2 presents the aims and objectives of the experimental software environment, and section 5.3 presents an overview of the experimental set-up. A detailed design has been made of the experimental environment, and this has been documented using Unified Modelling Language (UML) diagrams. UML is one of the popular object oriented analysis and design methods which were reviewed in section 3.2.2. The UML based design of the experimental environment is presented in section 5.4.

5.2 Aims of the implementation work

The implementation has the following aims:

1. To demonstrate that the reuse value net can be generated for a particular organisation and that it can identify the problem areas for reuse implementation and possible solutions. The following activities were carried out to achieve this aim:-
   - i to design a questionnaire to find out the current state of the reuse thinking in the organisation.
   - ii to design a data collector for collecting the questionnaire results.

iii

iv to generate an experimental reuse agent knowledge base to analyse the data collected through the questionnaire.

2. To show that the process patterns can guide the designers through the process of reuse implementation. The following activities were carried out to achieve this aim:-

   i to design a process pattern catalogue

   ii to design a process pattern library

5.3 The Experimental Set Up

A prototype implementation of a reuse support tool has been produced as part of this research, in order to test the proposed concepts and solutions. The experimental set up is shown in figure 5.1.

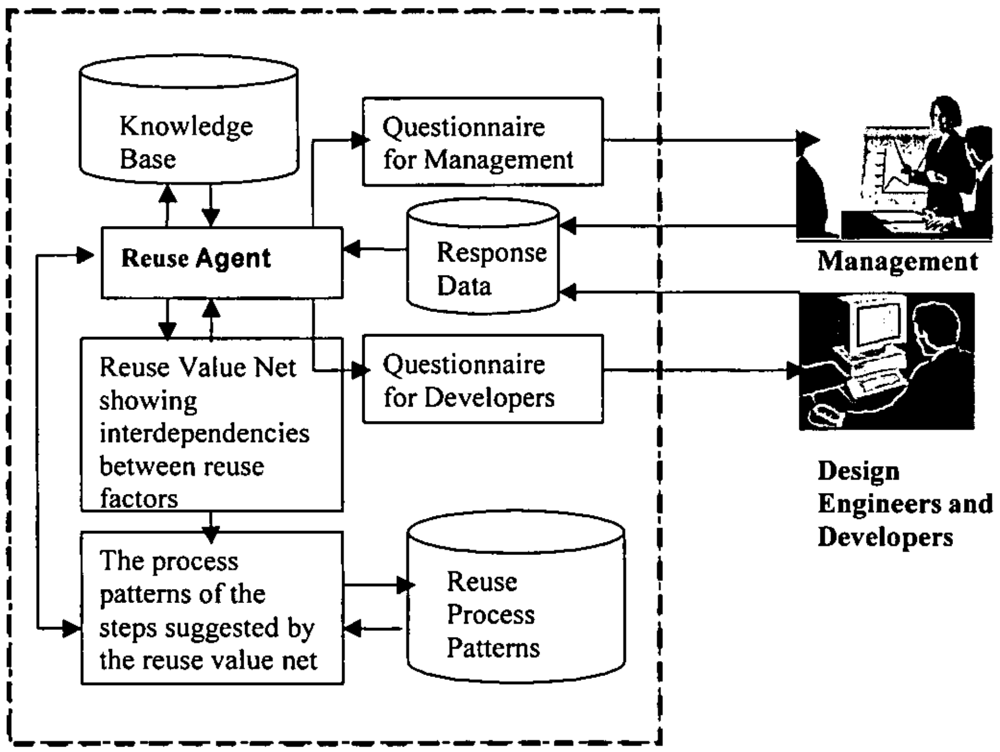

Knowledge Base
Questionnaire for Management
Reuse Agent
Response Data
Reuse Value Net showing interdependencies between reuse factors
Questionnaire for Developers
The process patterns of the steps suggested by the reuse value net
Reuse Process Patterns
Management
Design Engineers and Developers

Fig. 5.1 Experimental set up

The Reuse Agent performs the following important tasks:

- ***Sends questionnaires to the members of management and designers and collects data.*** The Reuse Agent is used to determine appropriate activities during the Reuse Project. Questionnaires are sent to members of management and design / modelling teams, to establish whether any reuse activities currently exist in the company. The completed questionnaires are collected and the response data stored in the database. The questionnaires are designed to also identify whether individual design engineers are reusing components informally; this may be by using elements of models they have previously built, or even working informally with colleagues and sharing components.
- ***Analyses response data and generates a Reuse Value Net for the particular company.*** The Reuse Agent processes the response data, and determines the problem areas that initially need to be addressed. As explained, the Reuse Agent initially uses its knowledge, which is stored in the ReuseKnowledgeBase, to analyse the questionnaire response data and produce a Reuse Value Net.
- ***Search for appropriate Reuse Process Pattern in the Reuse Process Pattern Catalogue and Library.*** To support its recommendations, the Reuse Agent then consults the Reuse Process Patterns database (shown in figure 5.1) and suggests activities that need to be undertaken to progress the reuse project. Once again the Reuse Agent can guide users through the activities necessary to provide solutions to these identified problems by indicating useful steps for developing necessary resources. The process pattern steps suggested by the Reuse Agent are generic, and need specialising to suit the needs of the particular company.

5.4 The Implementation Environment

5.4.1 The experimental software development environment

The prototype software tool has been developed using JBuilder5 and JDataStore database. Rational Rose 98 has been used to design the experimental environment. Rational Rose supports object oriented analysis and design methods and enables users to model their software in several different modelling languages, including UML. Figure 5.2 shows the logical view of the packages in the software application. The global package Jbuilder is used in all the packages whereas the Database package is linked to the PeopleInfo, ReuseSupport and OrgArtifacts packages to store information. Figure 5.3 shows the class structure of the Reuse Agent, and the operations in the class show all the tasks performed by the Reuse Agent.

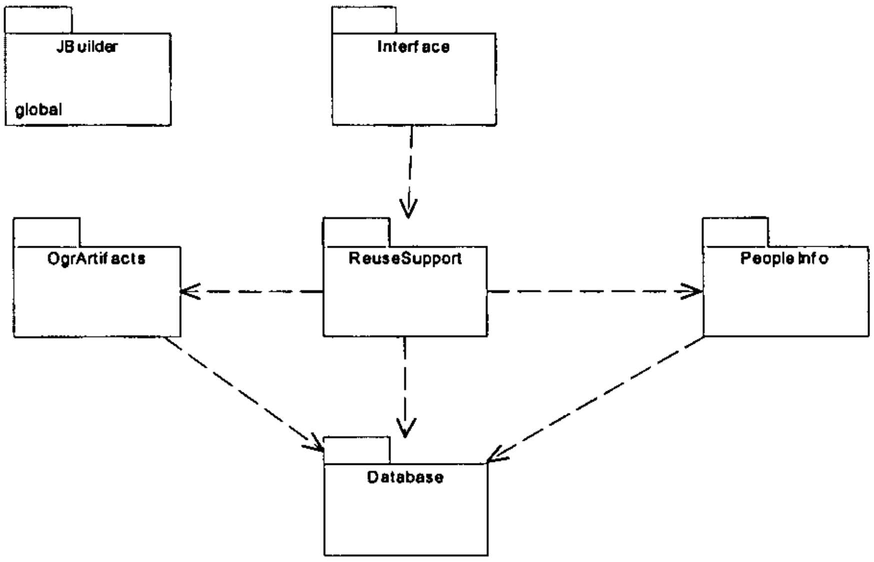
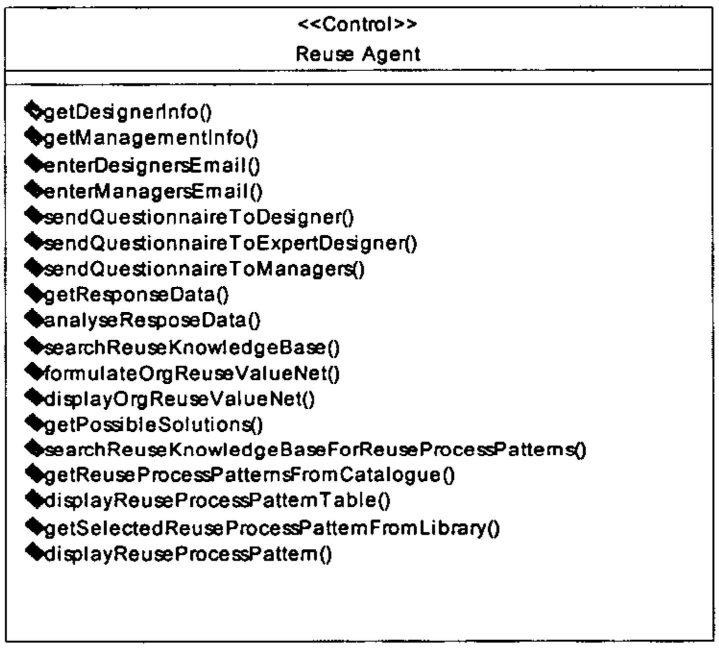


Fig. 5.2 Experimental Set up -UML Class Diagram (Logical View)


<<Control>>
Reuse Agent

```
◆getDesignerInfo()
◆getManagementInfo()
◆enterDesignersEmail()
◆enterManagersEmail()
◆sendQuestionnaireToDesigner()
◆sendQuestionnaireToExpertDesigner()
◆sendQuestionnaireToManagers()
◆getResponseData()
◆analyseResposeData()
◆searchReuseKnowledgeBase()
◆formulateOrgReuseValueNet()
◆displayOrgReuseValueNet()
◆getPossibleSolutions()
◆searchReuseKnowledgeBaseForReuseProcessPatterns()
◆getReuseProcessPatternsFromCatalogue()
◆displayReuseProcessPatternTable()
◆getSelectedReuseProcessPatternFromLibrary()
◆displayReuseProcessPattern()
```

Fig. 5.3 The ReuseAgent UML Class Structure

5.4.2 Identification of Current Reuse Status

In order to acquire knowledge of the current level of reuse practised with a company, the Reuse Agents sends out questionnaires to the system designers, experts and managers who are members of the reuse project team, and collects their response data in a database. This interaction of the Reuse Agent with the other actors is shown by the Fig. 5.4 Send Questionnaire UML collaboration diagram. Figure 5.5 shows the UML class diagram of Interface package.

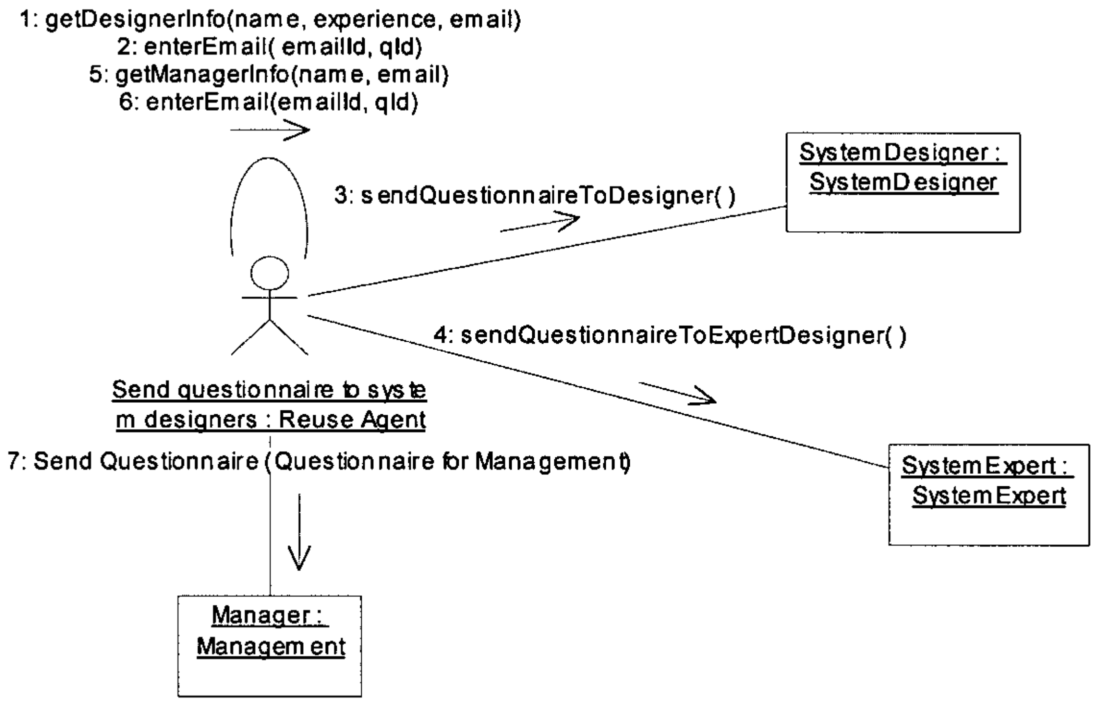

1: getDesignerInfo(name, experience, email)
2: enterEmail(emailId, qId)
5: getManagerInfo(name, email)
6: enterEmail(emailId, qId)
3: sendQuestionnaireToDesigner( )
SystemDesigner : SystemDesigner
4: sendQuestionnaireToExpertDesigner( )
Send questionnaire to system designers : Reuse Agent
7: Send Questionnaire (Questionnaire for Management)
SystemExpert : SystemExpert
Manager : Management

Fig. 5.4 Send Questionnaire collaboration diagram

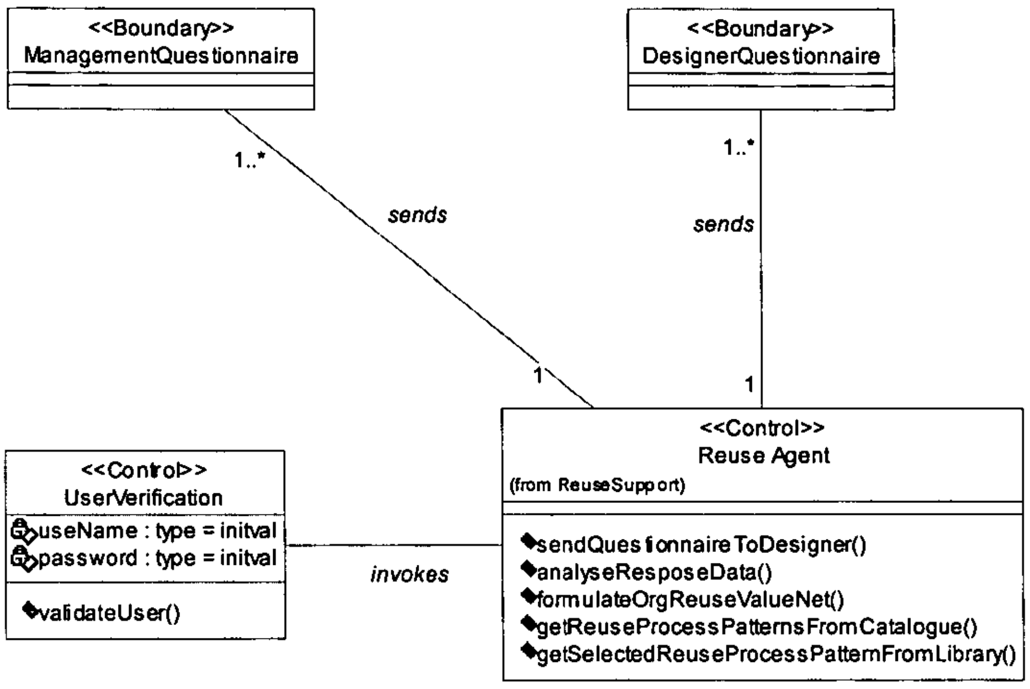


Fig. 5.5 Interface package -Class Diagram(Logical View)

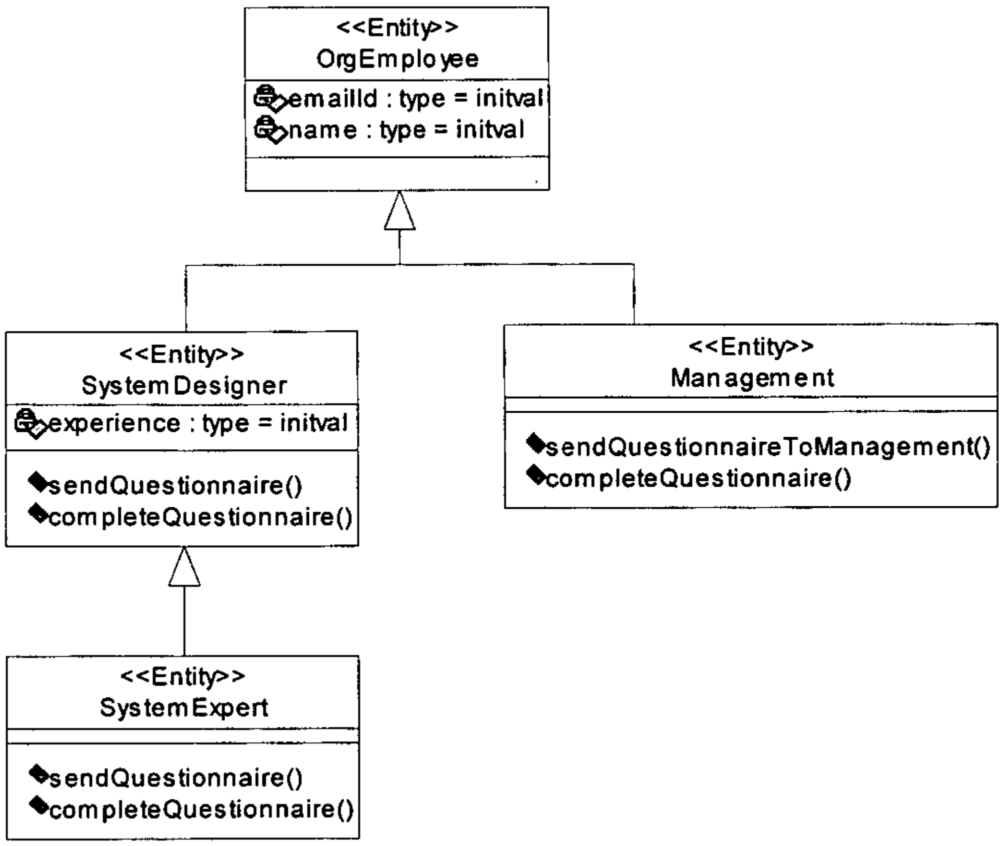


Fig. 5.6 PeopleInfo package -Class Diagram(Logical View)

Figure 5.6 shows UML class diagram of the package PeopleInfo. Class OrgEmployee has two attributes, name and emailId of the employees. The child classes Management and System designers classes are the child classes of class OrgEmployee. The attributes are name of the employee and the email address. The questionnaire sent by the ReuseAgent to the SystemDesigners and the Management class are completed and sent back. This operation is performed by the completeQuestionnaire() operation.

5.4.2.1 Questionnaire design

The questionnaire design is an important step in the development of the Reuse Agent as the solutions suggested by the reuse agent are based on the data collected through the questionnaire. The questionnaire design was influenced by the questionnaires in the literature reviewed (McClure, 1997).

The questionnaire uses closed questions, to collect important information about the system development process in the company. The questionnaire tries to look at the reuse thinking in the organisation, designer's and management's knowledge about reuse etc. The information is collected individually from the management and from the designers as their perspectives and knowledge of current reuse practises within the organisation may well differ. Hence, as figure 5.10 shows, the questionnaire has been split into two sections:-

- Questionnaire for management and
- Questionnaire for designers.

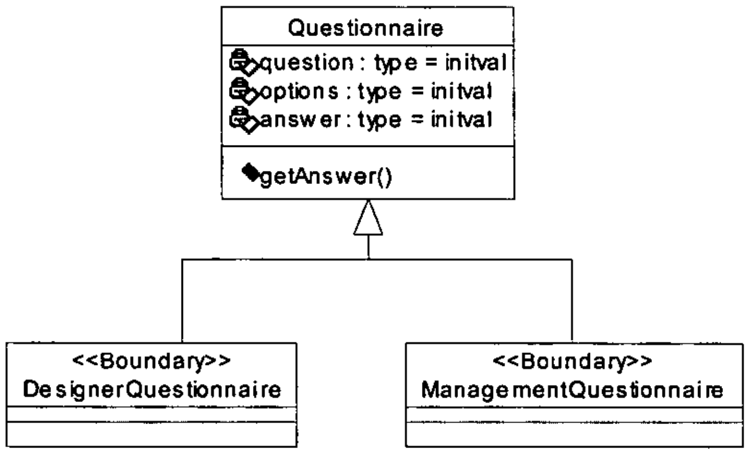


Fig. 5. 7 Reuse Agent Class Diagram/Interface Attributes and Operations

Each of the two questionnaires is organised in different sections. The common sections of the questionnaires are as follow:

- ***Management and funding.*** This section addressed the issues of management support and funding. Literature and past experiences shows that the management support is one of the most important factors for reuse implementation. It is therefore important to know if the management is ready to fully support the reuse implementation. The other factor, which is as important as the management commitment, is the funding needed to implement reuse program.
- ***Organisational culture.*** This section addresses the current practices in the organisation, mainly related to reuse. It addresses issues such as, the reuse resources (such as supporting tools, reuse repository etc.), motivation etc.

Apart from the two sections mentioned above the questionnaire for designers includes:

- ***System design information.*** The aim of this section is to find out about the system development practice in the organisation. It questions whether designers are practicing reuse on an individual level, and whether any (possibly informal) knowledge or component sharing already exists amongst the designers. This section of the questionnaire tries to address issues such as Not-Invented-Syndrome.

5.4.2.2 The Response Data

The answers from the Management and Designers questionnaires are stored in the Response Data database. This is shown in Fig. 5.8 Collect Response Data – UML collaboration diagram. Fig 5.9 shows the structure of the ResponseData class

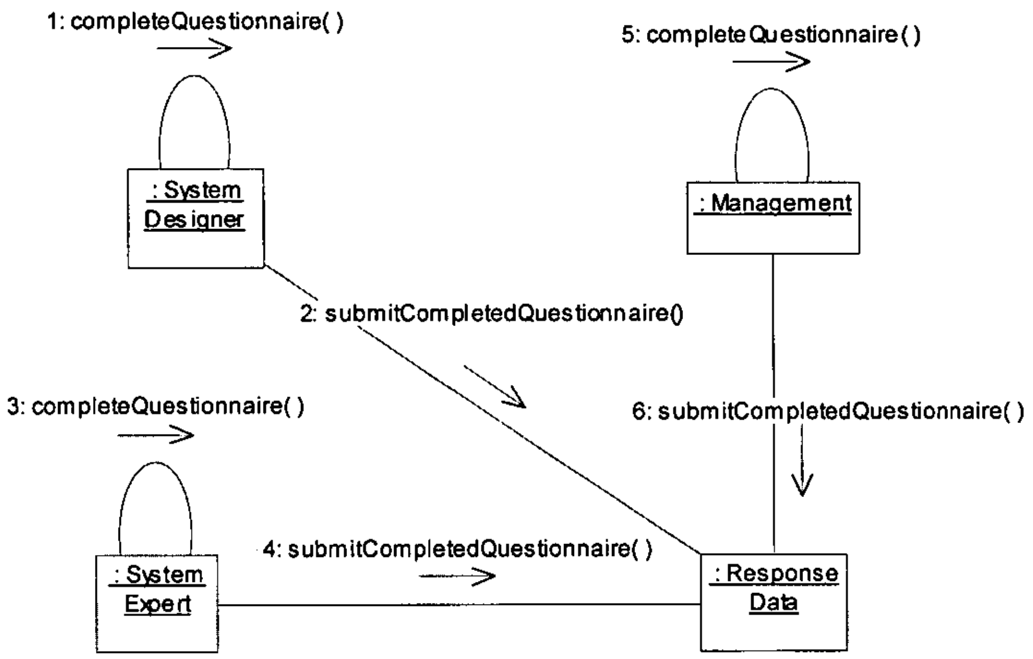

1: completeQuestionnaire( )
5: completeQuestionnaire( )
: System Designer
: Management
2: submitCompletedQuestionnaire()
3: completeQuestionnaire( )
6: submitCompletedQuestionnaire( )
4: submitCompletedQuestionnaire( )
: System Expert
: Response Data

Fig. 5.8 Collect Response Data UML collaboration diagram

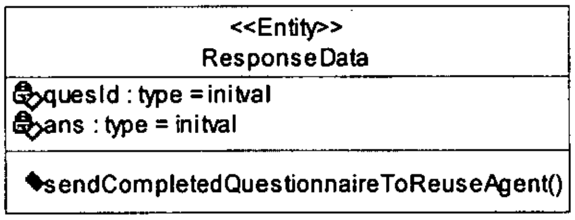


Fig. 5.9 Structure of the ResponseData Class

5.4.3 Analyses response data and generates Reuse Value Net for the particular company.

The Response Data is then accessed and processed by the Reuse Agent to identify the current status of reuse and any conflicts in the company in order to formulate the Reuse Value Net. Fig 5.10 shows this interaction.

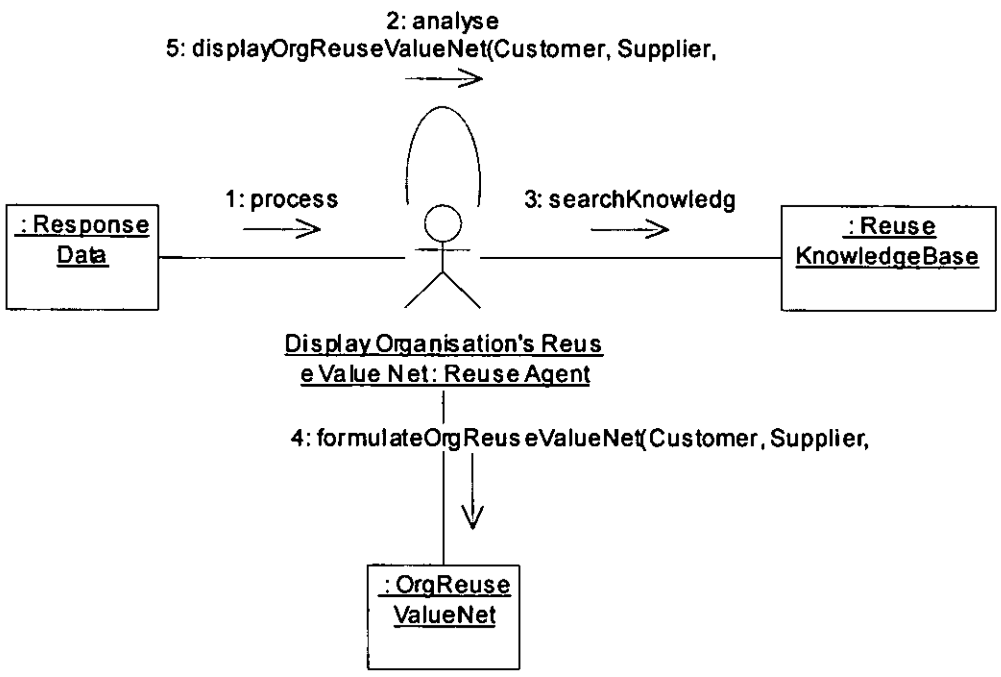

2: analyse
5: displayOrgReuseValueNet(Customer, Supplier,
1: process
3: searchKnowledg
: Response Data
: Reuse KnowledgeBase
Display Organisation's Reuse Value Net: Reuse Agent
4: formulateOrgReuseValueNet(Customer, Supplier,
: OrgReuse ValueNet

Fig. 5. 10 Analyse Response Data UML collaboration diagram

5.4.3.1 Reuse Knowledge Base

After the data has been collected, in order to analyse the data, the reuse agent needs knowledge to apply as the basis for analysing the data and arriving at a conclusion about the reuse problem(s) identified in the organisation. The knowledge required includes understanding which factors are complementors and which are competitors to reuse, their relationships and inter-relationships, and how each factor is likely to inhibit or promote reuse. In the prototype version of the Reuse Agent, the required knowledge has been acquired from the secondary case studies that were reviewed in chapter 4. The structure and content of the Reuse Knowledge Base are shown in the UML object classes presented in figure 5.11.

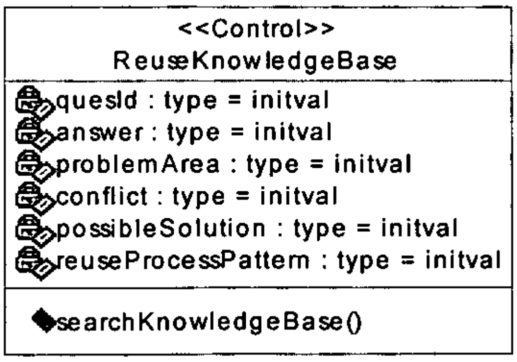

<<Control>>
ReuseKnowledgeBase
quesId : type = initval
answer : type = initval
problemArea : type = initval
conflict : type = initval
possibleSolution : type = initval
reuseProcessPattern : type = initval
searchKnowledgeBase()

Fig. 5. 11 Structure of the ReuseKnowledgeBase Class

The ReuseKnowledgeBase class has the following attributes:

- **quesId:** This field contains the question id for each question in the questionnaire.
- **answer:** This field contains a list of different options for a particular question in the Questionnaire. The option selected is the Primary_Key as each answer is unique.
- **problemArea:** This field contains a list of category of the player from the reuse value net that is resisting reuse implementation. It identified a broad area where the problem lies.
- **conflict:** This field contains a list of actual problem identified from the answer opted by the designer/member of management team.
- **possibleSolution:** This field contains a list of the possible solutions for the problem identified through the questionnaire analysis.
- **reuseProcessPattern:** This field contains a list of process patterns related to the possible solution identified through the questionnaire analysis.

5.4.3.2 Organisations Reuse Value Net

The response data collected by the Reuse Agent through the questionnaires are analysed. From the analysis of the data the Reuse Agent identifies problems for reuse implementation in the company and the possible solutions. All this information is then stored in the OrgReuseValueNet class along with the reuse players that exist in the company. The formulateOrgReuseValueNet() method draws the Organisation's or company's Reuse Value Net.

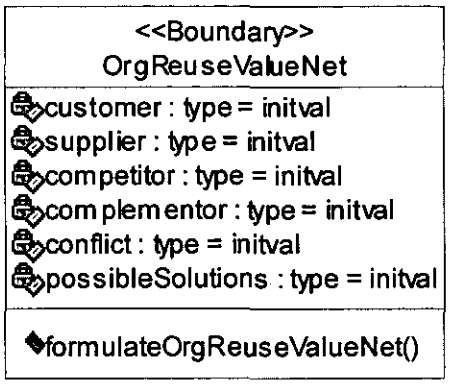

<<Boundary>>
OrgReuseValueNet
customer : type = initval
supplier : type = initval
competitor : type = initval
complementor : type = initval
conflict : type = initval
possibleSolutions : type = initval
formulateOrgReuseValueNet()

Fig. 5. 12 Structure of the OrgReuseValueNet Class

5.4.4 Search Reuse Process Pattern in the Reuse Process Pattern Catalogue and Library.

Using the organisation's reuse value net, the reuse agent identifies the problems in the current reuse implementation. The possible solutions to these problems are given in the form of reuse process patterns. The reuse agent searches for appropriate process patterns in the process pattern library and catalogue and then displays them. This interaction is shown in fig. 5.13.

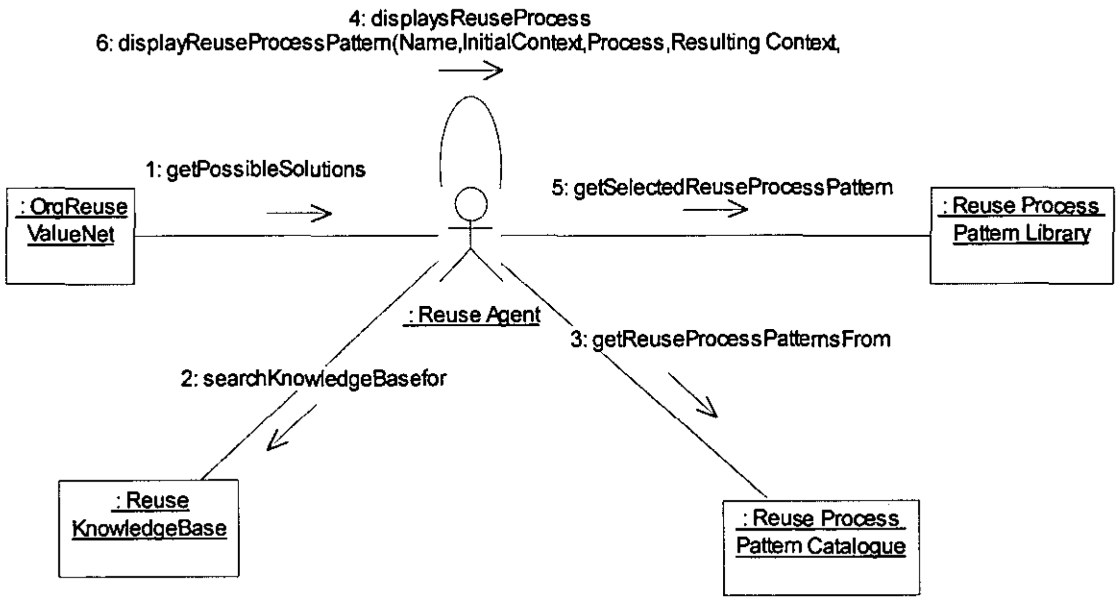
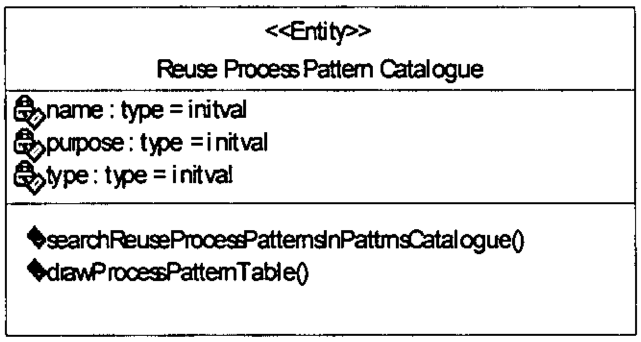

4: displaysReuseProcess
6: displayReuseProcessPattern(Name,InitialContext,Process,Resulting Context,
1: getPossibleSolutions
5: getSelectedReuseProcessPattern
: OrgReuse ValueNet
: Reuse Process Pattern Library
: Reuse Agent
2: searchKnowledgeBasefor
3: getReuseProcessPatternsFrom
: Reuse KnowledgeBase
: Reuse Process Pattern Catalogue

Fig. 5. 13 Find Reuse Process Pattern collaboration diagram

5.4.4.1 The reuse process pattern catalogue

The reuse process pattern catalogue is used to store the information about the reuse process pattern and not the actual reuse process pattern. The purpose of this catalogue is to provide the purpose of different reuse process patterns at a glance, in order to simplify searching for the appropriate patterns.



<<Entity>>
Reuse Process Pattern Catalogue
name : type = initval
purpose : type = initval
type : type = initval
searchReuseProcessPatternsInPattrnsCatalogue()
drawProcessPatternTable()

Fig. 5. 14 Structure of the ReuseProcessPatternCatalogue Class

The main attributes in the catalogue are:

- **name** : This field contains the name of the process pattern. This is the primary_key in the table i.e., the name of the pattern is unique.
- **purpose** : This field contains the purpose of the pattern.
- **type:** This field tells whether a pattern is Phase, Stage or Task process pattern.

The searchReuseProcessPatternCatalogue() method searches the catalogue when a pattern name or purpose is entered. The second method drawProcessPatternTable() method draws a table to display the results of the search.

5.4.4.2 The Reuse Process Pattern Library

The Reuse Process Pattern Library is where the actual Reuse Process Patterns are stored. Figure 5.15 shows the structure of the ReuseProcessPatternLibrary. The searchReuseProcessPatternLibrary() method searches the catalogue when a pattern name or purpose is entered. The second method drawProcessPatternTemplate() method draws a template to display the results of the search.

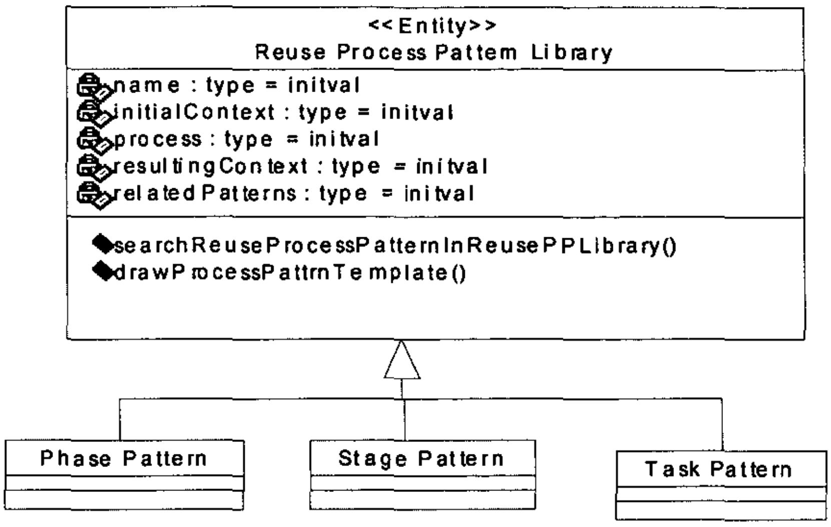

<<Entity>>
Reuse Process Pattern Library
name : type = initval
initialContext : type = initval
process : type = initval
resultingContext : type = initval
relatedPatterns : type = initval
searchReuseProcessPatternInReusePPLibrary()
drawProcessPattrnTemplate()
Phase Pattern
Stage Pattern
Task Pattern

Fig. 5. 15 Structure of the ReuseProcessPatternLibrary

5.5 Summary

A summary of the tasks performed by the Reuse Agent (as described in sections 5.4.2 and 5.4.3) is provided by figures 5.16 and 5.17.

This chapter has explained the design and implementation of the experimental environment. The next chapter explores the implementation in detail through a case study example.

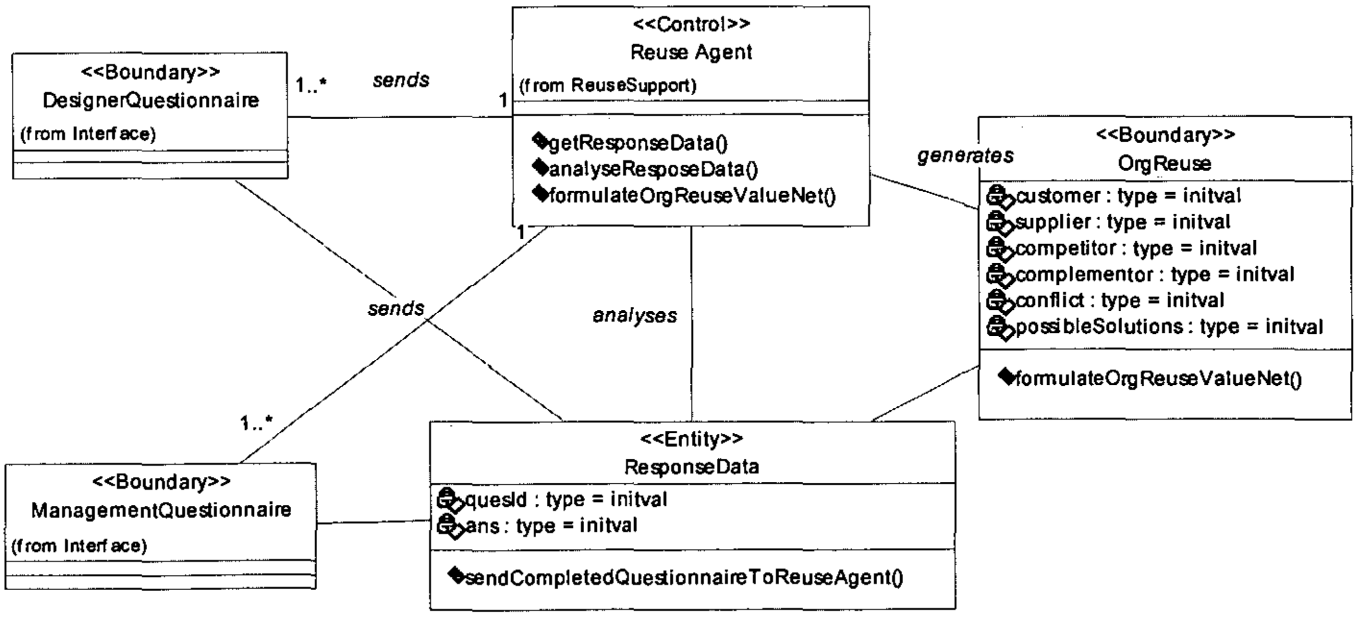
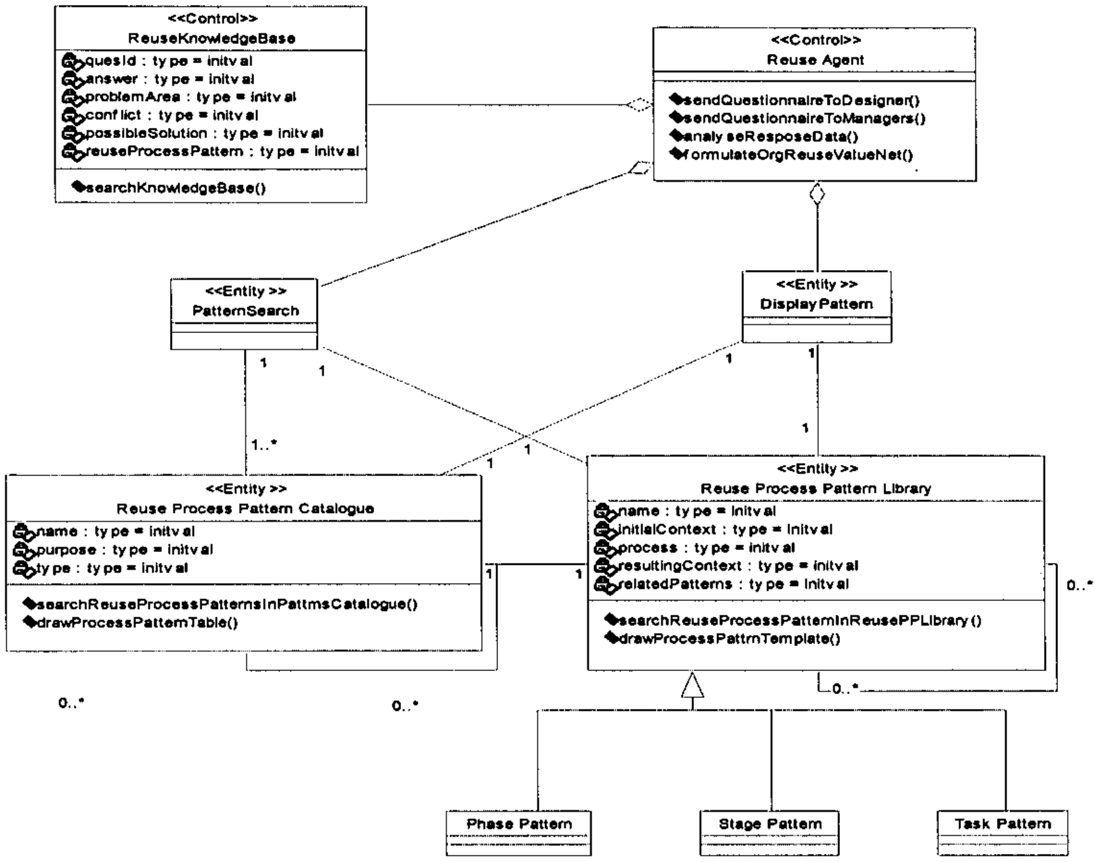


Fig. 5.16 OrgArtifacts package -Class Diagram(Logical View)



<<Control>>
ReuseKnowledgeBase
quesId : type = initval
answer : type = initval
problemArea : type = initval
conflict : type = initval
possibleSolution : type = initval
reuseProcessPattern : type = initval
searchKnowledgeBase()
<<Control>>
Reuse Agent
sendQuestionnaireToDesigner()
sendQuestionnaireToManagers()
analyseResposeData()
formulateOrgReuseValueNet()
<<Entity >>
PatternSearch
<<Entity >>
DisplayPattern
<<Entity >>
Reuse Process Pattern Catalogue
name : type = initval
purpose : type = initval
type : type = initval
searchReuseProcessPatternsInPattmsCatalogue()
drawProcessPatternTable()
<<Entity >>
Reuse Process Pattern Library
name : type = initval
initialContext : type = initval
process : type = initval
resultingContext : type = initval
relatedPatterns : type = initval
searchReuseProcessPatternInReusePPLibrary()
drawProcessPattrnTemplate()
Phase Pattern
Stage Pattern
Task Pattern
1
1..*
0..*

Fig. 5.17 ReuseSupport package -Class Diagram(Logical View)

Chapter 6
CASE STUDY EXAMPLE

6.1. Introduction

This chapter reports on an industrial case study example to demonstrate the use of the combined value net and process pattern approach and to examine the prototype software. The example used in this section is based on case study material. Due to the nature of the business and issues of confidentiality, the exact case study material is not available. However, great care has been take to ensure that the following example provides an accurate representation and demonstration of the reuse support system in an industrial context. The company is introduced briefly first. The application of the reuse support system is then demonstrated and problems and research results are illustrated. At the end of the chapter some conclusions restricted to the case study are also reported.

6.2 Description of the company

The Company is part of an international group and they manufacture many types of specialist fuses. The group has different manufacturing sites located globally, and various components are manufactured at different sites and then brought for assembly to a facility managed by the Company.

6.2.1 Products and Production Line

The Company currently manufactures a range of products, which will be referred to here as P. Different varieties of this product are also planned, and individual types will be referred to as P1, P2, P3 .... Pn in this study. A current product, P1, is manufactured from two types of component, 1st component (C1) and 2nd component (C2), and P1 contains 2 x C1, and 1 x C2.

There are 4 main processes in the line: ***Component Preparation, Crimping, Assembly and Inspection***. The components are transported from process to process in a container, and each container carries a batch of the components to manufacture 12 products at a time.

Within the main ***Component Preparation*** process, there are some very slow sub-processes, which are highly dependent on a critical manufacturing resource, which is a type of furnace. This critical resource needs to be kept working almost continuously. The efficient utilisation of this critical resource, and its performance in these processes is crucial to the design and development of the whole production line.

The second process is ***Crimping*** and here the line splits into two identical stations which each perform this process, i.e. ***Crimping1*** and ***Crimping2***, each (containing one sub-process) hence there are two parallel lines at this stage. The lines remain split for the third process, ***Assembly*** which again is implemented as two identical cells, on parallel production lines, i.e. ***Assembly1*** and ***Assembly2***, each containing 7 sub-processes. Finally, both lines join before entering the fourth and final process, ***Inspection***.

A clearer insight into the production line processes is provided by Fig.6.1, which shows each of the main processes and lines of flow of product along the production line. Each of the four main processes will now be described in turn, in further detail.

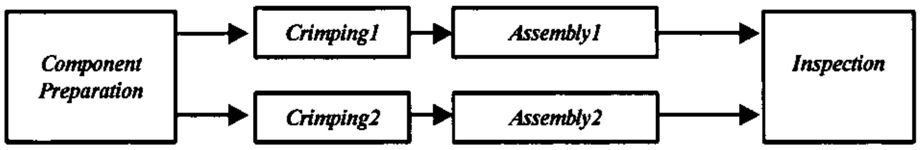


Fig. 6.1 Full Production Line

6.2.1.1 THE FIRST PROCESS, COMPONENT PREPARATION

The ***Component Preparation*** stage begins the production activity. Product P1 are manufactured in batches of 12 at a time. Therefore, initially the two types of components, C1 and C2 are collected in a container, in a proportion of 24:12 items, and they are then released and transported to the next manufacturing stage, the ***Crimping*** process.

Fig. 6.2 shows a rough picture of the **Component Preparation** process.

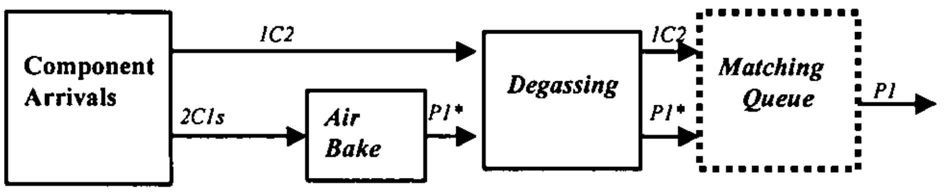


Fig. 6.2. Component Preparation.

6.2.1.2 THE SECOND PROCESS, CRIMPING

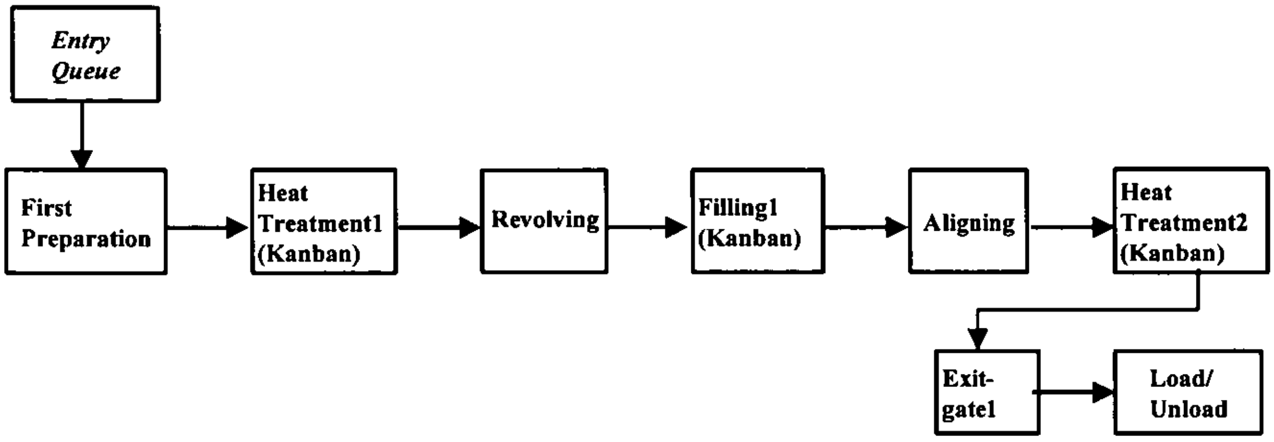
At the second process, ***Crimping***, the line splits into two separate but identical lines, each of them performing the same process, named ***Crimping1*** and ***Crimping2*** respectively. A ***Crimping*** process only contains one sub-process and all the delivery containers are split equally so that equal quantities are transported through each of the two lines.

A human operator is required for the Crimping process, and the human resources, will be referred to as ***HR3*** (for ***Crimping1***) and as ***HR6*** (for ***Crimping2***). A manufacturing resource is also required, and these machines will be referred to as ***Crimper1*** or ***Crimper2***, depending on where the machine is located.

6.2.1.3 THE THIRD PROCESS: ASSEMBLY

The third process, ***Assembly***, is also performed by two different and identical cells, named ***Assembly1*** and ***Assembly2.*** Each assembly process contains seven sub-processes. Each of the assembly lines work independently from each other and have completely different (but identical) sets of machinery and operators, and each undertake identical sets of tasks.

Products are chosen randomly to enter either ***Assembly1*** or ***Assembly2***, and will then have to flow through the seven sub-processes before finally being unloaded from ***Assembly1*** or ***Assembly2***.



Human resources **HR1** and **HR2.**

Fig. 6.3 Example Assembly process (Assembly1)

6.2.1.4 THE FOURTH PROCESS, INSPECTION

Before this fourth process, both separated lines rejoin, and the combined throughput from the parallel lines must be processed by the single process ***Inspection***, which is the last process.

Inspection has just one operator working, **HR8**. Once the containers have gone through the entire process, they are ready to leave this production line. The product manufacture has been completed.

6.3 The Case Study Example

The objective of this case study example is to test the methodology established during this research with realistic industrial data.

When the Company designs a new product they use manufacturing models which capture information about manufacturing resources and knowledge and processes in databases and they also using simulation models to predict and test the performance of new manufacturing lines. It is a time consuming and expensive task to build and evaluate new models. They therefore wish to introduce a Reuse Project to improve their reuse of existing information and knowledge from the simulation and manufacturing models.

The proposed solution works in two stages:

1. The ***Reuse Value Net*** determines the current status within the organisation and targets any identified reuse problem areas for improvement.
2. The ***Process Patterns*** are used to explain the actions that are required to overcome the identified problems.

6.4 The First Step: Formation of the Reuse Value Net

The figure below shows the steps that were performed in the first stage of the solution.

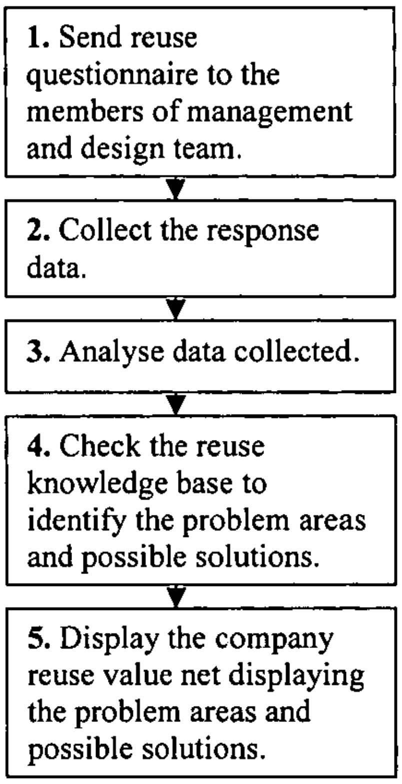

1. Send reuse questionnaire to the members of management and design team.
2. Collect the response data.
3. Analyse data collected.
4. Check the reuse knowledge base to identify the problem areas and possible solutions.
5. Display the company reuse value net displaying the problem areas and possible solutions.

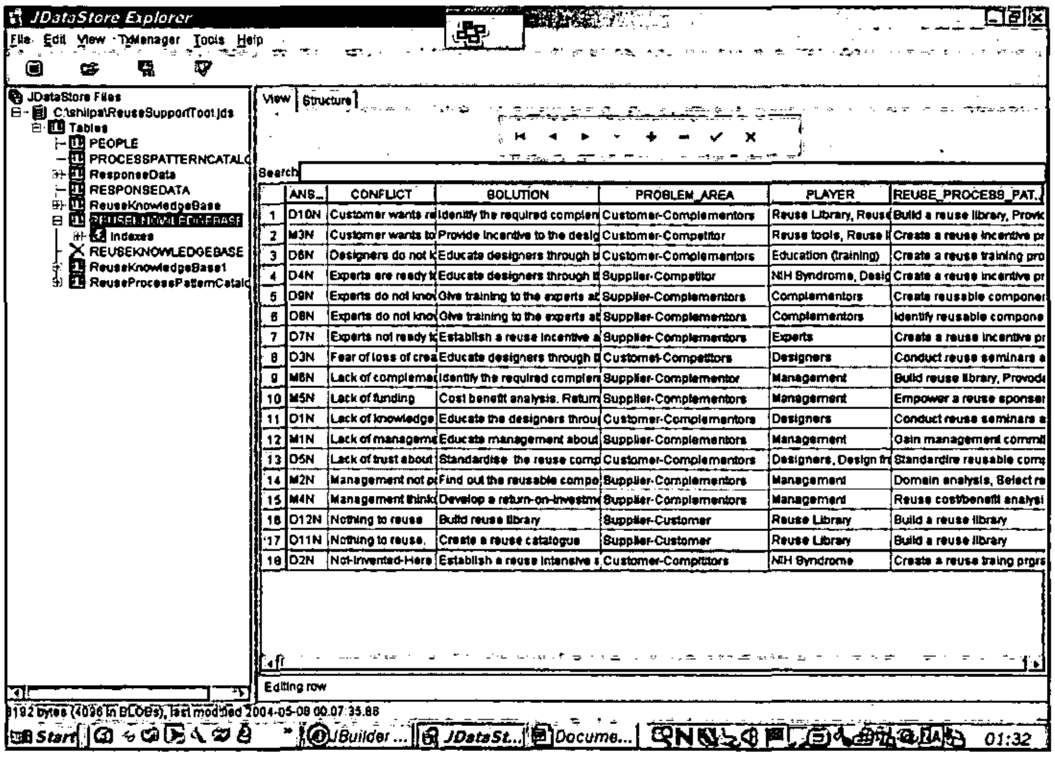
Fig. 6.4 Formation of the Reuse Value Net

6.4.1 **Send the reuse questionnaire to the members of management and design team.** The Company wants to implement reuse practices and it was therefore important to determine the current design practices in the organisation as well as establishing the current status of reuse thinking by management and designers. Questionnaires were therefore given to members of management, designers and users of the simulation models, to establish whether any reuse activities currently exist in the company. The Company has never formally practised reuse in the past, however, the questionnaires are designed to also identify whether individual design engineers are reusing components informally; this may be by using elements of models they have previously built, or even working informally with colleagues and sharing components.

6.4.2 Collect response data. The completed questionnaires were collected and the response data was fed into the questionnaires of the experimental software- the Reuse Agent.

6.4.3 Analyse data collected. The data collected was analysed by the Reuse Agent. The Reuse Knowledge base of the Reuse Agent was used for this purpose. The Reuse Knowledge base is populated on the basis of the relationships identified by the Reuse Value Net. More details of this can be found in Section 5.8. An example screen showing parts of the Reuse Knowledge Base used in the prototype system is shown in figure 6.5.

6.4.4 **Check the reuse knowledge base to identify the problem areas and possible solutions.**



	ANS_	CONFLICT	SOLUTION	PROBLEM_AREA	PLAYER	REUSE_PROCESS_PAT.
1	D10N	Customer wants re	Identify the required complen	Customer-Complementors	Reuse Library, Reuse	Build a reuse library, Provic
2	M3N	Customer wants to	Provide incentive to the desig	Customer-Competitor	Reuse tools, Reuse l	Create a reuse incentive pr
3	D6N	Designers do not k	Educate designers through b	Customer-Complementors	Education (training)	Create a reuse training pro
4	D4N	Experts are ready t	Educate designers through b	Supplier-Competitor	NIH Syndrome, Desig	Create a reuse incentive pr
5	D9N	Experts do not kno	Give training to the experts at	Supplier-Complementors	Complementors	Create reusable componer
6	D8N	Experts do not kno	Give training to the experts at	Supplier-Complementors	Complementors	Identify reusable compone
7	D7N	Experts not ready t	Establish a reuse incentive a	Supplier-Complementors	Experts	Create a reuse incentive pr
8	D3N	Fear of loss of crea	Educate designers through b	Customer-Competitors	Designers	Conduct reuse seminars a
9	M6N	Lack of complemer	Identify the required complen	Supplier-Complementor	Management	Build reuse library, Provode
10	M5N	Lack of funding	Cost benefit analysis. Return	Supplier-Complementors	Management	Empower a reuse sponser
11	D1N	Lack of knowledge	Educate the designers throu	Customer-Complementors	Designers	Conduct reuse seminars a
12	M1N	Lack of managem	Educate management about	Supplier-Complementors	Management	Gain management commit
13	D5N	Lack of trust about	Standardise the reuse comp	Customer-Complementors	Designers, Design fr	Standardize reusable comp
14	M2N	Management not p	Find out the reusable compo	Supplier-Complementors	Management	Domain analysis, Select re
15	M4N	Management think	Develop a return-on-investm	Supplier-Complementors	Management	Reuse cost/benefit analysi
16	D12N	Nothing to reuse	Build reuse library	Supplier-Customer	Reuse Library	Build a reuse library
17	D11N	Nothing to reuse.	Create a reuse catalogue	Supplier-Customer	Reuse Library	Build a reuse library
18	D2N	Not-Invented-Here	Establish a reuse intensive s	Customer-Compititors	NIH Syndrome	Create a reuse traing prgra

Fig. 6.5 Reuse Knowledge Base

The Reuse Agent processed the response data, and in this case, the analysis of the questionnaires identified that :

a. The Management wants to implement reuse in the organisation.

b. Very few designers have practiced reuse individually, but some do occasionally reuse parts of their own codes or programmes. Most designers are not currently practising reuse, they are designing systems from scratch. When they work on a new project, they generally do not try to reuse code or models from previous projects.

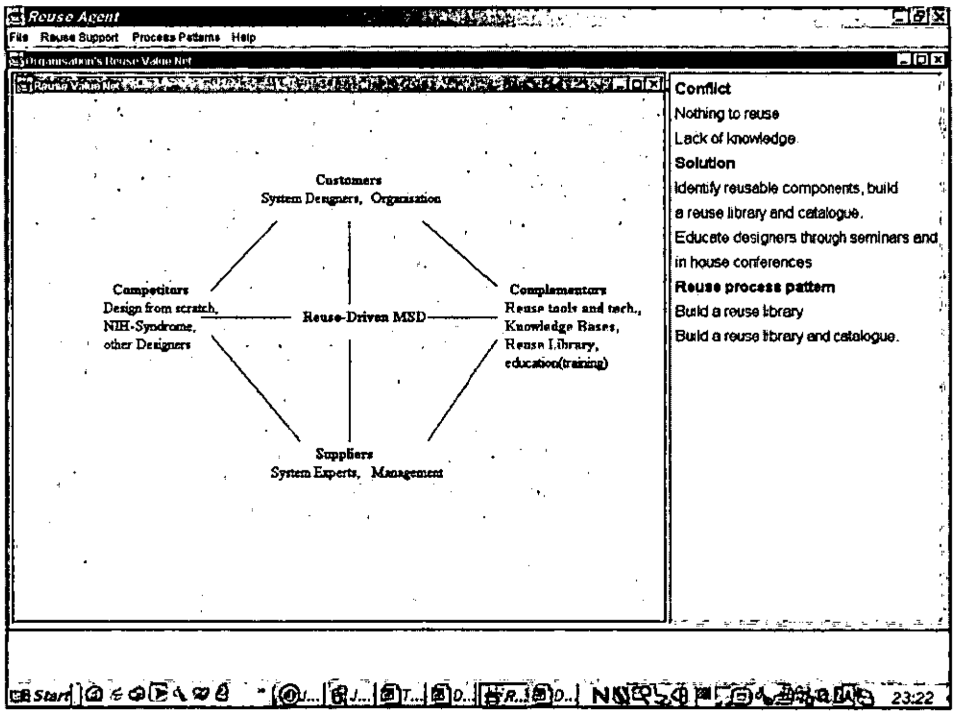
c. Most of the designers have never tried to reuse code or models created by other designers.

d. There are no formal incentive programs or mechanisms to motivate reuse either in place or planned.

e. No reuse library or reuse catalogue currently exists in the organisation.

f. The only sharing across business units that currently exists is of software applications.

6.4.5 **Display the company reuse value net displaying the problem areas and possible solutions.**



Conflict
Nothing to reuse
Lack of knowledge
Solution
Identify reusable components, build a reuse library and catalogue.
Educate designers through seminars and in house conferences
Reuse process pattern
Build a reuse library
Build a reuse library and catalogue.

Fig. 6.6 Company Specific Reuse Value Net

The Reuse Agent analyses these results and determines that the following problem areas needs to be addressed initially:

The possible solutions in order to overcome the problems identified by the Reuse Agent (from the Reuse Knowledge Base):

a. Create a Reuse Catalogue and Reuse Library (Complementor-reuse library)
b. Educate the management and designers (Complementor- Education)

The company specific Reuse Value Net generated by the prototype support tool is shown in fig. 6.6.

6.5 The Second Step: Reuse Process Pattern Guidelines

The figure 6.7 shows the steps involved in the second stage of the solution.

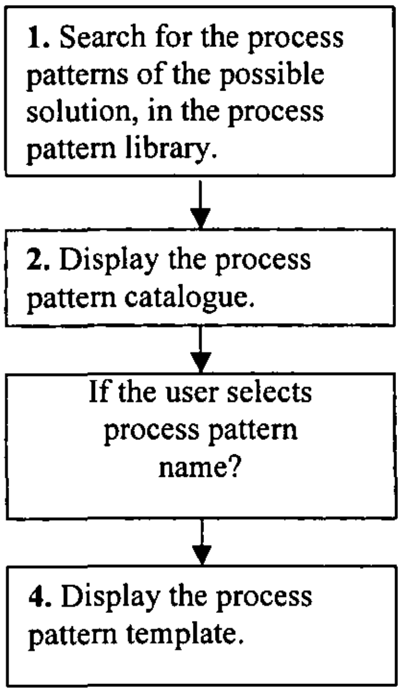
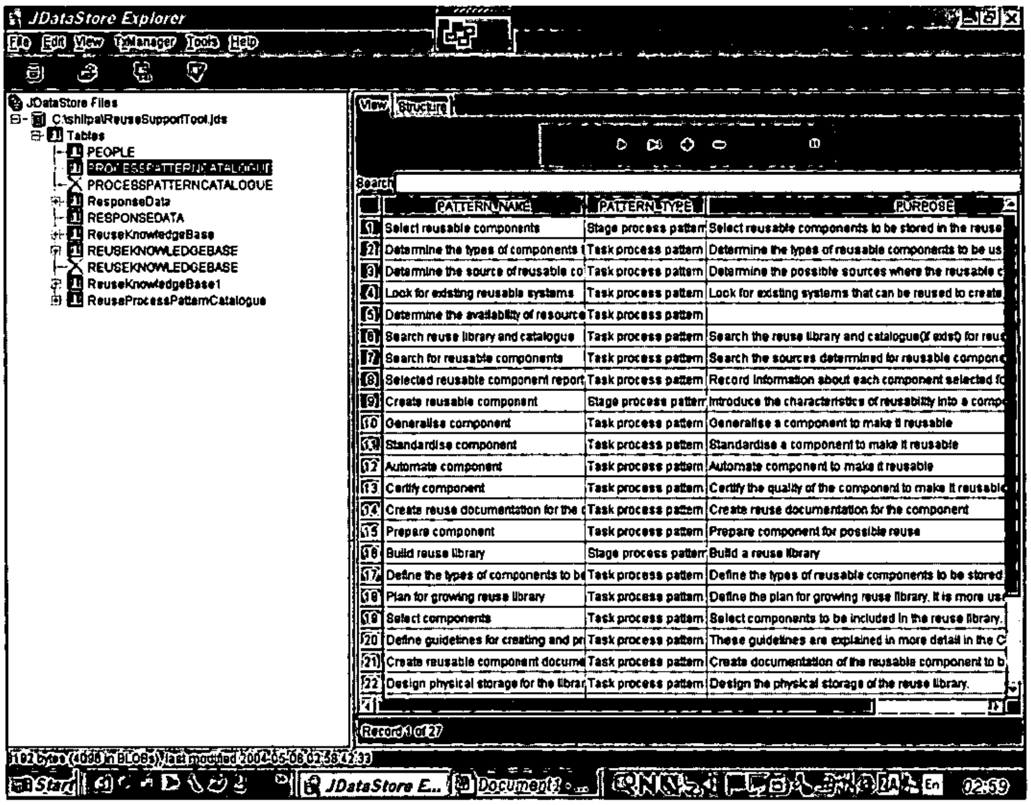

1. Search for the process patterns of the possible solution, in the process pattern library.
2. Display the process pattern catalogue.
If the user selects process pattern name?
4. Display the process pattern template.

Fig. 6.7 Reuse Process Pattern Guidelines

6.5.1 Search for the process patterns of the possible solutions in the process pattern catalogue and library. The possible solutions suggested by the Reuse Agent in the Reuse Value Net formation are stored in the form of patterns in the Reuse Process Pattern Library. The general information of these patterns is provided through the Reuse Process Pattern Catalogue. The Reuse Agent does a search of the related reuse process patterns suggested as solutions.

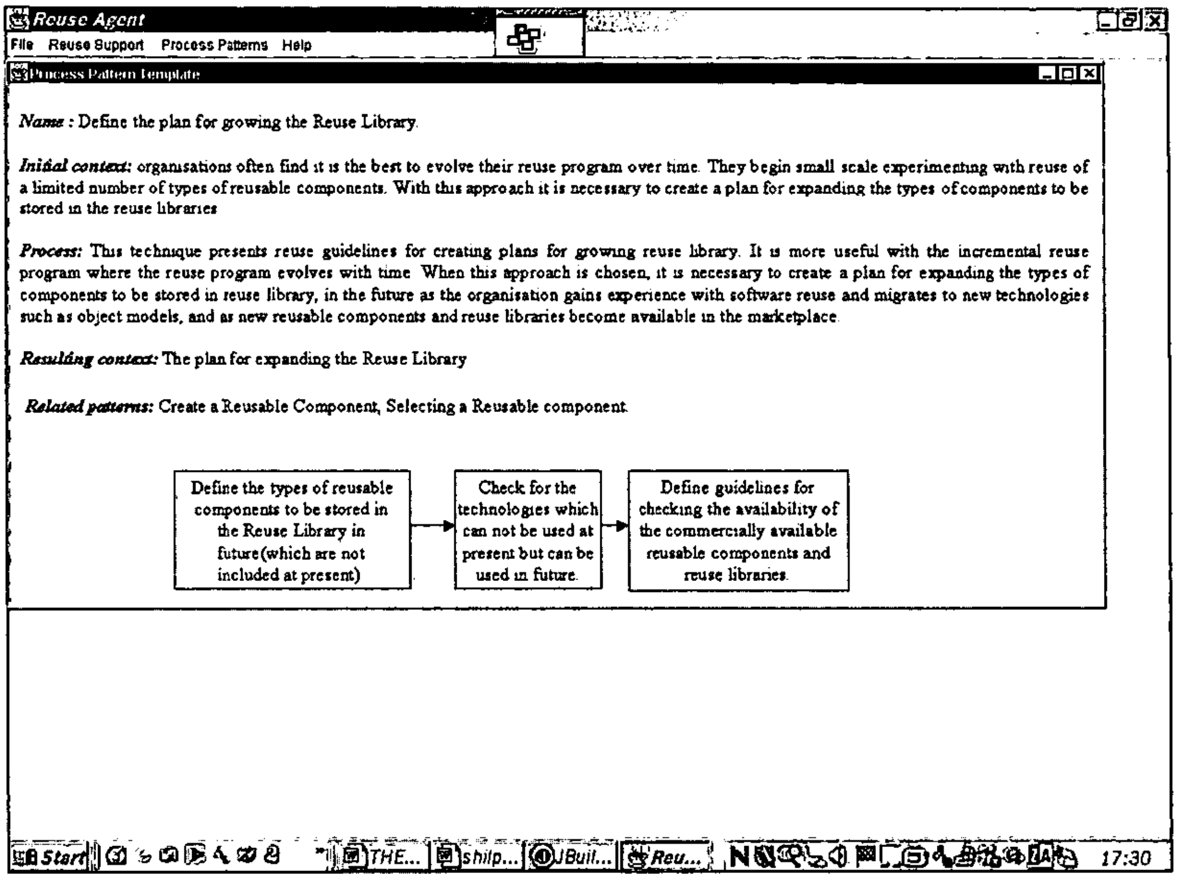
6.5.2 Display the related process patterns from the catalogue. The results of the search by the Reuse Agent for the reuse process patterns are displayed in the form of a table containing the information about the available reuse process patterns. The user can access the reuse process patterns that are stored in the reuse process pattern library by selecting the required reuse process patterns from those displayed from the patterns catalogue. This is shown in fig. 6.8.



	PATTERN_NAME	PATTERN_TYPE	PURPOSE
1	Select reusable components	Stage process patter	Select reusable components to be stored in the reuse
2	Determine the types of components t	Task process pattern	Determine the types of reusable components to be us
3	Determine the source of reusable co	Task process pattern	Determine the possible sources where the reusable c
4	Look for existing reusable systems	Task process pattern	Look for existing systems that can be reused to create
5	Determine the availability of resource	Task process pattern	
6	Search reuse library and catalogue	Task process pattern	Search the reuse library and catalogue(if exist) for reus
7	Search for reusable components	Task process pattern	Search the sources determined for reusable compone
8	Selected reusable component report	Task process pattern	Record information about each component selected fo
9	Create reusable component	Stage process patter	Introduce the characteristics of reusability into a comp
10	Generalise component	Task process pattern	Generalise a component to make it reusable
11	Standardise component	Task process pattern	Standardise a component to make it reusable
12	Automate component	Task process pattern	Automate component to make it reusable
13	Certify component	Task process pattern	Certify the quality of the component to make it reusabl
14	Create reuse documentation for the c	Task process pattern	Create reuse documentation for the component
15	Prepare component	Task process pattern	Prepare component for possible reuse
16	Build reuse library	Stage process patter	Build a reuse library
17	Define the types of components to be	Task process pattern	Define the types of reusable components to be stored
18	Plan for growing reuse library	Task process pattern	Define the plan for growing reuse library. It is more us
19	Select components	Task process pattern	Select components to be included in the reuse library.
20	Define guidelines for creating and pr	Task process pattern	These guidelines are explained in more detail in the C
21	Create reusable component docume	Task process pattern	Create documentation of the reusable component to b
22	Design physical storage for the librar	Task process pattern	Design the physical storage of the reuse library.

Fig. 6.8 Process Pattern Catalogue

6.5.3 Display the process pattern template. The reuse process pattern template of the selected reuse process pattern is displayed from the library. Each template contains the name, intent, process, resulting context, and related patterns information about a particular pattern. The reuse process pattern template contains the guidelines for performing a particular process, and this can be made specific for a particular organisation. an example of this is shown in fig. 6.9.



Name : Define the plan for growing the Reuse Library.

Initial context: organisations often find it is the best to evolve their reuse program over time. They begin small scale experimenting with reuse of a limited number of types of reusable components. With this approach it is necessary to create a plan for expanding the types of components to be stored in the reuse libraries

Process: This technique presents reuse guidelines for creating plans for growing reuse library. It is more useful with the incremental reuse program where the reuse program evolves with time. When this approach is chosen, it is necessary to create a plan for expanding the types of components to be stored in reuse library, in the future as the organisation gains experience with software reuse and migrates to new technologies such as object models, and as new reusable components and reuse libraries become available in the marketplace.

Resulting context: The plan for expanding the Reuse Library

Related patterns: Create a Reusable Component, Selecting a Reusable component.

Define the types of reusable components to be stored in the Reuse Library in future(which are not included at present) → Check for the technologies which can not be used at present but can be used in future. → Define guidelines for checking the availability of the commercially available reusable components and reuse libraries.

Fig. 6.9 Process Pattern Template

The application of the two-stage solution explained above is summarised in the Fig. 6.10. The experimental set up is shown in the top left of the figure 6.10. As explained above, the Reuse Agent initially uses its knowledge to analyse the questionnaire response data and produce a Reuse Value Net. The bold dotted arrow marked 'Stage 1' in fig. 6.10 shows the Reuse Value Net that has been produced for the Company in this example. The elements highlighted in bold show aspects requiring immediate attention. Typically two different types of problems are identified through the Reuse Value Net for the Company, and these have been marked '*1' and '*2' in fig. 6.10. Problems in the area marked '*1' are competitors, and therefore must be tackled by either decreasing their influence or by increasing the influence of reuse (to make reuse a more competitive option).

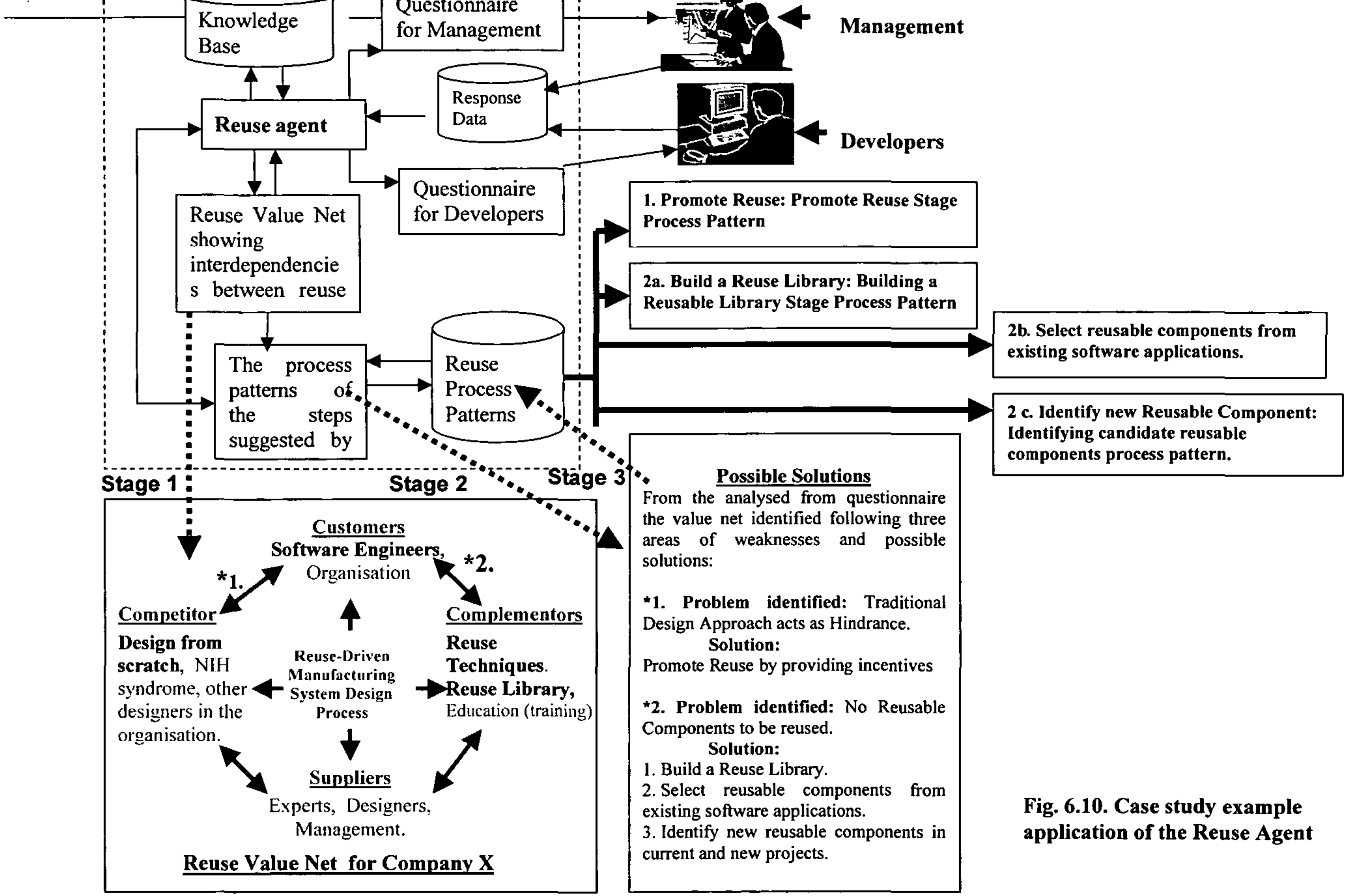
Possible solutions recommended by the Reuse Agent are also shown in fig. 6.10 to the right of the Reuse Value Net. This is indicated by the bold dotted arrow marked 'Stage 2'. In this example, the recommended solution to problem '*1' is to promote reuse by providing incentives. As indicated by the arrows in the experimental set-up figure, the Reuse Agent would also provide a process pattern showing steps that can be taken to achieve this. This is indicated by the bold dotted arrow marked 'Stage 3'. Detailed process pattern steps are not given in fig. 6.10 for problem 1, due to space limitations. However, these steps would include, for example, activities such as setting up a bonus scheme to reward productivity increases through reuse of components and introduction of staff performance measures to assess whether reuse is being used effectively and whether staff are making it easy for others to reuse their components. The process steps suggested by the Reuse Agent are generic in form and need to be specialized by engineers within the Company to suit the requirements and environment of that individual company. The Reuse Agent can guide users through the important steps of specialization, such as the identification of suitable metrics for measurement of productivity, determination of periods of measurement, establishment of types and levels of incentives, etc. However, the actual details have to be decided by the management and users within the company, as successful applications of reuse will inevitably be different and specific to individual companies.

Problems identified in the Reuse Value net in the area marked '*2' on figure 6.10 relate to complementors, and typically any problems in this area indicate the absence of some key elements that are necessary to enable or facilitate reuse. They therefore must commonly be tackled by the provision of additional resources of some kind. In the example of this Company, there are three important activities that should be undertaken:

- Building a Reuse Library.
- Retrospective creation of reusable components from existing sources.
- Ongoing creation of reusable components from current and new projects (this must continue as an ongoing activity).

In figure 6.10, these three solutions are shown as 2.a, 2.b and 2.c.

Figure 6.11 shows the different Building Reuse Library process pattern at the Phase, Stage and Task levels. In order to complete the Building Reuse Library Phase the different stages of Building Reuse Library, Select a Reusable Component and Create a Reusable Components are required, and each of these in turn involve one or more tasks.


Knowledge Base
Questionnaire for Management
Management
Response Data
Developers
Reuse agent
Questionnaire for Developers
Reuse Value Net showing interdependencie s between reuse
The process patterns of the steps suggested by
Reuse Process Patterns
1. Promote Reuse: Promote Reuse Stage Process Pattern
2a. Build a Reuse Library: Building a Reusable Library Stage Process Pattern
2b. Select reusable components from existing software applications.
2 c. Identify new Reusable Component: Identifying candidate reusable components process pattern.
Stage 1
Stage 2
Stage 3
Customers
Software Engineers, Organisation
*1.
*2.
Competitor
Design from scratch, NIH syndrome, other designers in the organisation.
Reuse-Driven Manufacturing System Design Process
Complementors
Reuse Techniques. Reuse Library, Education (training)
Suppliers
Experts, Designers, Management.
Reuse Value Net for Company X

Possible Solutions

From the analysed from questionnaire the value net identified following three areas of weaknesses and possible solutions:

***1. Problem identified:** Traditional Design Approach acts as Hindrance.

Solution:

Promote Reuse by providing incentives

***2. Problem identified:** No Reusable Components to be reused.

Solution:

1. Build a Reuse Library.
2. Select reusable components from existing software applications.
3. Identify new reusable components in current and new projects.

Fig. 6.10. Case study example application of the Reuse Agent

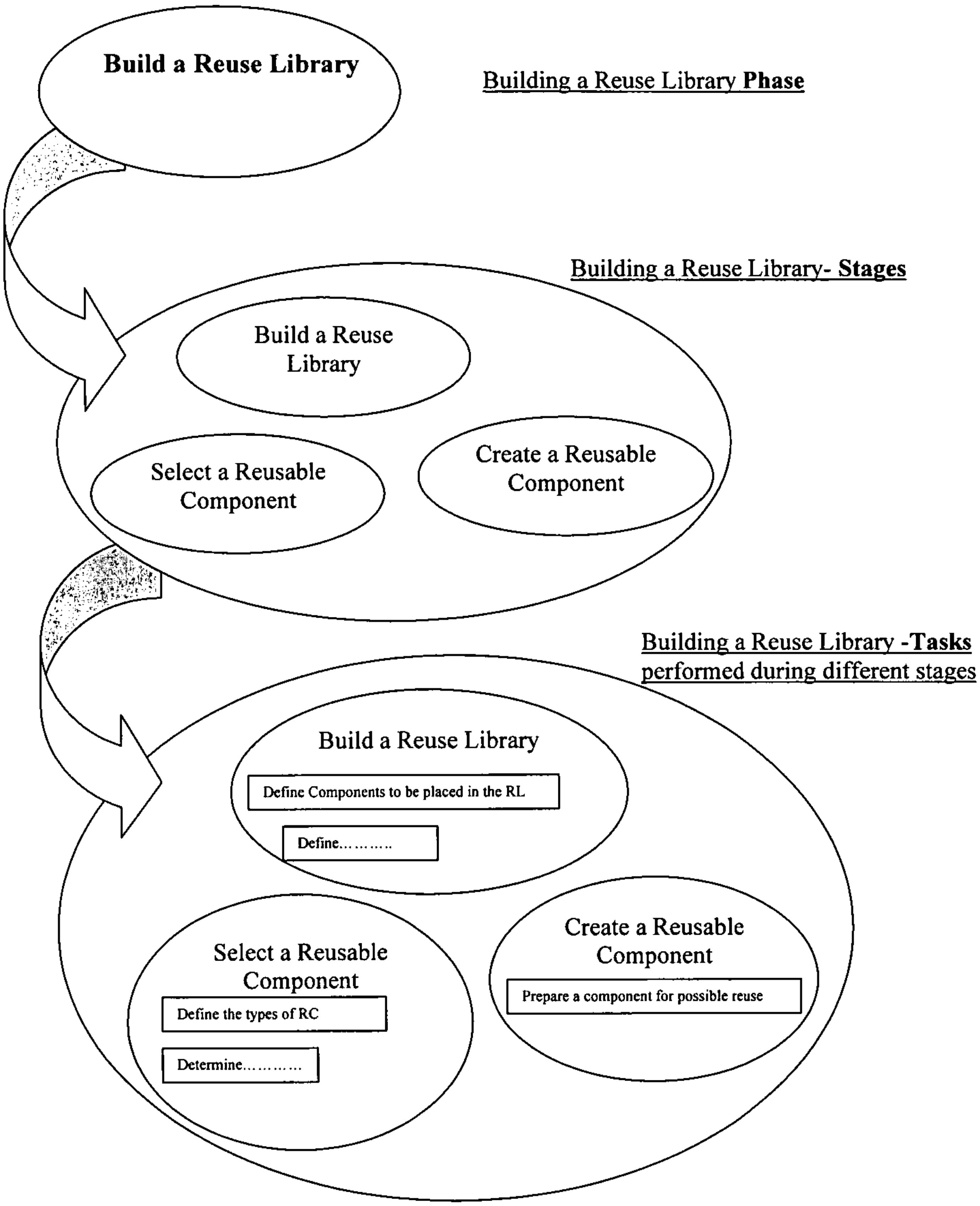

Build a Reuse Library
Building a Reuse Library Phase
Building a Reuse Library- Stages
Build a Reuse Library
Select a Reusable Component
Create a Reusable Component
Building a Reuse Library -Tasks performed during different stages
Build a Reuse Library
Define Components to be placed in the RL
Define…………
Select a Reusable Component
Define the types of RC
Determine…………
Create a Reusable Component
Prepare a component for possible reuse

Fig. 6.11 Building Reuse Library Process Patter at Phase, Stage, and Task levels

Name : Build Reuse Library.

Initial context: A prerequisite of practicing reuse is having something to reuse, that is, reusable components such as code, class libraries, templates, design models, and so on. A Reuse Library is an important element of formalising the practice of reuse in a corporation. It provides the mechanism to properly manage reusable components and make them available to software system developers across a corporation.

Process: This technique presents reuse guidelines for defining the types of reusable components to be stored in the reuse library, physically and logically organise them, define a classification scheme for the reuse library, set up a reuse catalogue and to select the tools to support a Reuse Library and Reuse Catalogue.

Resulting context: The Reuse Library and Catalogue.

Related patterns: Create a Reusable Component, Selecting a Reusable component.

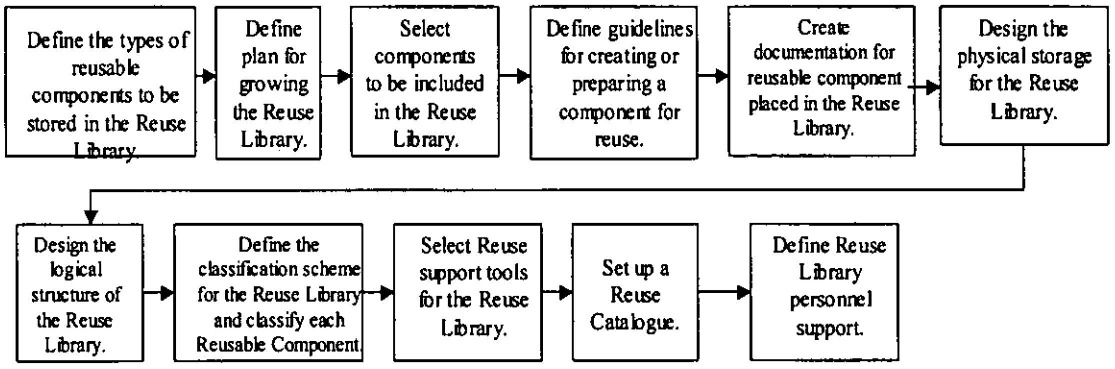

Define the types of reusable components to be stored in the Reuse Library.
Define plan for growing the Reuse Library.
Select components to be included in the Reuse Library.
Define guidelines for creating or preparing a component for reuse.
Create documentation for reusable component placed in the Reuse Library.
Design the physical storage for the Reuse Library.
Design the logical structure of the Reuse Library.
Define the classification scheme for the Reuse Library and classify each Reusable Component
Select Reuse support tools for the Reuse Library.
Set up a Reuse Catalogue.
Define Reuse Library personnel support.

Fig. 6.12 Build Reuse Library-phase process pattern template

Name : Define the types of reusable components to be stored in the Reuse Library.

Initial context: An essential element of practicing reuse is to define and plan what to reuse. The types of components that are to be reused must be decided early in the Reuse Program. They must be known to define the requirements for the Reuse Library, and how the software process model must be changed to accommodate both the creation and use of these types of components.

Process: This technique presents reuse guidelines for defining types of components to be placed in the Reuse Library. Begin by building a list of types of components that are to be stored in the Reuse Library. When deciding upon the components it is important to note that often there are larger payoffs and grater benefits associated with larger reusable components and the components that are used in the early phases of system design. For each type of reusable component, determine if it can be created or acquired such as by extraction from existing legacy applications, by acquisition from outside sources such as vendors, or creation from scratch.

Resulting context: List of types of components to be placed in the Reuse Library.

Related patterns: Create a Reusable Component, Selecting a Reusable component.

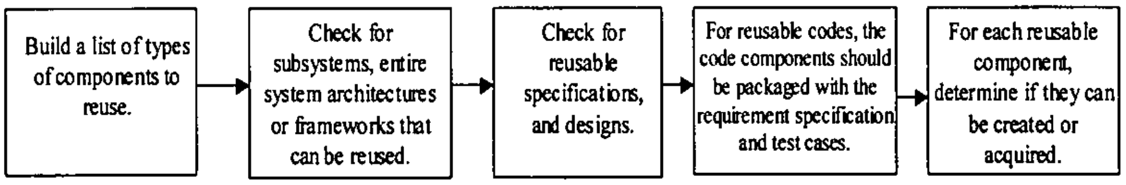

Build a list of types of components to reuse.
Check for subsystems, entire system architectures or frameworks that can be reused.
Check for reusable specifications, and designs.
For reusable codes, the code components should be packaged with the requirement specification and test cases.
For each reusable component, determine if they can be created or acquired.

Fig. 6.13 Define the types of reusable components to be stored in the Reuse Library- task process pattern template

Figure 6.12 shows the Build Reuse Library process pattern template. The several requirements to be completed for this process are as shown at the bottom of the template, and are each described in the following numbered list.

1. Defining the types of reusable components to be stored in the Reuse Library.

At this stage the Company needs to decide what kind of reusable components should be stored in the reuse library. Potentially many different types of reusable components could be stored ranging from entire simulation models of production line designs for particular product varieties, e.g. P1, to partial models or simulation models of particular cells. Any types of Knowledge that needs to commonly be applied could also be turned into reusable components. These might include descriptions of particular manufacturing resources (including their capabilities, e.g. product types used on, constraints for use, e.g. dimensional constraints, cost constraints, batch size constraints, etc., average throughput rates, layout footprint, etc.). Smaller pieces of knowledge may also make relevant reusable components, for example, simple sets of rules for the selection of resources for particular process types, or knowledge relating to quality or safety checks which need to be carried out with particular product types or process types.

In the current example, potentially reusable components can be identified in section 6.2. It is unlikely that the complete simulation models for particular products (such as P1) would make a useful reusable component as details within particular processes will vary between product types. The batch sizes (i.e. 12 in the case of P1), will also vary and affect the descriptions of some resources (such as the transporting container). However some processes, such as ***Crimping***, are very similar across product varieties, as are some of the sub-processes within ***Assembly*** and ***Inspection***, and therefore small models of these processes could be valuable reusable components. Similarly, in the current example, descriptions of particular manufacturing resources, such as furnaces or crimpers, or descriptions of training required for human resources or knowledge of successful task allocation sequences, may also be developed as reusable components. Figure 6.13 shows the process pattern template for the task Define the types of reusable components to be stored in the Reuse Library.

2. *Define a Plan for growing the Reuse Library.*

The Reuse Library is not a static entity. It should be dynamic and grow and mature as reuse becomes more universally accepted within the Company, and as more useful reusable components are progressively identified. However, it is not cost effective to allow the Reuse Library to grow without purpose or reason. For example, if several different types of a particular machine may be used, it may be more appropriate to generate one 'general purpose' reusable component to store the manufacturing resource descriptions or simulation models for these. The generic parts may be fully described, and instructions provided on how to specialise the remaining parts depending on which type of the machine is required for a particular application. Also, there must be a growing case for defining reusable components that are partial simulation models of single processes or cells, rather than larger-scale models, as the technologies for distributed simulation mature and become easier to apply. (This last point is particularly relevant for the second activity in this process pattern template, i.e. *Check for the technologies which cannot be used at present but can be used in the future.*)

Name : Define the plan for growing the Reuse Library.

Initial context: organisations often find it is the best to evolve their reuse program over time. They begin small scale experimenting with reuse of a limited number of types of reusable components. With this approach it is necessary to create a plan for expanding the types of components to be stored in the reuse libraries.

Process: This technique presents reuse guidelines for creating plans for growing reuse library. It is more useful with the incremental reuse program where the reuse program evolves with time. When this approach is chosen, it is necessary to create a plan for expanding the types of components to be stored in reuse library, in the future as the organisation gains experience with software reuse and migrates to new technologies such as object models, and as new reusable components and reuse libraries become available in the marketplace.

Resulting context: The plan for expanding the Reuse Library.

Related patterns: Create a Reusable Component, Selecting a Reusable component.

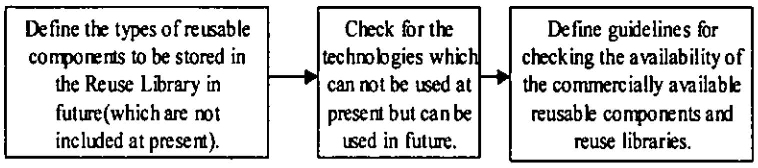

Define the types of reusable components to be stored in the Reuse Library in future(which are not included at present).
Check for the technologies which can not be used at present but can be used in future.
Define guidelines for checking the availability of the commercially available reusable components and reuse libraries.

Fig. 6.14 Define the plan for growing Reuse Library- task process pattern template

3. Select components to be included in the Reuse Library.

The next requirement is to assist the Company in selecting the reusable components to be included in the reuse library, and the Select Reusable Component process pattern can be used to guide the developer at this stage. Fig. 6.15 shows the Select Reusable Component process pattern template. The first task shown in this template is to determine the types of reusable components to search for. In this case the types of reusable component to search for are the same as those defined in the task process pattern *Define the types of reusable components to be stored in the Reuse library*, in the Build Reuse Library stage process pattern, i.e. those discussed in step 1 above.

The second task in this requirement, is to look for existing systems that could be reused to create this (new) system. So, for example, if the company is considering the introduction of a new product, P2, that has some similarities with the existing product P1. Some parts of the P1 models ore knowledge used in their creation, could be used to generate appropriate reusable elements for P2. The next task in the Select Reusable Component stage is to determine the sources of the reusable components to search for. In the current case, the Company could identify potential sources within existing manufacturing models and simulation models (for example those for P1), and those existing from previous or earlier projects. Therefore existing legacy systems and the other concurrent projects in the organisation may be determined as the sources of the reusable components which may be searched. Figure 6.16 shows the Determine the Sources of Reusable Components – task process pattern.

The fifth task in the Select Reusable Component process pattern is to search for the reusable components in the Reuse Library and Catalogue. At present there is no Reuse Library or Catalogue for the present Company example. Hence, it is necessary to move on to the following stage, which is to search for the reusable components. As the existing system for product P1, and existing legacy systems and other concurrent projects in the organisation were determined as the potential sources of reusable components, this is the point when these sources should be searched for the reusable components. Figure 6.17 shows the process pattern for Look for existing systems that can be reused to create a new system.

Name : Select reusable component.

Initial context: To implement reuse strategy it is important to find and select appropriate reusable components that can be used to develop the various system projects deliverables.

Process: This phase pattern consists of stage process patterns determining the types of reusable components, sources of reusable components, availability of appropriate resources, search for different reusable components, and recording information about the reusable components.

Resulting context: A list of and information about reusable components selected for reuse in current project.

Related patterns: Building a Reuse Library, Creating a Reusable component.

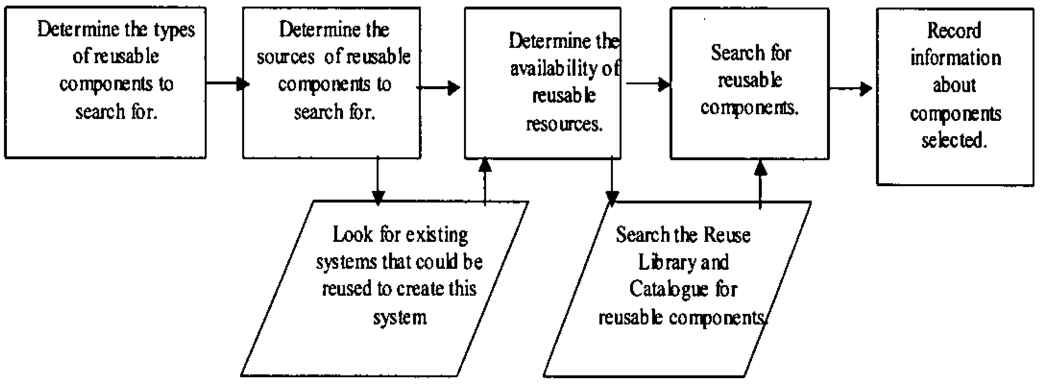

Determine the types of reusable components to search for.
Determine the sources of reusable components to search for.
Determine the availability of reusable resources.
Search for reusable components.
Record information about components selected.
Look for existing systems that could be reused to create this system
Search the Reuse Library and Catalogue for reusable components

Fig. 6.15 Select Reusable Components process pattern template

Name : Determine the source of reusable components.

Initial context::

Forces: To determine the possible sources where the the reusable components to be used in the current project can be searched.

Process: Check all the possibilities where reusable components can be found.

Resulting context: One or more sources to search for the reusable components.

Related patterns: Building a Reuse Library, Creating a Reusable component.

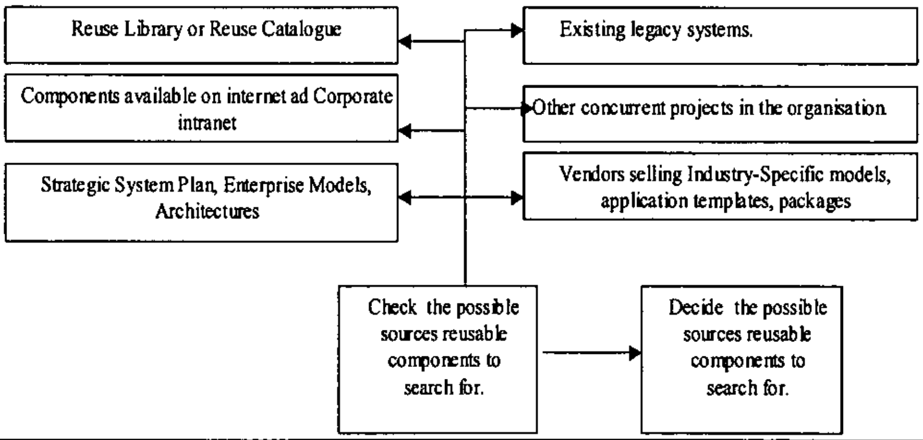

Reuse Library or Reuse Catalogue
Components available on internet ad Corporate intranet
Strategic System Plan, Enterprise Models, Architectures
Existing legacy systems.
Other concurrent projects in the organisation.
Vendors selling Industry-Specific models, application templates, packages
Check the possible sources reusable components to search for.
Decide the possible sources reusable components to search for.

Fig. 6.16 Determine the Source of Reusable Component – process pattern template

Name : Look for existing system that can be reused to create the new system.

Initial context: This system might be a redesign or replacement of an existing system.in such cases as much as 60% of the functionality from the existing system will be duplicated in the new system.

Process: This pattern guides to search for an existing system and to determine whether or not to use the existing system in whole or in part in the development of the new system. To asses the quality of the existing system check : Is it properly modularised, well documented, thoroughly tested, and standards compliant? What is its usage history, reliability record, and change history? In order to consider the maintenance consequences check, will reuse of the existing system result in two sets of system parts that must be maintained? Or will the existing system parts be used to create new generic parts that will then be replace the existing parts and be used in both the new and existing systems?

Resulting context: An existing system in whole or in parts that can be used in the new system design.

Related patterns: Reuse Cost/Benefit Analysis, Building a Reuse Library, Creating a Reusable component.

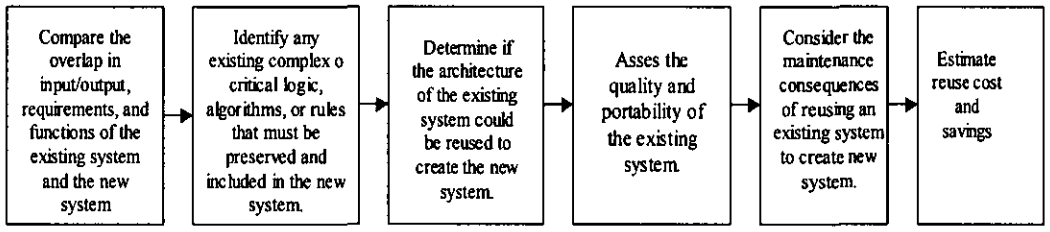

Compare the overlap in input/output, requirements, and functions of the existing system and the new system
Identify any existing complex o critical logic, algorithms, or rules that must be preserved and included in the new system.
Determine if the architecture of the existing system could be reused to create the new system.
Asses the quality and portability of the existing system.
Consider the maintenance consequences of reusing an existing system to create new system.
Estimate reuse cost and savings

Fig. 6.17 Look for existing systems that can be reused to create the new system

In order to apply reuse in the design of product P2, the Company needs to identify particular similarities and differences between the two products, P1 and P2. Close examination of the requirements of the two products confirms that they have many similarities and therefore the architectures for the design information and simulations should be very similar, and the architecture of the existing system (P1) can be reused to create the new system. Models for some sub-processes of P1, e.g. ***Crimping***, can also be reused. In addition, safety check and quality check and documentation templates can also be reused from existing legacy systems. However, the following points identify some differences that the designers have identified as needing to be considered in the new work on P2:

- P2 is a 'Safety Critical' fuse, which needs additional quality checks to be introduced. Hence additional processes must be defined and modelled as required, or perhaps additional reusable components may be obtained from previous safety critical projects within the legacy systems.
- The locations of the three new quality checks are determined to be:- 1 after crimping, 2-after assembly and 3- after degassing.

- The component size is different, and therefore the number of components in each container will need to be different. Hence a new specification and model are required for the new container transporter. This will also affect how many components can go through the degassing process at one single time and the matching queues process.
- Size also affects batch quantities that can be processed through Heat treatment 1 and Heat treatment 2.
- Basic Kanban models would be kept the same as P1 for initial performance simulation checks, but these may be modified if required performance on the new line is not achieved.

The last task in Selecting a Reusable Component is to make a Selected Reusable Component Report and the process pattern template for this is shown in fig. 6.18.

Name : Selected reusable component report.

Initial context: This process pattern provides a template to record the information about the selected reusable component.

Process: Use the Selected Reusable Component template to record the information about the each component selected for reuse in the current project.

Resulting context:. A Selected Reusable Component Report for each component that is selected for reuse in the new system/project.

Related patterns: Building a Reuse Library, Creating a Reusable component.

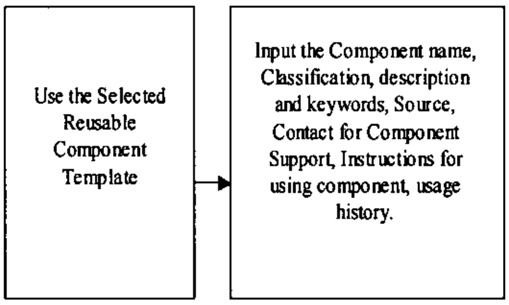

Use the Selected Reusable Component Template
Input the Component name, Classification, description and keywords, Source, Contact for Component Support, Instructions for using component, usage history.

Fig. 6.18 Selected Reusable Component Report process pattern template

4. Define guidelines for creating and preparing a component for reuse.

This is the fourth requirement in the Build Reuse Library process pattern template and the process pattern template for this is shown below.

Name : Define guidelines for creating and preparing a component for reuse.

Initial context:For each type of component, it is necessary to define guidelines for creating or preparing the component for reuse. This is because reusable components by nature are more generic and standardised than components that have been designed for one time use. Also, more effort is needed to guarantee high quality and high confidence in the components and to develop additional documentation needed for the reuser.

Process: This technique presents reuse guidelines that are used to introduce and/or determine the presence of these characteristics in a component intended for reuse.in general there are five basic characteristics that make a component reusable: Generic, Standardised, Automated, Documented, and Certified.

Resulting context: Reusable Component.

Related patterns: Creating a Reusable Component, Selecting a Reusable component.

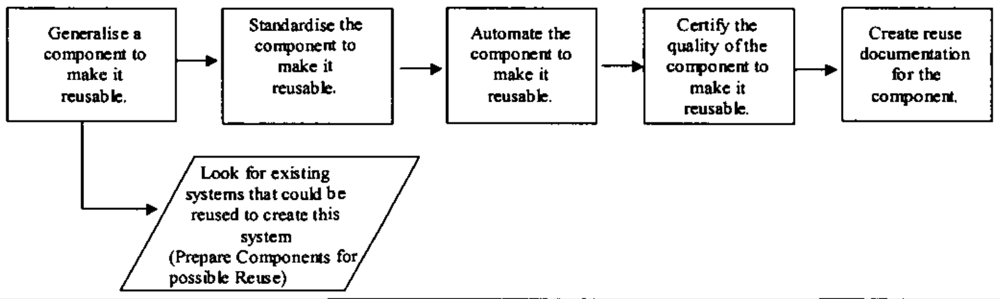

Generalise a component to make it reusable.
Standardise the component to make it reusable.
Automate the component to make it reusable.
Certify the quality of the component to make it reusable.
Create reuse documentation for the component.
Look for existing systems that could be reused to create this system (Prepare Components for possible Reuse)

Fig. 6.19 Define guidelines for creating and preparing a component for reuse-process pattern template

5. Create documentation for reusable components placed in the Reuse Library.

Once suitable reusable components have been identified, it is essential that suitable documentation is created so that it can be retrieved quickly and efficiently from the library. The following process pattern template, shown in Fig 6.20, lists the necessary activities to achieve appropriate documentation for successful future searching. As usual, these activities should be specialised to meet the requirements of the particular company.

For completeness, the process patterns for steps 6 to 11 of the Build Reuse Library pattern have also been included, in the following numbered sections. These activities are clearly very important in the introduction and implementation of the Reuse programme within a company, however, they could not be undertaken in detail in this case study example. The main reasons for this is that these activities need to be specialised to particularly satisfy the existing computing requirements and environments of the individual company. In addition, the necessary activities are very labour intensive and require substantially greater resources than are available in a single PhD research project.

Name : Create documentation for reusable components placed in the Reuse Library.

Initial context: A reusable component is not placed as a stand alone entity in the library. In addition, it is documented to enable a reuser to easily and quickly identify it, understand it, and use it as a building block in a new system.

Process: This technique presents reuse guidelines for documenting information about a reusable component to be placed in the reuse library. The minimal documentation that should be associated with a reusable component is : Component name, short description, Classification of component based on the classification scheme, Domain for which the component was created, Instruction about using the component, including physical environment required, performance constraints, and legal restrictions, Test suit including test plan, test data, and expected test results, Component itself or a pointer to its actual location, Quality of component such as error rate, documentation quality, and standard compliance, Recommendations for improving the component, and History of component's reuse.

Resulting context: Reusable Component Documentation.

Related patterns: Create a Reusable Component, Selecting a Reusable component.

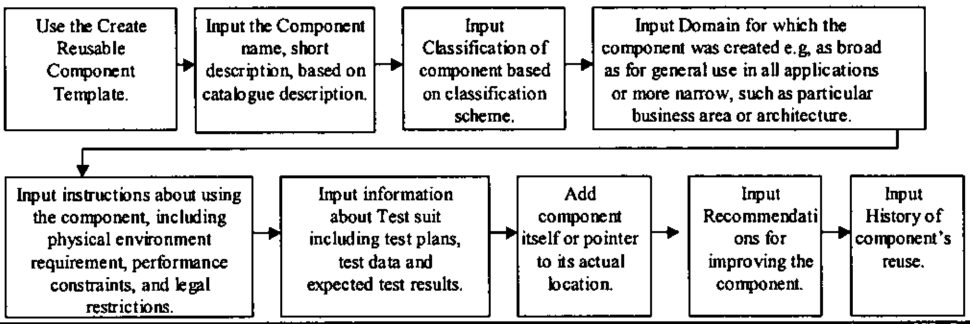

Use the Create Reusable Component Template.
Input the Component name, short description, based on catalogue description.
Input Classification of component based on classification scheme.
Input Domain for which the component was created e.g, as broad as for general use in all applications or more narrow, such as particular business area or architecture.
Input instructions about using the component, including physical environment requirement, performance constraints, and legal restrictions.
Input information about Test suit including test plans, test data and expected test results.
Add component itself or pointer to its actual location.
Input Recommendations for improving the component.
Input History of component's reuse.

Fig. 6.20 Create documentation for reusable component placed in the reuse library- process pattern template

6. *Design the physical storage for the Reuse Library.*

Name : Design the physical storage for the Reuse Library.

Initial context: The Reuse Library can be designed as a physically centralised library that can be accessed by system builders across the enterprise or as several local, distributed libraries that support different software development organisations or different domains..

Process: This technique presents reuse guidelines for designing the physical stored in for the Reuse Library. The three types of physical storage can be considered, Centralised library, Distributed, local reuse libraries, or Centralised and distributed reuse library.

Centrally organised libraries are often formally defined in the sense that they follow corporate standards such as naming conventions and certification criteria. They are usually supported by an official librarian to support population, access, administration and maintenance. The problem with a centralised reuse library is that it can become very large, (since it contains reusable components for the entire organisation) increasing the difficulty and cost of maintenance.

A Reuse Library can be organised as several **local distributed libraries** that individually support these reuse needs of a particular development group or domain, the size problem can be avoided. The problem with local distributed libraries is that there may be duplication of similar components across various local libraries, or there may be components in one local library that would be very useful to other groups and should be included in their local libraries.

The third type of storage combines the local and centralised structures, the reusable components are first placed in a local library and later promoted to the central library by the Corporate Reuse Administration Group if the component seems to be useful for multiple groups. When both central and local reuse libraries are created, each development group has access to both a shared central library and to its own local library.

Resulting context: The Reuse Library and Catalogue.

Related patterns: Create a Reusable Component, Selecting a Reusable component.

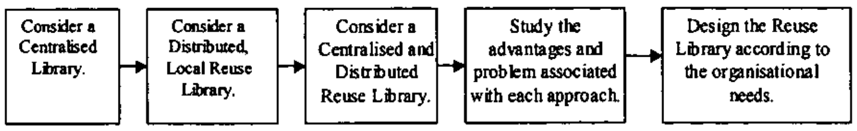

Consider a Centralised Library.
Consider a Distributed, Local Reuse Library.
Consider a Centralised and Distributed Reuse Library.
Study the advantages and problem associated with each approach.
Design the Reuse Library according to the organisational needs.

Fig. 6.21 Design physical storage for reuse library-process pattern template

7. *Design the logical structure of the Reuse Library*

Name : Design the logical structure of the Reuse Library.

Initial context: Reuse libraries are often organised into logical layers where each layer represents a higher level of certification. In this manner the libraries can be created more quickly and cheaply since components do not have to be fully prepared or certified for reuse. The reuser knows for certain what to expect from the component, since its position in the library defines the quality and reuse certification.

Process: This technique presents reuse guidelines for designing the logical structure of the reuse library. One possible way of organising the reuse library logically is in four layers, where components at the top are designed or reengineered for reuse and are fully certified. At the bottom layer, components are placed into the library without any preparation for reuse. Components that are frequently used can be promoted to higher levels in the library.

Resulting context: Logical structure of the reuse library.

Related patterns: Create a Reusable Component, Selecting a Reusable component.

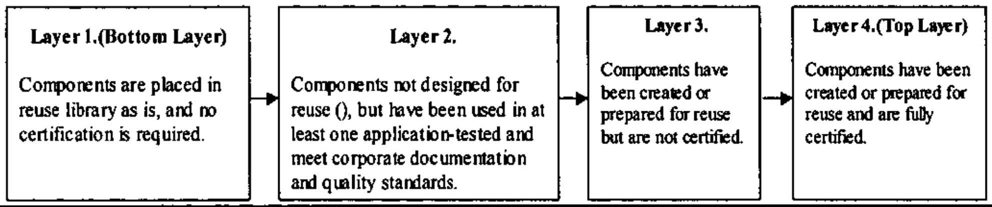

Layer 1.(Bottom Layer)
Components are placed in reuse library as is, and no certification is required.
Layer 2.
Components not designed for reuse (), but have been used in at least one application-tested and meet corporate documentation and quality standards.
Layer 3.
Components have been created or prepared for reuse but are not certified.
Layer 4.(Top Layer)
Components have been created or prepared for reuse and are fully certified.

Fig. 6.22 Design logical structure of the reuse library.

8. *Define the classification scheme for the Reuse Library and classify each Reusable Component.*

Name : Define the classification scheme for the Reuse Library and classify each Reusable Component.

Initial context: Classifying the reusable components into meaningful structured allows the library to be easily searched by providing a way for reusers to match their current needs to reusable components contained in the library.

Process: This technique presents reuse guidelines for defining the classification scheme for the reuse library.most reuse classification schemes are based on either uncontrolled: free text; or controlled: enumerated, faceted, or attribute-value methods.

Resulting context: Reuse library classification scheme, classified reusable components.

Related patterns: Create a Reusable Component, Selecting a Reusable component.

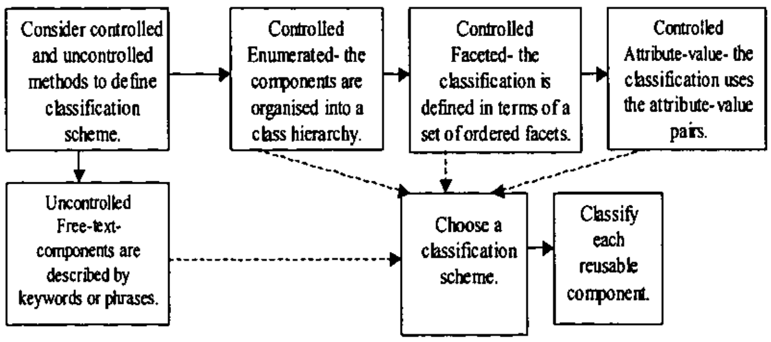

Consider controlled and uncontrolled methods to define classification scheme.
Controlled Enumerated- the components are organised into a class hierarchy.
Controlled Faceted- the classification is defined in terms of a set of ordered facets.
Controlled Attribute-value- the classification uses the attribute-value pairs.
Uncontrolled Free-text- components are described by keywords or phrases.
Choose a classification scheme.
Classify each reusable component.

Fig. 6.23 Define Classification scheme

9. Select Reuse support tools for the Reuse Library.

Name : Select reuse support tools for the Reuse Library.

Initial context: Reuse support tools are useful as they make search of a reusable component quick and easy.

Process: This technique presents the different support tools those can be considered to support the reuse library. The repository tool enhances communication and sharing of information across tools, lifecycle activities, teams, and applications. Configuration management tool helps to manage and keep track of each reusable component in terms of its versions, changes made to the component and reusers of the components. Browser tools are very important reuse support tools because they help the reuser locate desired components. Reuse catalogue tools can be used to automatically scan a library or file of existing components and extract some descriptive information about the reusable components.

Resulting context: List of reuse support tools.

Related patterns: Create a Reusable Component, Selecting a Reusable component.

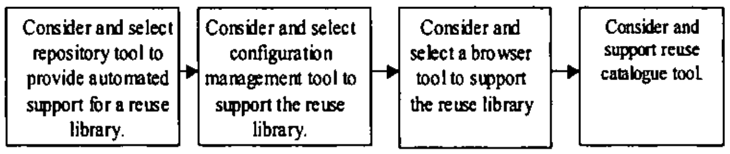

Consider and select repository tool to provide automated support for a reuse library.
Consider and select configuration management tool to support the reuse library.
Consider and select a browser tool to support the reuse library
Consider and support reuse catalogue tool.

Fig. 6.24 Select reuse support tools

10. Set up a Reuse Catalogue.

Name : Set up a Reuse Catalogue.

Initial context: At the early stages of introducing system reuse, a corporation may choose to set up a Reuse Catalogue before building a Reuse library in order to learn about the different types of reusable components and which one in particular would be most beneficial to redesign or create for reuse in place in a Reuse Library.

Process: This technique presents reuse guidelines to set up a Reuse Catalogue. A catalogue is used to assist the identification and retrieval of candidate reusable components and does not require the creation of Reuse Library. It is much faster and cheaper to create than Reuse Library. Many cataloguing tools are available in commercially. A Reuse Catalogue requires someone to be responsible for setting up and maintaining the catalogue, which includes creating components descriptions, entering them into the catalogue, maintaining the descriptions, and controlling access to the catalogue.

Resulting context: The Reuse Library and Catalogue.

Related patterns: Create a Reusable Component, Selecting a Reusable component.

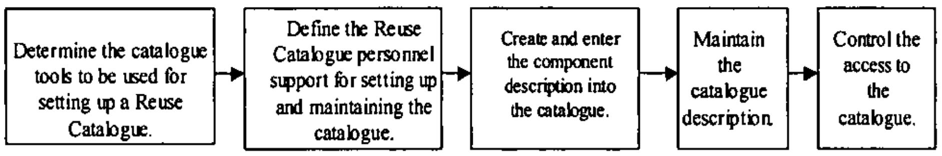

Determine the catalogue tools to be used for setting up a Reuse Catalogue.
Define the Reuse Catalogue personnel support for setting up and maintaining the catalogue.
Create and enter the component description into the catalogue.
Maintain the catalogue description.
Control the access to the catalogue.

Fig. 6.25 Set up Reuse Catalogue

11. *Define Reuse Library personnel support.*

Name : Define Reuse Library personnel support.

Initial context: Library staff is needed to establish, manage, and support the Reuse Library.

Process: This technique presents reuse guidelines for defining the types of reusable components to be stored in the reuse library, physically and logically organise them, define a classification scheme for the reuse library, set up a reuse catalogue and to select the tools to support a Reuse Library and Reuse Catalogue.

Resulting context: The Reuse Library and Catalogue.

Related patterns: Create a Reusable Component, Selecting a Reusable component.

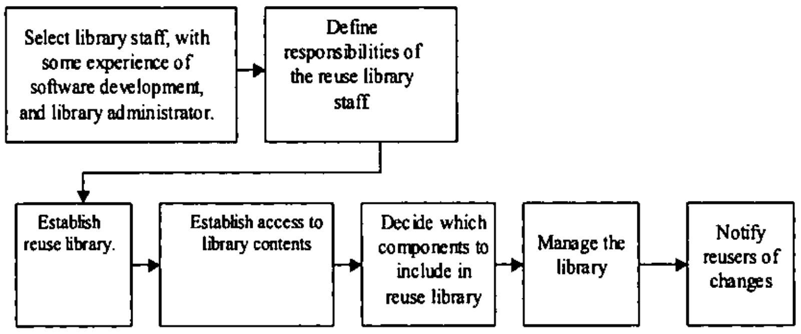

Select library staff, with some experience of software development, and library administrator.
Define responsibilities of the reuse library staff
Establish reuse library.
Establish access to library contents
Decide which components to include in reuse library
Manage the library
Notify reusers of changes

Fig. 6.26 Define Reuse Library Personnel Support

Chapter 7

DISCUSSION, CONCLUSION AND FURTHER WORK

7.1 Introduction

Chapter 6 demonstrated some of the major results and features of this research work through the case study example. This chapter summarises the achievements of this research, and also identifies its limitations, whilst providing recommendations about the requirements for further developments and the potential directions for future research in this area. Hence, this chapter aims to critically appraise the research that is exhibited in this thesis.

The main research contributions are related to the combined use of the value net concept with the process pattern approach, as this has enabled a new reuse implementation process to be designed. The combined approach has been demonstrated through the design and implementation of a prototype support system, and the usefulness and limitations of this support system are also considered here. The prototype support system could be developed further to provide a useful consultancy tool for use in consultancy on implementation and extension of reuse programs in manufacturing environment. Recommendations are also given for further work, which could be carried out to improve the support system design and implementation based on the achievements of this research. Suggestions are also provided for the design of a new system, which could handle dynamic events, and dynamically handle changes in requirements for reusable knowledge components and changing status within the organisation.

7.2 Original Contribution

The originality of this work is the design of the Reuse-Driven Manufacturing System Design Process using a combined approach of a reuse value net and reuse process patterns. The value net concept originates from game theory, and a novel aspect of this research is its application to the reuse process. The value net provides a high level view of the key relationships that drive any company's ability to succeed sustainably.

It has also been shown, through examination of past case studies of software reuse projects, that knowledge of all the key factors and their relationships before starting the reuse process can enable more accurate predictions to be made of whether the process will be successful or not. Further, this research has also shown that if the reuse process includes a study of the relationships and interdependencies between different reuse factors, then the problem area(s) for reuse implementation and possible solutions can be identified more clearly. A further contribution of this research is that it has demonstrated that explaining the possible solutions as reuse process patterns, helps the requirements of the possible solutions to be understood more clearly and therefore the tasks required to implement the reuse process be accomplished more easily.

7.3 Summary of research results

The objectives set in section 2.3, have been achieved and the detailed contributions of the research are:

- The research has been tested and proven through the case study example.
- The research has designed a support tool for incorporating reuse in the manufacturing system design process. The prototype tool that has been developed can help the organisation to identify the problems or obstacles that currently exist in their reuse implementation process and the tool can also propose possible solutions to overcome the identified problems. This is achieved by using a reuse knowledge base, by the creation of a Reuse Value Net focussed on the particular needs of the company.
- In this research different reuse factors were identified from the existing literature on reuse. A reuse value net was created to study the relationships between these factors and their interdependencies. The identified relationships and interdependencies were stored and used to build the prototype reuse knowledge base.
- Examples of a reuse process pattern catalogue and a reuse process pattern library have been created during this research. The reuse process patterns have been designed to enable an organisation to put into practice the solutions

suggested by the reuse agent, to overcome the problems identified for reuse implementation.

- The example process patterns can be specialised to suit the requirements and existing processes of a particular organisation. The content of the specialised patterns, particularly for those relating to reuse library production and representation will largely depend on the IT tools and models currently in use with the manufacturing enterprise. For example, reusable elements may be identified from any of the potentially reusable knowledge and information sources discussed in chapter 3.
- A strength of the research reported in this thesis is that it provides a generic process for the analysis and development of reuse programs, and therefore generic process pattern have been provided up to the level that company specific information is required. Company specific information would include details of IT and manufacturing system support tools currently in use, e.g., ERP, PLM, simulation systems etc. as details of potentially reusable components that they hold, and how the components may be accessed, would need to be included in order to tailor a reuse process that meets the specific needs of the organization.

7.4 Limitations of the research

The limitations of the research are as follows:

- The problems identified by the reuse agent are based on the data collected through a questionnaire. Therefore, the problems identified and the recommended solutions are dependant on the accuracy of the information given by the designers and the members of the management. It is therefore necessary to investigate other or additional ways of collecting the information or of verifying the accuracy of the information.

The construction of the reuse value net and the population of the reuse knowledge base is based on the current reuse factors that have been identified in the literature. In future more reuse factors could be identified, and some factors may only be relevant to particular situations or environments. Therefore, the reuse agent should be able to identify what kind of player it is and its relation or dependency with the existing reuse factors. Ideally it should also be able to "learn" from experiences as further reuse projects are undertaken.

7.5 Further work

In this research the reuse agent works as an independent program. The reuse agent does not have any access to the different models of the organisation e.g., product and manufacturing models. The reuse agent could be made more efficient by integrating it with other parts of the organisation's software system, and IT tools. The reuse agent could also be fed the objectives of the different designers. By accessing different models such as product and manufacturing models the reuse agent could then find the reusable components from the existing reuse library, and inform the relevant designers. This would then facilitate adoption and uptake of reuse within the organisation. A Reuse Predictor could then be conceived.

This research has designed a support tool that guides an organisation in the implementation of a reuse driven manufacturing system design process. The two main aspects of the tool are:

- The process designed for identifying obstacles for the reuse process, through the application of the reuse value net.
- The reuse process patterns for guiding the organisation through the different processes involved in implementing reuse.

A prototype tool has been developed in this research, and a number of issues were identified, which are now recognised as further work.

- The data in the reuse knowledge base has been populated based on published experiences and case studies. An analysis of more diverse case studies need to be undertaken in a range of manufacturing industrial settings, in order to enhance the knowledge base and make it more robust. Improved advice could be provided with this enhanced knowledge.

- The support system designed in this research acts independently. It has been observed that the support tool helps the organisation in implementing reuse in the manufacturing system design process. The reuse process patterns also suggest several techniques to encourage the designers and experts to practice reuse. The support tool can be incorporated in the system development process software. The intelligent support tool can then be educated with the requirements of the designers and knowledge of the tasks to be performed by them. The intelligent tool could then find the possible reusable components from the reuse library, and automatically offer them to the designers at the appropriate stages of the design.
- As mentioned earlier the prototype support system could be developed further to provide a useful consultancy tool for use in consultancy on implementation and extension of reuse programs in manufacturing environment. Additional knowledge about the best practices for reuse implementation can be included in the reuse knowledge base. in addition, the support tool should be educated with the project objectives, since more appropriate and specific advice can be given if it is based on the project objectives, current status of the company and the best practices for reuse implementation.

7.6 Conclusion

This study offers a new approach to the implementation of reuse in the manufacturing system design process. In the past, all the factors affecting reuse have been studied separately. This research work contributes to the solution of those difficulties, by enabling all the reuse factors to be studied simultaneously. The reuse factors identified were mapped as reuse value net to study the relationship between them and their interdependencies. This knowledge was then used to design a support tool to help organisations to identify the obstacles for the reuse implementation. Reuse process patterns are then used to guide the organisation through the necessary steps for the implementation of an effective reuse process.

References:

Al-Ahrami A.M.A., Ridgway K.(1999). *An integrated modeling method to support manufacturing system analysis and design.* Computer in Industry, 38. 225-238.

Al Hamando, M. M. S. and S. R. T. Kumura (1994). *Models of Conceptual Design for Concurrent Engineering.* Intelligent systems in design and manufacturing. C. H. Dagli and A. Kusiak. New York, ASME Press: 61-88.

Al-Salka, M. A., Cartmell, M.P., Hardy, S.J., (1998). *A Framework for Generalized Computer-based Support Environment for Conceptual Engineering Design.* Joumal of Engineering Design 9(1): 57-88.

Alting, L, Zhang, H.C, (1989). *Computer aided process planning: the state of the art survey.* International Journal of Production Research 27: 553-85.

Altmeyer, J., Schuermann, B., Ohnsorge, S., (1994). *Reuse of Design Objects in CAD Frameworks.* Proceeding of International Conference of Computer Aided 3esign, San Jose, California..

Altmeyer, J., Schuermann, B., Schuetze M., (1994). *Towards Design Formalization to Support Reuse in ECAD Frameworks.* Proceeding of EDAF94, Porto Alegre, Brazil.

AMR Research, (1999). *Enterprise Resource Planning Software Report 1998-2003.* AMR Research, Boston, MA.

AMR Research, (1999). *Application Spending/Penetration by Vertical Market Report 1999.* AMR Research, Boston, MA.

Ambler, S. (1998). *Process Patterns: Building Large-Scale Systems Using Object Technology.* Cambridge University Press.

Andrel, R., (1997). *Trends in Production Modelling.* 11th International Conference on Engineering Design – ICED'97, Tampere – Finland, Tampere University of Technology, 1:113-120.

Arslan, T.S, Bottaci, L, Taylor, G.E, (1993). *A fault dictionary based expert system for failure diagnosis in a multiple-PCB environment.* Engineering Applications of Artificial Intelligence, 6(5): 447-56.

Askin, R.G., Stanridge, C.R., (1993). *Modeling and Analysis of Manufacturing Systems*, Wiley, New York, NY.

Babuska, R, Verbruggen, H.B, Van Can, H.J.L, (1999). *Fuzzy modeling of enzymatic penicillin-G conversion.* Engineering Applications of Artificial Intelligence, 12 (1): 79-92.

Bahrami Ali.,(1999). *Object-Oriented System Development using the modified language.* The McGraw-Hill Companies, Inc.

Baker, A.D, Parunak, H.V, Erol, K, (1999). *Agents and the Internet: infrastructure for mass customisation,* IEEE Internet Computing, 3(5): 62-9.

Ball, P., (1996). *Introduction to Discrete Event Simulation.* **http://www.dmem.strath.ac.uk/~pball/simulation/simulate.html#Introduction** Presented in 2nd DYCOMANS workshop on *Management and Control : Tools in Action* in the Algarve, Portugal. 15th - 17th May 1996, pp. 367-376 updated in 2001.

Ballarkur, A., Steudel, H. J., (1987). *A within-cell based heuristic for designing cellular manufacturing system.* International Journal of Production Research, 24: 639-65.

Basili, V., Caldierra, G., (1988). Reusing existing software, Computer Science Technical Report Series 27UMIACS-TR 88-72 (1988).

Barbuceanu, M., Fox, M., (1997). *Integrating Communicative Action, Conversations and Decision Theory to Coordinate Agents.* In Proceedings of Autonomous Agents'97, Marina del Rey, CA, 1997. (http://www.ie.utoronto.ca/EIL/ABS-page/ABS-intro.html)

Basu, A., Hyer, N., Shtub, A., (1995). *An expert-system based approach to manufacturing cell design.* International Journal of Production Research, 33(10): 2739-2755.

Batanov, D.B., Nagarur, N., Nitikhunkasem, P., (1993), *EXPERT-MM: a knowledge-based system for maintenance management.* Artificial Intelligence in Engineering, 8: 283-91.

Baya, V.,J. Gevins,C. Baudin,A. Mabogunje,G. Toye, and L. Leifer (1992). *An experimental Study of Design Information Reuse.* 4th International Conference on Design Theory and Methodology, Scottsdale, Arizona, ASME.: 141-147.

Ben-Arieh, D, Chopra, M, (1997). *A case-based NC code generating system for prismatic parts.* International Journal of Production Research, 35(7): 1925-1944.

Benbasat, I., (1989). *Laboratory Experiments in Information Systems Studies with a Focus on Individuals: A Critical Appraisal.* The Information Systems Research Challenge: Experimental Research Methods, Harvard Business School Research Colloquium, Vol. 2,: 33-47.

Bernstein, P., (1996). *Middleware: a model for distributed system services.* Communications of the ACM, 39(2) : 86-98.

Blanchard, B.S., Fabrycky, W.J., (1981). *Systems Engineering and Analysis, Prentice-Hall,* Englewood Cliffs, NJ.

Bloor, M.S., Owen, J., (1991). *CAD/CAM product-data exchange: the nest step.* Computer-aided design, 23: 237-243.

Bender, S., Fish, A., (2000). The transfer of knowledge and the retention of expertise: the continuing need for global assignments. Journal of Knowledge Management 04 (2) (2000) :125-137.

Bernus, P., Nemes, L., Williams, T.J., (1996). *Architectures for Enterprise Integration*, Chapman and Hall, London, 1996.

Bhatnagar, R., and Vishwanathan S., (2000). *Re-engineering global supply chains: alliances between manufacturing firms and global logistics services providers.* International Journal of Physical Distribution & Logistics Management, 30(1): 13–34.

Bhaskara Reddy, S.V, Shunmugam, M.S, Narendran, T.T, (1999). *Operational sequencing in CAPP using genetic algorithms.* International Journal of Production Research, 37(5): 1063-1074.

Bigus, J.P., Bigus, J.,(2001). *Constructing Intelligent Agents with Java.* Wiley computer publishing, John Wiley & Sons., Inc.

Biswas G, Bagchi S and Saad A. (1995). Holonic Planning and Scheduling for Assembly Tasks. TR CIS-95-01, Center for Intelligent Systems, Vanderbilt University, 1995.
http://shogun.vuse.vanderbilt.edu/CIS/IMS/index.html

Bose, A, Gini, M, Riley, D., (1997). *A case-based approach to planar linkage design.* Artificial Intelligence in Engineering, 11(2) :107-119.

Booch, G., (1994). *Object-oriented analysis and design with applications.*, 2nd edition ed, California, The Benjamin/Cumming Publishing Company, Inc., 1994.

Boothroyd, G., Dewhurst, P., Knight, W., (1994). *Product Design for Manufacture and Assembly.* New York, Marcel Drekker, Inc.

Borenstein, D., Becker, J.L.,uiz; Santos, E.R.,(1999). *A systemic and integrated approach to flexible manufacturing systems design.* Integrated Manufacturing Systems, 10(1): 6-14.

Braganza, A., (2002), *Enterprise integration: creating competitive capabilities.* Integrated manufacturing Systems, 13(8): 562-572.

Bravoco R.R., Yadav S.B. (1985). *Requirement Definition Architecture - An Overview.* Computers in Industry, 6: 237-51

Brickley, D., Guha, R. V., (2000). ***Resource Description Framework*** (RDF) Schema Specification 1 0, World Wide Web Consortium, 2000 http://www w3 org/TR/rdf-schema/

Bugtai N.T.,(2002). *Fixturing Information Models in Data Model Driven Product Design and Manufacture.* PhD Thesis Loughborough University.

Bugtai N.T., Young, R.I.M., (1998). *Information models in an integrated fixture decision support tool.* Journal of Material Processing Technology, 76(1-3): 29-35.

Burbidge, J.L., (1984). *Automated production control with a simulation capability.* Proceedings of the IFIP Conference, WG5-7, Copenhagen, :1-14.

Burbidge, J., (1996). *Back to production management.* Manufacturing Engineer, 75(2): 66-71.

Burdea, G.C., (1999). *Invited review: the synergy between virtual reality and robotics.* IEEE Transactions on Robotics and Automation, 15(3): 400-10.

Busby, J. S. (1999). *The Problem with Design Reuse: An Investigation into Outcomes and Antecedents.* Journal of Engineering Design 10(3): 277-296.

Bussmann, S., (19998). *An Agent-Oriented Architecture for Holonic Manufacturing Control.* In Proceedings of First International Workshop on IMS, Switzerland, : 1-12, 1998. (http://www.daimler-benz.com/)

Butler, J., Ohtsubo, H., *ADDYMS: Architecture for Distributed Dynamic Manufacturing Scheduling.* In A Famili, DS Nau, and SH Kim (eds.), Artificial Intelligence Applications in Manufacturing, The AAAI Press, :199-214.

Candido, M.A.B, Khator, S.K, Barcia, R.M., (1998). *A genetic algorithm based procedure for more realistic job shop scheduling problems.* International Journal of Production Research, 36(12) : 3437-3457.

Caprihan, R., Kumar, S., Wadhwa, S., (1997). *Fuzzy systems for control of flexible machines operating under information delays.* International Journal of Production Research, 35(5): 1331-1348.

Catron, B., Ray, S., (1991). *ALPS: A Language for Process Specification.* International Journal of Computer Integrated Manufacturing, Vol. 4, No. 2, 105-113.

Chaharbaghi, K., (1991). *DSSL II: A Powerful Tool for Modelling and Analysing Complex Systems.* International Journal of Operations and Production Management; 11(4): pp..

Chakraborty, T.K, (1992). *A class of single sampling plans based on fuzzy optimisation.* Operations Research, 25(4): 11-20.

Chakraborty, T.K., (1994). *A class of single sampling inspection plans based on possibilistic programming problem.* Fuzzy Sets and Systems, 63(1): 35-43.

Champati, S, Lu, W.F., Lin, A.C., (1996). *Automated operation sequencing in intelligent process planning: a CBR approach.* The International Journal of Advanced Manufacturing Technology, 12 (1): 21-36.

Chan, Y.E., Huff, S.L., 1992, "*Strategy: an information systems research perspective*", Journal of Strategic Information Systems, 4, 4, 191-201.

Chao, P.Y., Wang, Y., (2001). A data exchange framework for networked CAD/CAM. Computers in Industry, 44 : 131-140.

Chen, N, Li, C, Qin, P., 1998, "*KDPAG expert system applied to materials design and manufacture*", Engineering Applications of Artificial Intelligence, 11, 5, 669-74.

Chinnan, R.B., Kolarik, W.J, 1997, "*Neural network-based quality controllers for manufacturing systems*", International Journal of Production Research, 35, 9, 2601-20.

Chon, Y, Kim, K.I., Kim, K, 1993, "*A knowledge-based system for centrifugal fan blade design*", Engineering Applications of Artificial Intelligence, 6, 5, 425-35.

Choudhury, V., 1997, "*Strategic choices in the development of interorganisational information systems*", Information Systems Research, 8, 1, 1-24.

Christensen, H. B., Ron, H., (2000). *A Case Study of Horizontal Reuse in a Project-Driven Organisation.* In Proceedings of 7th Asia-Pacific Software Engineering Conference (APSEC'00), December 05 - 08, 2000 Singapore. pp. 292.

Christensen, J.H., Struger, O.J., Norrie, D., Schaeffer, C.,(1994). *Material Handling Requirements in Holonic Manufacturing Systems.* In Proceedings of the 1994 International Material Handling Research Colloquium, Grand Rapids, MI, The Material Handling Industry of America, pp. 22, 1994. (http://www.automation.rockwell.com/)

Chwelos, P., Benbasat, I., Dexter, A., 2001, "*Research report: empirical test of an EDI adoption model*", Information Systems Research, 12, 3, 304-21.

Clark, G, Mehta, P, Thomson, T, 1992, "*Application of knowledge-based systems to optimised building management maintenance*", Lecture Notes in Artificial Intelligence,, 604, 69-78.

Clarke, S., Elliman, T., Lehaney, B., 2000, "*Re-engineering an information system: a case study in risk reduction*", International Journal of Flexible Manufacturing Systems, 12, 4, 305-20.

Coakes, E., Elliman, T., 1999, "*The role of stakeholders in managing change*", Communications of the Association for Information Systems, http://cais.aisnet.org/articles/default.asp?vol=2&art=4.

Coates, J., 2000, "*Manufacturing in the 21st century*", International Journal of Manufacturing Technology and Management, 1, 1, 42-59.

Coello, J.M.A, DosSantos, R.C., 1999, "*Integrating CBR and heuristic search for learning and reusing solutions in real-time task scheduling*", Lecture Notes in Artificial Intelligence, 1650, 89-103.

Collins' Paperback Dictionary, Harper Collins Publishers, 1995.

Copeland, J. (2000). *What is Artificial Intelligence?* ©Copyright B.J. Copeland, May 2000.

Costa, C.A., (2000). Product Range Models in Injection Mould Tool Design. PhD Thesis. Loughborough University, 2000.

Court, A.W., Culley, S.J., McMohan, C. A., (1995). *Modelling the Information Access Methods of Engineering Designers.* Design Engineering Technical Conferences- DE, AMSE. 83:547-554.

Cutkosky MR, Engelmore RS, Fikes RE, Genesereth MR, Gruber TR, Mark WS, Tenenbaum JM and Weber JC. PACT: An Experiment in Integrating Concurrent Engineering Systems. IEEE Computer, Vol 26, No 1, pp 28-37, 1993. (http://cdr.stanford.edu/PACE/)

Davenport, T., (2000). *ERP: still alive in the Internet age.* Network World, 17, 9, 51.

Delen, D., Benjamin, P., (2003). *Towards a Truly Integrated Enterprise Modeling and Analysis Environment.* Computers In Industry, 51 : 257-268.

Deen, S.M., (1994). *A cooperation framework for holonic interactions in manufacturing.* In Proceedings of the Second International Working Conference on Cooperating Knowledge Based Systems (CKBS'94), Keele University, 1994. (http://www.keele.ac.uk/depts/cs/Research/Dake/home.html)

Dennis, A., Wixom, B. H., (2000). *Systems analysis and Design.* New York. Chichester Wiley c2000.

Deslandres, V., Pierreval, H., (1995). *SYSMIQ - a knowledge-based system for assisting quality-control.* International Journal of Production Research, 33(5) : 1201-1212.

Dimla, S., (1999). *Application of perceptron neural networks to tool-state classification in a metal-turning operation.* Engineering Applications of Artificial Intelligence, 2(4) : 471-477.

Dixon, J. R. (1995). *Knowledge-Based Systems for Design.* Transactions of the ASME 117:11-16.

Dorador, J. M. Young, R.I.M., (1999). *Information Models to Support the interaction between Design For Assembly and Assembly Process Planning.* International Symposium on Assembly and task Planning. Porto, Portugal, :39-44.

Dorador, J.M. and **Young, R.I., (2000).** *'The Application of IDEF0, IDEF3 and UML Methodologies in the Creation of Information Models'.* International Journal of Computer Integrated Manufacture , 13(5) :430-445.

Dori, Dov.(1996). *Object-process analysis of computer integrated manufacturing documentation and inspection functions.* International Journal of Computer Integrated Manufacturing. 9 (5) : 339-353.

Doukidis, G., Poulymenakou, A., Terpsidis, J., Themistocleous, M., Miliotis, P., (1998). *Electronic Commerce: Challenging Issues and the Impact on Employment,* European Union, DG5, Brussels.

Doumeingts, G., (1998). *GIM, Grai Integrated Methodology.* Handbook of Life Cycle Engineering, ,Dordrecht, The Netherlands: Kluwer Academic Publishers, :227-288.

Doumeingts, G., Vallespir, B., Darracar, D., and Roboam, M., (1997). *Design Methodology for Advanced Manufacturing Systems.* Computers in Industry, 9(4): 271-296.

Dowlatshahi, S. (1994a). *A Comparison of Approaches to Concurrent Engineering.* International Journal Advanced Manufacturing Technology 9: 106-113.

Dowlatshahi, S. (1994b). *A Morphological Approach to Product Design in a concurrent Engineering Environment.* International Journal Advanced Manufacturing Technology 9: 324-332.

Drakos, N., (1996). *Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey.* 1996 Copyright © 1996, Computer Based Learning Unit, University of Leeds. http://www.ncst.ernet.in/kbcs/vivek/issues/13.3/weiming/weiming.html

Duffy, A. H. B.,A. Persidis, and K. J. MacCallum (1996). *NODES: a numerical and object based modelling system for conceptual engineering design.* Knowledge-Based Systems 9: 183-206.

Duffy, A. H. B.J. S. Smith, and S. M. Duffy (1998). *Design reuse research: a computational perspective.* Engineering Design Conference'98 - Design Reuse, Brunel, Professional Engineering Publishing Ltd.: 43-56.

Dunn, M., Knight, J.C., (1991). *Software reuse in an industrial setting: a case study.* International Conference on Software Engineering Proceedings of the 13th international conference on Software engineering, Austin, Texas, United States, pp. 329-338, ISBN:0-89791-391-4.

Dym, C.L., Levitt, Raymonds, E.,(1991). *Knowledge-Based Systems in Engineering.*

Eck, J., Marchetti, N., (2000). *Combing e-commerce and EAI.* EAI Journal, January : 42-43.

Edwards, P., Newing, R., (2000). Application Integration for E-business, Business Intelligence 2000, London.

Emmelhainz, M., (1993). *EDI: A Total Management Guide*, Van Nostrand-Reinhold, New York, NY.

ENV 40 003 Computer Integrated Manufacturing—Systems Architecture—Framework for Enterprise Modelling CEN/CENELEC, 1990.

Engelke, H., Grotrian, J., Scheuing, C., Schmackpfeffer, A., Schwarz, W., Solf, B., Tomann, J., (1985). *Integrated manufacturing modelling.* IBM Journal of Research and Development, 29(4) :343-355.

Eskicigolu, H, (1992). *The use of expert systems building tools in process planning.* Engineering Applications of Artificial Intelligence, 5(1) : 33-42.

Esprit Consortium AMICE (Eds.), *CIMOSA: Open System Architecture for CIM*, Springer-Verlag, Berlin, 1993.

Ettl, M., Schwehm, M., (1995). *Determining the optimal partition and kanban allocation in JIT production lines.* Biethahan, J, Evolutionary Algorithms in Management Applications, Springer, Berlin, : 139-52.

European Software Institute: (1995a). *Software Reuse Concise Guide*, ESI-1995-CG001.02, Geoff Norman.

European Software Institute: (1995b). *Technical Report on Reuse*, Rev.1. Vers. 1., John Favaro.

Evbuomwan, N. F. 0.,S. Sivaloganathan, and A. Jebb (1996). *A survey of design philosophies, models, methods and systems.* Proc. Instn. Mech. Engrs., Part B, Journal of Engineering Manufacture 210: 301-320.

Fafchamps, D., (1994). *Organizational factors and reuse.* IEEE Software, September, 31–41.

Feigenbaum, E, A. (1983). *The fifth generation: Artificial Intelligence and Japan's Computer Challenge to the World.* 1983. Reading, Mass: Addison-Wesley.

Ferreira R.J.F., Pradin, B., (1993). *A methodology for cellular manufacturing design.* International Journal of Production Research, 31(1) : 235-50.

Fielder, K, Galletly, J.E, Bicheno, J, (1993). *Expert advice for JIT implementation.* International Operation & Production Management, 13(6) : 23-30.

Filipic, B., Urbancic, T., Krizman, V., (1999). *A combined machine learning and genetic algorithm approach to controller design.* Engineering Applications of Artificial Intelligence, 12, 4, 401-9.

Finger, S. (1998). *Design reuse and design research.* Engineering Design Conference'98 - Design Reuse, Brunel, Professional Engineering Publishing Ltd.

Flynn, D. J., Diaz, O.F.,(1996). *Information Modelling: An International Perspective.* Prentice Hall Europe.

Fothergill, P.,I. Arana, and J. Forster (1996). *Constraint Management and Design Models in Supporting Re-Design.* 6th International Conference on Flexible Automation and Intelligent Manufacturing, Atlanta, Begell House Inc.: 363-372.

Fothergill, P.J. ForsterJ. A. Lacinza,F. Plaza, and I. Arana (1995). *DEKLARE: A Methodological Approach to RE-DESIGN.* Proceedings of Conference on Integration in Manufacturing, Viena.

Fowler, J. E. (1996). *Variant Design for Mechanical Artifacts: A State-of-the-Art Survey.* Engineering with Computers 12: 1-15.

Fox, M.S, Chionglo, J.F., Barbuceanu M., (1993). *The Integrated Supply Chain Management System,* Internal Report, Univ. of Toronto, 1993. http://www.ie.utoronto.ca/EIL/iscm-descr.html

Fox, M.S., Berbuceanu, M., Gruninger, M., (1996). *An organisational ontology for enterprise modelling: Preliminary concepts for linking structure and behavior.* Computers in Industry, 29(3) : 123-134.

Frakes, W.B., Gandel, P.B., (1990). Representing reusable software. *Information and Software Technology* 32 (10) : 653–664.

Frakes, W. B., Biggerstaff, T.J., Prieto-Diaz, R., Matsumura, K., Schaefer, W., (1991). *SOFTWARE REUSE: IS IT DELIVERING?,* International Conference on Software Engineering Proceedings of the 13th international conference on Software engineering, Austin, Texas, United States, pp. 52 – 59, ISBN:0-89791-391-4.

Frakes, W. B., and Isoda, S., (1994). Success factors of systematic reuse. IEEE Software, September, pp.14–19.

Frank, U., (2002). *Multi-Perspective Enterprise Modelling (MEMO)—conceptual framework and modelling languages,* in: Proceedings of the 35th Hawaii International Conference on System Sciences, IEEE Computer Society Press, 2002, : 1–10.

Frank, P.M, Koppen-Seliger, B, (1997). *New developments of AI in fault diagnosis.* Engineering Applications of Artificial Intelligence, 10(1) : 3-14.

Fujikawa, S, Ishii, K, (1995). *Diagnostic expert systems for defect in forged parts.* Journal of Intelligent Manufacturing, 6(3) : 163-173.

Fulcher, J., (1998). *ERP + PDM equals productivity.* Manufacturing Systems, 16(8) : 36-40.

Gao, Y.,I. Zeid, and T. Bardasz (1998). *Characteristics of an effective design plan system to support reuse in case-based mechanical design.* Knowledge-Based System 10: 337-350.

Gascoigne, B., (1995). *PDM: the essential technology for concurrent engineering.* World Class Design to Manufacture; 2(1):38-42.

Gill, A, Bector, C.R, (1997). *A fuzzy linguistic approach to data quantification and construction of distance measures for the part family formulation problem.* International Journal of Production Research, 35(9) : 2565-2578.

Goh, A., Hui, S.C., Song, B., (1996). *An integrated environment for product development using STEP/EXPRESS.* Computers in Industry. 31: 305-313.

Gokhale, A., Schmidt, D.C., Natarajan, B., Wang, N., (2002). *Applying model integrated computing to component middleware and enterprise applications.* Communications of the ACM 45(10) : 65–70.

Gorti, S. R.,A. Gupta,G. J. Kim,R. D. Sriram, and A. Wong (1998). *An object-oriented representation for product and design process.* Computer-Aided Design 30(7): 489-501.

Grabowski, H.,R.S. Lossack, and C. Weis (1996). *Supporting the design process by an integrated knowledge based design system.* Advances in Formal Design Methods for CAD. J. Gero. London, Chapman & Hall,: 209-229.

Griss, M.L.,(1993). *Software reuse: From library to factory.* IBM Systems Journal 32(4): 548-566.

Gruber, T.R. (1993). *A Translation Approach to Portable Ontology Specification.* Knowledge Acquisition **5**: 199-220.

Guiffrida, A.L, Nagi, R, (1998). *Fuzzy set theory applications in production management research: a literature survey.* Journal of Intelligent Manufacturing, 9 (1) : 39-59.

Gunasekera, J.S., Jia, Z., Malas, J.C., Rabelo, L., (1998) *Development of a neural network model for a cold rolling process.* Engineering Applications of Artificial Intelligence, 11(5) : 597-603.

Gupta, Y.P., Gupta M.C., Kumar, A., Sundaram, C., (1995). *Minimizing total intercell and intracell moves in cellular manufacturing: a genetic algorithm approach.* International Journal of Computer Integrated Manufacturing, 8(2) : 92-101.

Gupta, Y., Gupta, M., Kumar, A., Sundaram, C., (1996). *A genetic algorithm-based approach to cell composition and layout design problems.* International Journal of Production Research, 34(2) : 447-82.

Gutierrez, I, Carmona, S, (1995). *Ambiguity in multicriteria quality decisions.* International Journal of Production Economics, 38(2-3) : 2215-2224.

Hamamoto, S, Yih, Y., Salvendy, G., (1999). *Development and validation of genetic algorithm based facility layout - a case study in the pharmaceutical industry.* International Journal of Production Research, 37(4) : 749-768.

Hao, G., Shang, J.S., Vargas, L.G., (1995). *A neural-network model for online control of flexible manufacturing systems.* International Journal of Production Research, 33(10) : 2835-2854.

Harding, J. A., Popplewell, K., (1998). *Simulation: an application of factory design process methodology.* Journal of Operations Research Society, 51, 440–448.

Harding, J. A., Yu, B., Popplewell, K., (1999). *Information modelling: an integration of views of a manufacturing enterprise.* International Journal of Production Research, 37(12), 2777–2792.

Hasegawa, T., Gou, L., Tamura, S., Luh, P.B., Oblak, J.M., (1994). *Holonic Planning and Scheduling Architecture for Manufacturing.* In Proceedings of the 2nd International Working Conference on Cooperating Knowledge-based Systems, 1994.
http://www.toshiba.co.jp/

Hendler, J., McGuinness, D.L., (2000). *The DARPA Agent Markup Language".* IEEE Intelligent Systems : 67-73;

Howard, L., Lewis, H., (2003). *The development of a database system to optimise manufacturing processes during design.* Journal of Materials Processing Technology 134, 374-382.

IFIP-IFAC Task Force, (1997). *GERAM, The Generalised Enterprise Reference Architecture,* Version 1.3, 1997.

Ip, W.L.R., (1998), *A fuzzy basis material removal optimisation strategy for sculptured surface machining using ball-nosed cutters.* International Journal of Production Research, 36(9) : 2553-2571.

Irani, Z., Love, P.E.D., (2002). *Developing a frame of reference for ex-ante IT/IS investment evaluation.* European Journal of Information Systems, 11(1) : 74-82.

Irani, Z., Themistocleous, M., Love, P.E.D., (2003). *The impact of enterprise application integration on information system lifecycles.* Information and Management 41(2) : 177-187.

ISO10303-1, (1994). Industrial automation systems and integration-product data representation and exchange, Part 1. Description Methods: the EXPRESS Language Reference Manual.

ISO 14258, (1996). *Industrial Automation Systems—Concepts and Rules for Enterprise Models,* ISO TC184/SC5/WG1, 1996.

ISO 15704, (1998). *Requirements for Enterprise Reference Architectures and Methodologies,* ISO TC184/SC5/WG1 N423, 1998.

Jennings, N.R., Wooldridge, M.J., (1998). *Applications of Intelligent Agents.* In NR Jennings and MJ Wooldridge (eds.), Agent Technology: Foundations, Applications, and Markets, Springer, : 3-28, http://www.elec.qmw.ac.uk/dai/

Jeon, J., (2000). *Development of a hybrid intelligent maintenance optimisation system.* PhD thesis, University of Salford.

Jochem, R., Mertins, K., Süssenguth, W., (1992). *An object oriented method for integrated enterprise modelling as a basis for enterprise coordination,* in:. C.J. Petrie (Ed.), Enterprise Integration Modelling. Proceedings of the First International Conference, MIT Press, 1992, : 249–258.

Jorysz, H.R., Vernadat, F.B., (1990). *CIM-OSA Part 1: Total enterprise modelling and function view.* International Journal of Computer Integrated Manufacturing 3(3):144-180.

Kalakota, R., Robinson, M., (2001). *E-business 2.0: Roadmap for Success,* Addison-Wesley, Boston, MA.

Kalpic, B., Bernus, P., (2002). *Business process modelling in industry-the powerful tool in enterprise management.* Computers in Industry 47 : 299-318.

Kanagawa, A., Tamaki, F, Ohta, H, (1993). *Control charts for processing average and variability based on linguistics data.* International Journal of Production Research, 31(4) : 913-22.

Kempfer, L., (1998). *Linking PDM to ERP.* Computer-Aided Engineering, 17(10) : 58-64.

Kendall K. E., and Kendall J. E., (1998). *System Analysis And Design.* Upper Saddle River, N. J. : Prentice Hall, 1998.

Kennerley, Mike; Neely, A.,(2001) . "Enterprise resource planning: analysing the impact". Integrated Manufacturing Systems; 12(2):103-113.

Khoo, L.P., Ho, N.C., (1996). *Framework of a fuzzy quality function deployment.* International Journal of Production Research, 34(2) : 299-311.

Kim, T., Kumara, S.R.T., (1997). *Boundary defect recognition using neural networks.* International Journal of Production Research, 35(9) : 2397-2412.

Kim, C.O., Min, H.S, Yih, Y., (1998). *Integration of inductive learning and neural networks for multi-objective FMS scheduling.* International Journal of Production Research, 36(9) : 2497-2509.

Kim, K.K., Umanath, N., (1999). *An empirical investigation of electronic integration in a supply chain relationship.* Proceedings of 20th International Conference on Information Systems, Charlotte.

Kimura, F., (1992). Product and Process Modelling as a Kernel for Virtual Manufacturing Environment. Annals of the CIRP 42(1):147-150.

King, W.R., Teo, T.S.H., (1997). *Integration between business planning and information systems planning: validating a stage hypothesis.* Decision Sciences, 28(2) : 279-308.

Kiritsis, D., (1995). *A review of knowledge-based expert systems for process planning: methods and problems.* International Journal of Advanced Manufacturing Technology, 10 : 240-62.

Koonce, A. David., Judd P. Robert., Parks, M. Charles.(1996). *Manufacturing system engineering and design : an intelligent, multi-model, integration architecture.* International Journal of Computer Integrated Manufacturing 9(6) : 443-453.

Kopacek, P, (1999). *Intelligent manufacturing: present states and future trends.* Journal of Intelligent and Robotic Systems, 26(3-4) : 217-229.

Kosanke, K., (1997) *Enterprise Integration—international consensus: a Europe–USA initiative*, in: K. Kosanke, J.G. Nell (Eds.), ICEIMT 97, International Conference on Enterprise Integration and Modelling Technology, Springer-Verlag, Torino, Italy, October 1997, : 64–74.

Kosanke, K., Nell, N. G., (1999). *Standardisation in ISO for enterprise engineering and integration.* Computers in Industry, 40(2-3) :311-319.

Kosanke, K., Vernadat F., Zelm, M., (1999). *CIMOSA: enterprise engineering and integration.* Computers in Industry 40 : 83–97.

Kouvelis, P., (1992). *Design and planning problems in flexible manufacturing systems: a critical review.* Journal of Intelligent Manufacturing, 3(2) : 75-99.

Kovacs, G. L., Kopácsi, S., Nacsa, J., Haidegger, G., Groumpos, P.,(1999). *Application of software reuse and object-oriented methodologies for the modelling and control of manufacturing systems.* Computers in Industry, 39(3) : 177-189

Krause, F.-L.,F. KimuraJ. Kjelberg, and S. C.-Y. Lu (1993). *Product Modelling.* Annals of the CIRP 42(2): 695-705.

Kubicek, H., (1992). *The organization gap in large-scale EDI systems.* Streng, R.J., Ekering, C.F., van Heck, E., Schultz, J.F.H., Scientific Research on EDI "Bringing Worlds Together", Samsom, Amsterdam, : 11-41.

Kumar, R., Midha, P.S., (2001). *A QFD based methodology for evaluating a company's PDM requirements for collaborative product development.* Industrial Management & Data Systems; 101(3): 16-132.

Kunnathur, Anand S; Sundararaghavan, P.S (2004).; Sampath, Sriram"Dynamic rescheduling using a simulation-based expert system",Journal of Manufacturing Technology Management; 15(2), 199-212.

Kuo, T., Mital, A., (1993). *Quality control expert systems; a review of pertinent literature.* Journal of Intelligent Manufacturing, 4(4) : 245-257.

Kusiak, A., Park, K., (1990). *Concurrent design: decomposition of design activities.* Proceedings of the 2nd International Conference on Computer Integrated Manufacturing, IEEE, Troy, NY, : 557-63.

Kusiak, A, Lee, H, (1996). *Neural computing-based design of components for cellular manufacturing.* International Journal of Production Research, 34(7) : 1777-1790.

Labib, A.W., Williams, G.B., O'Conner, R.F., (1997). *An intelligence model (system): an application of A.H.P. and fuzzy logic rule-based controller.* Journal of the Operational Research Society, 49 : 745-757.

Lahti, A., Mantyla, M., Ranta, M., (1997). *Capturing and Deploying Design Decisions. Product Modeling for Computer Integrated Design and Manufacturing.* M. J. Pratt, Sriram, R. D. and Wozny, M. J. London, Chapman & Hall: 3-16.

Lam, F.S., Lin, B.C., Sriskandarajah, C, Yan, H., (1997). *Scheduling to minimize product design time using a genetic algorithm.* International Journal of Production Research, 37(6) : 1369-1386.

Lardner, J.F., (1988). *Integration and information in an automated factory.* Proceedings, SME Autofact 6 Conference, Anaheim, CA, : 34-44.

Larman C.,(1997). *Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design,* Prentice Hall PTR, Upper Saddle River, New Jersey 07458.

Law Hang-Wai., Tam Hon-Yuen,(2000). *Object-oriented analysis and design of computer aided process planning systems.* International Journal of Computer Integrated Manufacturing. 13(1) : 40-49.

Lederer, A.L., Sethi, V., (1996). *Key prescriptions for strategic information systems planning.* Journal of Management Information Systems, 13(2) : 35-62.

Lee, J., Gruninger, M., Jin, Y., Malone, T., Tate, A., Yost G., (1998). et al, *The PIF Process Interchange Format and Framework.* Knowledge Engineering Review , 13(1) : 91-120.

Lee, C.Y., Sha, W., Zhou, X., (1999). *Total quick response system: a flexible management approach for global operations.* International Journal of Management, 16(3) : 299-310.

Lee, K.,Kang, K.C., Chae, W., Choi, B.W.,(2000). *Feature-based approach to object-oriented engineering of applications for reuse.* Software-Practice and Experience, 30(9) : 1025-1046.

Lee, Choong Y., (2003). Total manufacturing information system: a conceptual model of a strategic tool for competitive advantage Integrated Manufacturing Systems. 14(2) : 114-122.

Lim, W. C., (1994). *Effects of reuse on quality, productivity, and economics.* IEEE Software, September, : 23–31.

Lin, GY-J, Solberg, J.J., (1992). *Integrated Shop Floor Control Using Autonomous Agents.* IIE Transactions: Design and Manufacturing, 24(3) : 57-71. http://IE.www.ecn.purdue.edu/IE/

Lin, E., Minis, I., Nau, D.S, Regli, W.C, (1995). *Contribution to virtual manufacturing background research.* http://www.isr.umd.edu/Labs/CIM/vm/report/report.html.

Ling, C.Y., (2001). *Model of factor influences on electronic commerce adoption and diffusion in small and medium sized enterprises.* Proceedings of ECIS 2001, Bled, Slovenia.

Linthicum, D.S., (2000). *Enterprise Application Integration.* Addison-Wesley, Longman, Reading, MA.

Linthicum, D.S., (2001). B2B Application Integration: E-business - Enable Your Enterprise, Addison-Wesley Longman, Boston, MA.

Lu, S.C.Y.,, Subramanyam, S., (1988). *A computer-based environment for simultaneous product and process design.* Anjanappa, M., Anand, D.K., Advances in Manufacturing System Engineering, 31.

Mackulak, G.T.,(1985). *An Examination of the IDEF Approach Used as a Potential Industry Standard for Production Control System Design.* Automated Manufacturing STP- 862, :136-149.

Madison, D.E., Wu, C.T., (1987). *An expert system interface and data requirement for integrated product design and manufacturing process* Proceedings, 3rd International Conference on Data Engineering, Los Angeles, CA, 3-5 February, : 610-618.

Mak, K.L., Lau, H.Y.K., (2000). *An object-oriented specification of a flexible manufacturing cell.* International Journal of Operations and Production Management, 20(5) : 534-548.

Malek, M., Toitgans, M.P, Wybo, J.L, Vincent, M, (1998). *An operator support system based on case-based reasoning for the plastic moulding injection process*", Lecture Notes in Artificial Intelligence, 1488 : 402-413.

Markus, M.L., Tanniru, M., Van Fenema, P.C., (2000). *Multisite ERP implementations.* Communications of the ACM, 43, 4, 42-46.

Mason-Jones, R; Berry, D; Naim, M.M.(1998). "A systems engineering approach to manufacturing systems analysis " *Integrated Manufacturing Systems*; 9(6), pp. 350-365.

Maturana, F., Norrie, D., (1996). *Multi-Agent Mediator Architecture for Distributed manufacturing.* Journal of Intelligent Manufacturing, 7 : 257-270, 1996. http://imsg.enme.ucalgary.ca/

McClure, C., (1997) *Software Reuse Techniques: Adding Reuse to the System Development Process (*Prentice Hall).

McEleney, B., O'Hare, GMP., Sampson, J., (1998). *An Agent Based System for Reducing Changeover Delays in a Job-Shop Factory Environment.* In Proc. of PAAM'98, London, 1998. http://www.co.umist.ac.uk/

McKay, A., Bloor, M.S., A. de Pennington (1996). *A Framework for Product Data.* IEEE Transactions on Knowledge and Data Engineering 8(5): 825-837.

Mertins K. ,Jochem R. , Jakel F.-W. ., (1997). *A tool for object-oriented modelling and analysis of business processes.* Computer in Industry. 33 : 345-356.

Meziane,F., Vadera, S., Kobbacy, K., Proudlove, N., (2000). *Intelligent systems in manufacturing: current developments and future prospects.* Integrated Manufacturing Systems, 11(4) : 218-238.

Miller, E., (2000). *ERP expands to serve engineering.* Computer-Aided Engineering, 19(7) : 34-6.

Min, L., Cheng, W., (1999). *A genetic algorithm for minimizing the makespan in the case of scheduling identical parallel machines.* Artificial Intelligence in Engineering, 13(4) : 399-403.

Minsky M. (1985). *The Society of Mind.* Touchtone Book, Simon & Schuster.

Mitra, S., Basu, P., Ghoshal, T.K., (1993). *Classification and representation of design knowledge for circulating fluidized bed boiler.* Engineering Applications of Artificial Intelligence, 6(5) : 457-464.

Molina, A,(1995a). *A Manufacturing Model to Sport Data-Driven Applications for Design and Manufacture,* PhD Thesis, (Loughborough University, UK) .

Molina, A., Al-Ashaab, A.H., Ellis, J.I.A., Young, R.I.M., Bell, R., (1995b). *Review of Computer-Aided Simultaneous Engineering Systems.* Research Engineering Design 7:38-63.

Molina, A., Bell, R.,(2002). *Reference models for the computer aided support of simultaneous engineering,* International Journal of Computer Integrated Manufacturing, 15(3) : 193-213

Murgatroyd, C., Clements, P., Roberts, S., Sumters, C., Weston, R., (1994). *An information modelling methodology applied to a manufacturing environment.* Advanced in Manufacturing Technology VIII , Proceeding of the tenth national conference on manufacturing research, Taylor and Francis Ltd. 1994.

Nakayasu, H, Nakagawa, M, Ishikawa, K, Nakamachi, E, Nakamura, Y, Katayama, T, 1999, "*Design of die geometry for metal sheet forming in virtual manufacturing*", International Journal of Industrial Engineering - Theory Applications and Practice, 6(4) : 271-281.

Nalebuff, B. J., Brandenburger, A. M., (1996). *Co-opetition: 1) A Revolutionary Mindset that Combines Competition and Cooperation 2) The Game Theory Strategy that's Changing the Game of Business* (Harper Collins Business).

Nevins, J.L., Whitney, D.E., (1989). in Defazio, T.L., Edsall, A.C. *Concurrent Design of Products and Processes.*

Nissen, M., (2000). *Supply chain process and agent design for e-commerce.* Proceedings of 33rd Hawaii International Conference on System Sciences, Hawaii, IEEE Computer Society Press, Los Alamitos, CA.

Noble, J.S., Tanchoco, J.M.A., (1993). *Design for economics.* In Kusiak, A., Concurrent Engineering: Automation, Tools, and Techniques, John Wiley & Sons, New York, NY.

Nonaka, I. (1991). *The knowledge creating company.* Harvard Business Review 69. (1991):

Nonaka, H. Takeuchi, (1995). *The Knowledge--Creating Company: How Japanese Companies Create the Dynamics of Innovation.* Oxford University Press, New York, 1995.

Noy & McGuiness (2000). "Ontology Development 101: A Guide to Creating Your First Ontology". http://www.ksl.stanford.edu/people/dlm/papers/ontology-tutorial-noy-mcguinness-abstract.html

Nwana H.S. (1996). *Software Agents: An Overview.* Knowledgė Engineering Review 1(3):205-244.

Obank, A., Leaney, P., Roberts, S., (1995). *Data management within a manufacturing organization.* Integrated Manufacturing Systems, 6(3) : 37-43.

Oliver, R.W., (1999). *ERP is dead! Long live ERP!* Management Review, 12.

Ong, N.S, Khoo, L.P, (1999). *Genetic algorithm approach in PCB assembly.* Integrated Manufacturing Systems, 10(5) : 256-65.

Open Systems Group, (1987). *Systems Behaviour*, Paul Chapman Publishing, London.

Orlicky, J., (1975). *Materials Requirement Planning*, McGraw-Hill, New York, NY.

Ormerod, T. C., Mariani, J., Spiers, G., (1997). *Supporting the process of re-use in innovative design environments.* IEEE Colloquium, London, IEE. 16: 9/1-9/3.

Ortega, G, Giron-Sierra, J.M, (1998). *Geno-fuzzy control in autonomous servicing of a space station.* Engineering Applications of Artificial Intelligence, 11(3) : 383-400.

Ortiz, A., Lario F., Ros, L., (1999). *Enterprise integration—business processes integrated management: a proposal for a methodology to develop enterprise integration programs.* Computers in Industry 40 : 155–171.

Oestereich, B., (1999). *Developing Software with UML: Object-Oriented Analysis and Design in Practice.* Addison-Wesley.

Ouelhadj, D., Hanachi, C., Bouzouia, B., (2000). *A Multi-agent architecture for distributed monitoring in flexible manufacturing systems.* In the Proceedings of the IEEE International Conference on Robotics and Automation (ICRA'2000), pp. 120-126, California, USA.

Ouelhadj, D., Hanachi, C., Bouzouia, B., (1999). *A Multi-contract net protocol for dynamic scheduling in manufacturing systems.* In the Proceedings the IEEE International Conference on Robotics and Automation (ICRA'1999), pp. 1114-1120, Detroit, USA.

Pahl, G., Beitz. W., (1996). *Engineering Design: A Systematic Approach* . Grat Britain, Springer-Verlag London Ltd.

Pande, S.S, Desai, V.S, (1995). *Expert CAPP system for single spindle automates.* International Journal of Production Research, 33(3) : 819-833.

Parnaby, J., (1979). *Concept of a manufacturing system.* International Journal of Production Research, 17(2) : 123-135.

Parnaby, J., (1991). *Designing effective organisations.* International Journal of Technology Management, 6, (1- 2) : 15-31.

Parnaby, J., (1994). *Business process systems engineering.* International Journal of Technology Management, 9, (3- 4) : 497-508.

Parunak V.D., Baker A., and Clark S.,(1998). *The AARIA Agent Architecture: From Manufacturing Requirements to Agent-Based System Design.* In Working Notes of the Agent-Based Manufacturing Workshop, Minneapolis, 1998. http://www.aaria.uc.edu/

Peng Y, Finin T, Labrou Y, Chu B, Long J, Tolone WJ and Boughannam A. A Multi-Agent System for Enterprise Integration. In Proceedings of PAAM'98, London, 1998. http://www.cs.umbc.edu/lait/research/ciimplex/

Pierre, S., (1993). *Application of artificial intelligence techniques to computer network topology design.* Engineering Applications of Artificial Intelligence, 6(5) : 465-72.

Plossl, G.W., (1985). *Production and Inventory Control: Principles and Techniques,* 2nd ed., Prentice-Hall, Englewood Cliffs, NJ.

Polany, M., (1958). Personal Knowledge, University of Chicago Press, Chicago, 1958.

Polany, M., (1966). Tacit Dimension, Doubleday, New York, 1966.

Prieto-Diaz, R., (1991). *Making Software Reuse Work: An Implementation Model.* ACM SIGSift Software Engineering Notes, Vol. 16, No. 3, pp.

Puschmann, T., Alt, R., (2001). *Enterprise application integration - the case of the Robert Bosch Group.* Proceedings of the 34th Hawaii International Conference on System Sciences.

Puschmann T., Alt, R., (2004). *Enterprise application integration systems and architecture- the case of the Robert Bosch Group.* The Journial of Enterprise Information Management, 17(2) : 105-116.

Rabbino, H., *Applying the Principles of Co-opetition With System Dynamics Tools.* SDSG, LLC (The Strategic Decision Simulation Group), http://www.sdsg.com/

Ransing, R.S, Lewis, R.W., (1997). *A semantically constrained neural network for manufacturing diagnosis.* International Journal of Production Research, 35(9) : 2639-2660.

Rao Ming, Wang Qun ,Cha Jianzhong,(1993). *Integrated Distributed Intelligent Systems in Manufacturing.* Chapman & Hall.

Rao, H.A, Gu, P, (1995). *A multi-constraint neural-network for the pragmatic design of cellular manufacturing systems.* International Journal of Production Research, 33(4) : 1049-1070.

Rine, C. D., And Nada, N., (2000). *Software reuse reference model: development and validation.* Journal of Information and Software Technology, 42 : 47–65.

Ring, K., Ward-Dutton, N., (1999). *Enterprise Application Integration: Making the Right Connections*, Ovum, London.

Rising, L., (2002) Reuse at AG Communication Systems = Patterns, http://www.agcs.com/supportv2/techpapers/patterns/papers/multi.htm (first published in MultiUseExpress, 4(3), June 1996, p. 3).

Rowland, J.G, Jain, L.C, (1993). *Knowledge-based systems for instrumentation diagnosis, system configuration and circuit and system design.* Engineering Applications of Artificial Intelligence, 6(5) : 437-46.

Ruh, W., Maginnis, F., Brown, W., (2000). Enterprise Application Integration: A Wiley Tech Brief, John Wiley & Sons, New York, NY.

Rzeveski Goerge, (1997). A framework for designing intelligent manufacturing systems. Computers in Industry. 34 : 211-219.

Sambamurthy, V., Venkataraman, S., DeSanctis, G., (1993). *The design of information technology planning systems for varying organizational contexts.* European Journal of Information Systems, 2(1) : 23-35.

Sanchez, R., Heene, A., (1996). *A competence perspective on strategic learning and knowledge management.* Strategic Learning and Knowledge Management, eds. R. Sanchez, A. Heene, Chichester, England, John Wiley & Sons, NA: 3-18.

Sarkis, Li L.,(1994). *An IDEF0 Functional Planning Model for the Strategic Implementation of CIM Systems,* International Journal of Computer Integrated Manufacturing. 7(2), :100-115.

Scheer, A.W.,(1999). *ARIS—Business Process Modelling.* second ed., Springer-Verlag, Berlin, 1999.

Schlenoff C, Gruninger M, Tissot F, Valois J, Lubell J, Lee J. , (2003). *The Process Specification Language (PSL) Overview and Version 1.0 Specification.* [Web Accessed 3rd October 2003]. http://ats.nist.gov/psl

Schnitger, M., (2003). *Digital Simulation to meet today's product development challenge.* White paper – DARATECH. http://www.ugs.com/products/nx/docs/wp_daratech_ford.pdf

Shahin, T. M., Andrews, T.J., Sivloganathan, S., (1999). *A design reuse system.* Proc. Instn. Mech. Engrs., Pat B, Journal of Engineering Manufacture 213:621-627.

Shen, W and Norrie, D. H., (1999). *Agent-based systems for intelligent manufacturing: A state-of-the-art survey.* Knowledge and Information System, an International Journal, 1(2), :129–156.

Shen, W., Norrie, D.H., (1998). ***An Agent-Based Approach for Dynamic Manufacturing Scheduling.*** In Working Notes of the Agent-Based Manufacturing Workshop, Minneapolis, 1998.
http://ksi.cpsc.ucalgary.ca/KSI/KSI.html
http://imsg.enme.ucalgary.ca/

Shen, W., Xue, D., Norrie, D.H., (1998). *An Agent-Based Manufacturing Enterprise Infrastructure for Distributed Integrated Intelligent Manufacturing Systems.* In Proceedings of PAAM'98, London, 1998. http://imsg.enme.ucalgary.ca/

Shen W, Maturana F and Norrie DH. (1998b). *Learning in Agent-Based Manufacturing Systems.* In Proceedings of AI & Manufacturing Research Planning Workshop, Albuquerque, The AAAI Press, pp. 177-183, 1998. http://imsg.enme.ucalgary.ca/

Shen, W. and Barthès J. P., (1996). Computer Supported Cooperative Environments for Engineering Design: A Survey. Tech. Report 96-122, CNRS UMR Heudiasyc, Université de Technologie de Compiègne, France, 1996. http://imsg.enme.ucalgary.ca/CIG/papers/csceed.pdf

Shingo, S., (1988). *Non-Stock Production: The Shingo System for Continuous Improvement.* Productivity Press, Cambridge, MA.

Shipley, M.F, De Korvin, A, Omer, K, (1996). *A fuzzy logic approach for determining expected values: a project management application.* Journal of the Operational Research Society, 47(4) : 562-9.

Shivathaya, S.S, Fang, X.D., (1999). *Fuzzy-logic-based ranking of steel grades generated by material design KBS.* Engineering Applications of Artificial Intelligence, 12(4) : 503-512.

Shtub, A., (1989). *Modelling group technology cell formation as a generalized assignment problem.* International Journal of Production Research, 27 : 775-782.

Sivaloganathan, S., And Shahin, T. M., (1999). *Design reuse: an overview.* Proceedings of the IMECH E Part B Journal of Engineering Manufacture, 213(7) : 641-654.

Skyrme, D., Amidon, D.,(1997). *The knowledge agenda.* Journal of Knowledge Management 01 (1) (1997) :27-37.

Smith, L.N., Midha, P.S, (1999). *A knowledge based system for optimum and concurrent design, and manufacture by powder metallurgy technology.* International Journal of Production Research, 37(1) : 125-137.

Sofranec, D., (2001). *Make the connection.* Computer-Aided Engineering, 20(9) : 39-42.

Sohlenius, G., (1992). *Concurrent Engineering.* Annals of the CIRP 41(2) :645-655.

Soliman, F., Youssef, M., (1998). *The role of SAP software in business process re-engineering.* An International Journal of Operations and Production Management; 18(9) : 886-895.

Sousa, P., Ramos, C., Neves, J., (2003). *The Fabricare cheduling prototype suit: Agent interaction and knowledge base.* Journal of intelligent manufacturing, 14: 441-455.

Spano, M.R., O'Grady, P.J., Young, R.E., (1991). *The design of flexible manufacturing systems,* Computers in Industry, 21(2) : 185-198.

Su, C.T, Chang, C.A, Tien, F.C, (1995). *Neural networks for precision measurement in computer vision systems.* Computers in Industry, 27 : 225-236.

Suh, N.P., (1984). *The future of the factory.* Robotics and Computer Integrated Manufacturing, 1(1) : 39-49.

Suh, N.P., (1995). *Axiomatic Design of Mechanical Systems.* Transactions of the ASME 117 :2-10.

Sun, S.H, Chen, J.L, (1995). *A modular fixture design system based on case-based reasoning.* International Journal of Advanced Manufacturing Technology, 10(6) 389-395.

Sung, C.S, Choung, Y.I., (1999). *A neural network approach for batching decisions in wafer fabrication.* International Journal of Production Research, 37(13) : 3101-3114.

Suresh, N.C, Slomp, J, Kaparthi, S, (1999). *Sequence-dependent clustering of parts and machines: a fuzzy ART neural network approach.* International Journal of Production Research, 37(12) : 2793-2816.

Suri, R., (1988). *A new perspective on manufacturing system analysis.* Design, Analysis of Integrated Manufacturing Systems, Ann Arbor, MI, : 8-15.

Szelke, E, Markus, G.A, (1997). *Learning reactive scheduler using CBR/L.* Computers in Industry, 33(1): 31-46.

Syan, C.S.,(1994). *Introduction to Concurrent Engineering.* Concepts, implementation and practices. C.S. Syan and U. Menon, Chapman and Hall:3-23.

Talavage, J., Hannam, F.G., (1988). *Flexible manufacturing systems in practice.* Marcel Dekker, New York, NY.

Tan, D.S., Uijttenbrock, A.A., (1997). *Information infrastructure management: a new role for IS managers.* Information Systems Management, 14(4) : 33-41.

Thacker, R.M., (1989). *New CIM Model: A Blueprint for the Computer Integrated Manufacturing Enterprise.* Society of Manufacturing Engineers, Michigan.

Themistocleous, M., (2002). *Evaluating the adoption of enterprise application integration in multinational organisations.* PhD thesis, Brunel University, London. In Themistocleous, M., Irani, Z., (2001a). *Benchmarking the benefits and barriers of application integration.* Benchmarking: An International Journal, 8(4): 317-31.

Themistocleous, M., Irani, Z., (2001a). *Benchmarking the benefits and barriers of application integration.* Benchmarking: An International Journal, 8(4): 317-31.

Themistocleous, M., Irani, Z., (2001b). *Evaluating application integration: an exploratory case study.* Strong, D., Straub, D., Proceedings of Seventh Americas Conference on Information Systems, (AMCIS 2001), Boston, MA, : 1376-1380.

Tichem, M., Storm, T., (1997). *Designer Support for product structuring-development of a DFx tool within the design coordination framework.* Computers in Industry 33:155-163.

Tien, F.C, Chang, C.A, (1999). *Using neural networks for 3D measurement in stereo-vision inspection systems.* International Journal of Production Research, 37(9) : 1935-1948

Tirpak, T., (2000). *Design-to-manufacturing information management for electronics assembly.* International Journal of Flexible Manufacturing Systems, 12 (2- 3) : 189-205.

Toh, K.T.K., (1999). *The realization of reference enterprise modelling architectures.* International Journal of Computer Integrated Manufacturing. 12(5), 403-417.

Tompkins, J. A., Reed, R., (1976). *An applied model for facilities design problem.* International Journal of Production Research, 14 : 583-595.

Turban, E., Aronson, J.E., (2002). *Decision Support Systems and Intelligent Systems.* Prentice-Hall, Englewood Cliffs, NJ, 2002.

Uschold, M, King, M., Moralee, S., Zorgios Y., et al (1998). *The Enterprise Ontology.* The Knowledge Engineering Review, 13(1) : 31-89.

Udo, Godwin J; Ehie, Ike C.,(1996). *Advanced manufacturing technologies: Determinants of implementation success.* International Journal of Operations and Production Management, 16 (12) : 6-26.

Ulrich, K. T., Eppinger (2003). *Product Design and Development.* Boston, Irwin, McGraw-Hill, 2003.

Urban, S.D., Ayyaswamy, K., Fu, L., Shah, J.I., Liang, J., (1999). *Integrated product data environment: data sharing across diverse engineering applications.* International Journal of Computer Integrated Manufacturing, 12(6): 525-540.

Van Everdingen, Y., Van Hillegersberg, J., Waarts, E., (2000). *ERP adoption by European midsize companies.* Communications of the ACM, 43(4) : 27-31.

Venkatachalam, A.R, Mellichamp, J.M., Miller, D.M., (1993). *Automated design for manufacturability through expert systems approaches.* Concurrent Engineering – Contemporary Issues and Modern Design Tools. H.R. Parsaei and W. G. Sullivan. Cambridge, Chapman & Hall:426-446.

Vergeest, J.S.M., (1991). *CAD surface data exchange using STEP.* Computer-aided design, 23(4) : 269-281.

Vernadat, F., (1993). *CIMOSA: enterprise modelling and Enterprise Integration using a process-based approach,* in: H. Yoshikawa, J. Goossenaerts (Eds.), Information Infrastructure Systems for Manufacturing, North-Holland, Amsterdam, 1993, : 65–84.

Vernadat, F., (1996). *Enterprise Modelling and Integration: principles and applications.* Chapman & Hall, 1996.

Vernadat, F., (1997) *Enterprise modelling languages,* in: K. Kosanke, J.G. Nell (Eds.), ICEIMT 97, International Conference on Enterprise Integration and Modelling Technology, Springer-Verlag, Torino, Italy, October 1997, : 212–224.

Vincenti, W.G., (1990). *What engineers know and how they know it- analytical studies form aeronautical history.* Baltimore and London, John Hopkins, 1990.

Vishnupad, P., (1996). *Neuro-fuzzy control of complex manufacturing systems.* International Journal of Production Research, 34(12) : 3291-309.

Vollmann, T.E., Berry, W.L., Whybark, D.C., (1997). *Manufacturing Planning and Control Systems,* 4th ed., McGraw-Hill, New York, NY.

Wallace, T.F., 1990, *MRP II: Making It Happen.* The Implementers' Guide to Success with Manufacturing Resource Planning, 2nd ed., Oliver Wight, Essex Junction, VT.

Wang, R.C, Chen, C.H, (1995). *Economic statistical np-control chart designs based on fuzzy optimisation.* International Journal of Quality and Reliability Management, 12(1) : 82-92.

Wang, C., Huang, S.Z, (1997). *A refined flexible inspection method for identifying surface flaws using the skeleton and neural network.* International Journal of Production Research, 35(9) : 2493-507.

Webster, S., Jog, D., Gupta, A., (1998). *A genetic algorithm for scheduling job families on a single machine with arbitrary earliness/tardiness penalties and an unrestricted common due date.* International Journal of Production Research, 36(9) : 2543-51.

Weiming Shen, Douglas H. Norrie and Jean-Paul A., (2001). *Multi-Agent Systems for Concurrent Intelligent Design and Manufacturing.* Barthes First published 2001 by Taylor & Francis New Fetter Lane, London EC4P 4EE

West, T.M., Randhawa, S.U., Brings, S.D., (1989). *The role of product design in the evaluation of new manufacturing technologies.* Proceedings, 1989 International Industrial Engineering Conference, Toronto, Canada, : 14-17.

Willcocks, L., Sykes, R., 2000. *The role of the CIO and IT function in ERP.* Communications of the ACM, 43(4) : 32-38.

Williams, T.J., (1994) . *The Purdue enterprise reference architecture.* Computers in Industry 24 (2–3) : 141–158.

Williams, T.J.,(1994). *The Purdue Enterprise Reference Architecture.* Computers in Industry 24 (2-3) : 141–158.

Williams, T.J.,(1997). *PERA Methodology: I.* International Workshop In Business Integration, Valencia, Spain, July 1997.

Wolfe, S., (1996). *Workgroup vs. enterprise-wide PDM.* Computer Graphics World.

Wong, T.N, Siu, S.L, (1995). *A knowledge-based approach to automated machining process selection and sequencing.* International Journal of Production Research, 33(12) : 3465-84.

Wooldridge MJ., Jennings, N.R., (1995). *Intelligent agents: Theory and practice.* The knowledge Engineering Review, 10(2) : 115-152.

Wortmann, J.C., (1998). *Evolution of ERP systems.* Bititci, U.S., Carrie, A.S., Strategic Management of the Manufacturing Value Chain, Kluwer Academic, Boston, MA. : 11-23.

Wu, G.S.K., (1989). *SGML theory and practices.* London: British Library Research and Development Department, 1989.

Wu, B., (1992). *Manufacturing System Design and Analysis.* Chapman & Hall, London.

Wysk, R.A., Yang, N., Joshi, S., (1994). *Resolution of deadlocks in flexible manufacturing systems: avoidance and recovery approaches.* Journal of Manufacturing Systems, 13(2) : 128-138.

Xia, Q.J, Rao, M., (1999). *Dynamic case-based reasoning for process operation support systems.* Engineering Applications of Artificial Intelligence, 12(3). : 343-61.

Yeomans, R.W., Choudry, A.,, Ten Hagen P.J.W., (1985). *Design Rules for a CIM System,* Elsevier Science, Amsterdam.

Yongting, C., (1996). *Fuzzy quality and analysis on fuzzy probability.* Fuzzy Sets and Systems, 83(2) : 283-90.

Young, R.I.M., Caliglieri-Jnr, O, Costa, C.A., (1998). *Information Interactions in Data Model Driven Design for Manufacturer.* Globalisation of Manufacturing in the Digital Communications Era of the 21st Century: Innovation, Agility, and the Virtual Enterprise. G. Jacucci, G. J. Olling, K. Preiss and M. Wonzy. Trento, Kluwer Academic Publishers. 1998 : 313-324.

Yu, B., Harding J. A., Poppelwell K.,(1999). *Information modelling: an integration of views of a manufacturing enterprise.* International Journal of Production Research, 37(12). : 2777-2792.

Yu, B., Harding J. A., Poppelwell K.,(2000). A reusable enterprise model. International Journal of Operations & Production Management, 20(1) : 50–69.

Zhang, H.C, Huang, S.H., (1995). *Application of neural networks in manufacturing: a state of the art.* International Journal of Production Research, 33(3) :705-728.

Zhang, F., Jardine, A.K.S., (1997). *A smart maintenance decision system.* Proceedings of the European Conference on Intelligent Management Systems Operations, : 79-86.

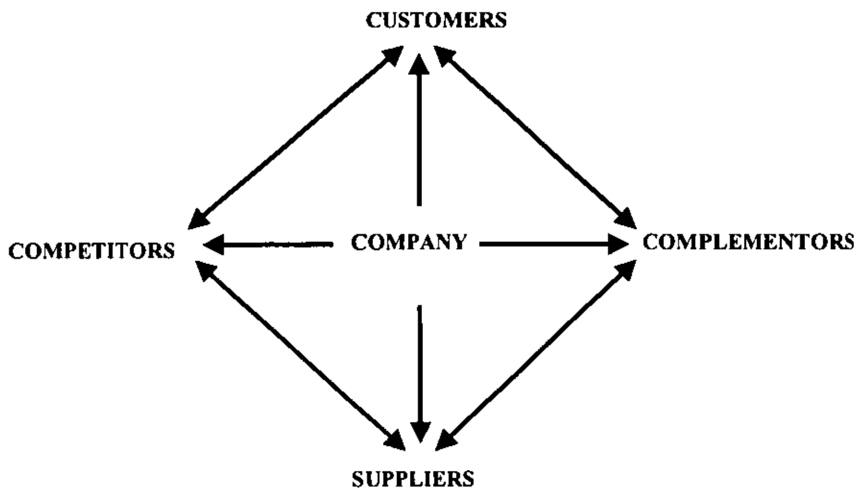
Zeid, I., Gupta, S.M., Bardasz, T., (1997). *A case-based reasoning approach to planning for disassembly.* Journal of Intelligent Manufacturing, 8(2): 97-106.

Appendix 1. –

The Value Net Theory

The Value Net

The game of business is all about creating and capturing value (Brandenburger, Nalebuf, 1995). The Value Net is a schematic map designed to represent all the players in the game and the interdependencies among them. Interactions take place along two dimensions. Along the vertical dimension are the company's customers and suppliers. Along the horizontal dimensions are the players with whom the company interacts but does not transact. They are its *competitors* and *complementors*.



Identifying the Competitors and the Complementors in reference to Customers:

- A player is a Complementor when customers value your product more when it is enhanced with the complementor's product than with your product alone.
- A player is a Competitor when customers value your product less when they have the competitor's product than when they have your product alone.

Identifying the Competitors and the Complementors in reference to Suppliers:

- A player is a Complementor when it's more attractive for a supplier to provide resources to you and the other player than when it only supplies you alone.
- A player is a Competitor if it is less attractive for a supplier to provide resources to you and the other player than when it's supplying you alone.

Two fundamental symmetries:

- Customers and suppliers play symmetric roles and both are equal partners in creating value.
- Competitors and Complementors play mirror-image roles.

The Value Net describes the various roles of the players. It is possible for the same player to occupy more than one role simultaneously. Drawing the Value Net for your business is the first step toward changing the game.

The second step is identifying all the elements of the game. According to game theory, there are five: players, added values, rules, tactics and scope or PARTS. These five elements fully describe all interactions and to change the game you have to change one or more of these elements.

The PARTS Strategy

PLAYERS

- Heisenberg Principle:
- ➢ You can't interact with a system without changing it.
- ♦ In business, it is the way you change a game by joining it.
- To become a player, you have to pay to play.
- Bringing in Complementors, and in some cases, even bringing in competitors can be beneficial.
- If you have a low added value, identify who stands to gain by your entry into the game.
- ➢ Those players may be willing to pay you to play.

ADDED VALUES

- The added value of a monopoly can severely limit the added value of competitors.
- You can have higher quality or lower costs, but accomplishing both is rare.
- ➢ Make intelligent trade-offs.
- Create a relationship with your customers.
- ➢ Create loyalty by rewarding it.

- Imitation:
- ➢ With Proven Strategies: everyone can do it, everyone can make money at it, and everyone can continue making money at it.
- Increase your added value and limit the other players' added value while preventing them from limiting yours.

RULES

- You, your customers, and your suppliers are partners in creating value.
- Contracts with customers and suppliers can positively or negatively affect your transactions with them.

- Contracts with customers
- ➢ When customers press you for price concessions, that is competition, not cooperation.
- ➢ The overall game is co-opetition.

- Contracts with suppliers
- ➢ When suppliers try to raise prices, that is competition.
- ➢ Create co-opetition.

- Mass-Market rules
- ➢ When sellers don't negotiate, buyers can't negotiate.

- Government rules
  - ➢ Government can make rules of the game through laws that govern transactions among all the players in the economy.

- Changing the rules
  - ➢ Both the significance of rules and the opportunities to change them are often under appreciated.
  - ♦ Do not blindly follow rules
  - ♦ You can change the rules, but remember: other people can change the rules, too; do not assume your rules will rule.

TACTICS

- Perceptions
  - ➢ Fundamental element of any game
  - ➢ Play a central role in negotiations.

- Tactics
  - ➢ Actions that players take to shape the perceptions of other players.
- Convince people that you have the goods
  - ➢ The Credibility Test

1. If you have the goods, put your money where your mouth is.
   - ♦ Accept a pay-for-performance contract
   - ♦ Offer a guarantee
   - ♦ Give free trials
   - ♦ Advertise
2. What you don't do sends a signal, too.
3. Ask other people to demonstrate their credibility to you:
   - ♦ Propose a pay-for-performance contract
   - ♦ Ask for a guarantee
   - ♦ Request a free trial

- Maintain a good perception
  - ➢ Hiding information
  - ♦ Having made a favourable impression, people try to preserve it by:

1. Burying projects they have turned down
2. Following the herd
3. Creating reasons to fail.

➢ Negotiating in a Fog

♦ Mistakes:

1. Revealing the minimum you need. You risk getting exactly that, and no more. Posturing is no solution; that risks deadlock.
2. Making threats explicit. Even is a threat is already implicit, making it explicit changes perceptions. There's no going back.
3. Trying to resolve differences of opinion between you and the other party. This is hard to do, and possibly counterproductive.

♦ Solutions:

1. Establish a settlement escrow to promote good faith negotiating by both parties.
2. Bring in a mediator to help the other party understand the consequences of non agreement.
3. Recognize what you and the other party do—and don't—have to agree on. Use differences of opinion to structure win-win deals.

- "If you can't convince'em, confuse'em." - Harry Truman

SCOPE

- Every game is linked to other games – there are no real boundaries.
- There is really one "big" game, which extends across space, over time, down generations.
- ➢ Any two games, even if conceived of as games in their own right, are really only components of the big game.
- Create a dilemma for the incumbent.
- Lengths of contracts can control scope.
- Links through tactics
- ➢ Threats and promises are the classic examples of creating a perceptual linkage.
- ♦ Designed to persuade other people to do something – or not do something – based on how you say you will respond.
- ➢ Set a precedent
- ♦ Take action in a game today to convince other people of what you can and will do tomorrow.
- There is always a larger game in which all other games take place within its context.

A useful checklist of questions that a business should ask itself, following the 'PARTS' framework outlined in Co-opetition is as follows:

Players Questions

- Have you written out the Value Net for your organization, taking care to make the list of players as complete as possible?
- What are the opportunities for cooperation and competition in your relationships with customers and suppliers, competitors and complementors?
- Would you like to change the cast of players? In particular, what new players would you like to bring into the game?
- Who stands to gain if you become a player in a game? Who stands to lose?

Added Values Questions

- What is your added value?
- How can you increase your added value? In particular, can you create loyal customers and suppliers?
- What are the added values of the other players in the game?
- Is it in your interest to limit their added values?

Rules Questions

- Which rules are helping you? Which are hurting you?
- What new rules would you like to have? In particular, what contracts do you want to write with your customers and suppliers?
- Do you have the power to make these rules? Does someone else have the power to overturn them?

Tactics Questions

- How do other players perceive the game? How do these perceptions affect the play of the game?
- Which perceptions would you like to preserve? Which perceptions would you like to change?
- Do you want the game to be transparent or opaque?

Scope Questions

- What is the current scope of the game? Do you want to change it?
- Do you want to link the game to other games?
- Do you want to de-link the game from other games?" (p.257. 258)

Being Ready for Change

- The unexamined game is not worth playing
- By viewing from a game theory perspective, you will stop taking many features of your business for granted.
- You do not have to accept the game you find yourself in.
- Changing the game is an ongoing process.
- There is no end to the game of changing the game.
- Game theory allows players to avoid missing win-win opportunities.

No game is an island. Games are linked across space and over time. You can expand or shrink the scope of a game. You can expand it by creating linkages to other games, or you can shrink it by severing linkages. Either approach may work to your benefit.

Changing the game is hard, since there are many potential traps. The Value Net, PARTS and coopetition are designed to help managers recognise and avoid these traps.

The first trap is think you have to accept the game your in. It's more rewarding to be a game maker than a game taker. The next trap is to think that changing the game must come at the expense of others. The coopetition mind-set of looking for both win-win and win-lose strategies is more rewarding. Another trap is to think that you have to find something to do that others can't. Being unique is not a prerequisite for success.

References:

Brandenburger, A. and Nalebuf, B., (1995). *The Right Game: Use Game Theory to Shape Strategy,* Harvard Business Review. July-August, pp. 57-71.

Brandenburger, A. and Nalebuf, B., (1997). *Co-opetition.* Doubleday, New York, 1997.

Book summary Co-opetition. http://www.bizsum.com/co-opetition.htm

Appendix 2-

Paper published in Int. J. of Computer Integrated Manufacturing

INT. J. COMPUTER INTEGRATED MANUFACTURING, APRIL–MAY 2004, VOL. 17, NO. 3, 185–194

Managing reuse in manufacturing system modelling and design: a value net approach

SHILPA S. DANI and JENNY A. HARDING

Abstract. In the past, reuse programs have assumed that technical solutions would overcome barriers to effective reuse. However, recent retrospectives of reuse programs show that organizational factors can greatly affect the implementation of a reuse program. Reuse is often approached as an independent collection of tools and techniques, and the technical and non-technical aspects are commonly examined separately. This research proposes that all the factors affecting reuse are interdependent and therefore should be studied simultaneously. It is therefore very important that all elements and relationships are identified and documented in a structured, clear manner. The concept of value nets, based on game theory has been identified as a means of capturing the different factors involved in the reuse driven software process. The value net is a useful model for viewing multiple interactions from various perspectives. The value net of the reuse-driven software development process is a map of the various factors and players identified in the reuse process, and the interactions occurring between the players. Hence the reuse value net helps in identifying the important factors that can provide the points of leverage to make the reuse process more efficient.

1. Introduction

Manufacturing system design or redesign generally includes some elements of modelling and simulation, so that the current system and processes can be better understood, and the changes, or new system and processes, can be evaluated before expensive modifications are implemented (Harding and Popplewell 1998, Harding *et al.* 1999). Building models of all types is an expensive process as it takes considerable time and effort fully to analyse and understand the systems and processes that form an essential part of the operation of an enterprise, but this knowledge must be gained before appropriate enterprise or simulation models may be built. Models must also be appropriate to meet the objectives of the project and tasks that they have been built to support. It can therefore be difficult to reuse existing models for other projects, particularly if different personnel are working on the projects. Unfortunately, a large (and expensive) amount of time is generally needed to re-examine all the possible existing models, and modify them as required, hence models are often used for a project and then rejected (and wasted) as new models are built for the next project. This is a waste of information and resources, as some elements of the existing models may well be appropriate for future use. When changes must be made within the manufacturing system, the application of an existing design or parts of an existing successful design may help to reduce the resources needed for original design and analysis. It will also help to avoid the error and uncertainty that accompany all human activities, and design or development in particular. It also helps maximize the familiarity of production staff with the selected design and helps clients to maintain consistent ways of using and maintaining the result (Busby 1999).

The research reported in this paper examines ways of reusing existing information, models and software in manufacturing system design. As this is a fairly new area of research, the longer established area of software reuse was initially studied to identify similarities and lessons, which may benefit the current work. Software reuse—broadly defined as the use of engineering knowledge or artefacts from existing systems to build new ones—is a technology for improving software quality and productivity (Frakes and Isoda 1994). Software Reuse is the process of creating software systems from predefined software components (McClure 1997). Reuse is not a new strategy, reusing an existing design in a new application is an obvious way of reducing effort and risk—not just in the design

Authors: S. S. Dani and J. A. Harding, Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, Loughborough, Leicestershire, LE11 3TU, UK. Email: J.A.Harding@lboro.ac.uk

International Journal of Computer Integrated Manufacturing
ISSN 0951-192X print/ISSN 1362-3052 online © 2004 Taylor & Francis Ltd
http://www.tandf.co.uk/journals
DOI: 10.1080/09511920310001607122

activity itself, but also in downstream functions. The potential of reuse has been identified by many researchers and research groups with differing motivations (Sivaloganathan and Shahin 1999). In the past, the different factors involved in a reuse process have been identified and studied separately by different research groups. However, the identification of the factors is not sufficient in itself to arrive at a decision about the requirements of an efficient reuse process. All the factors involved in the reuse process play different roles in the reuse process. The authors believe that all the reuse factors are interdependent and hence should be studied simultaneously to achieve an efficient reuse process. A study of the interdependencies between the factors provides knowledge of how a particular factor is affecting the reuse process. To achieve this, the factors have to be depicted in a fashion that makes it easier to analyse the reuse process. In this research, the concept of a value net has been identified as a means of studying the reuse factors. The value net of the reuse driven manufacturing system software development process is a map of the various factors and players identified in the reuse process, and the interactions occurring between the players.

Many companies are not achieving their full potential for reuse, some have had disappointing results from their reuse efforts, but many others are still avoiding reuse altogether. However, the problem is not the lack of technologies to support reuse. The problem arises when organizations approach reuse as an independent collection of tools and techniques, or when an organization focuses purely on the technical issues of reuse without adequately addressing the non-technical issues (Rine and Nada 2000). Too often in the past, organizations have treated reuse solely as a technical problem and neglected the welter of other critical factors. Recent retrospectives of reuse programs show that organizational factors can greatly affect the implementation of a reuse program. More recently, many of these issues have been discussed in the literature and at several workshops and conferences. The role of each factor in the reuse process needs to be studied in parallel with the relationship between these factors. This paper presents a novel approach of using Value Nets in order to design an efficient software reuse process.

2. The value net

An enterprise generates wealth by operating a business, which produces products or services for a market (Yu *et al.* 2000). In today's business context the imperatives of cost efficiency and customer responsiveness have driven firms to pursue two common business strategies aggressively—globalization (i.e. the worldwide distribution of production and associated facilities) and time-based competition. Both these strategies have dramatically transformed the way in which business activities are organized and carried out. Globalization is motivated by pressures like cost-effectiveness, access to new markets and economies of scale (Bhatnagar and Viswanathan 2000). It has led to the emergence of borderless organizations with globally distributed suppliers, production and related facilities. Enterprises are under continuous competitive pressure in the market place. However, all businesses cannot be considered to be competitors, there are businesses that complement too. There are also products or services, which provide complementary rather than competing products and services. The Complementors are those who cooperate to capture market share (Nalebuff and Brandenburger 1996). There are four entities affecting a business that should be considered when developing a business strategy, i.e. the suppliers, customers, competitors and complementors. Nalebuff and Brandenburger introduced a schematic map of a business called 'The Value Net'. The concept of a value net is based on Game Theory, where business is considered as a game. The Value Net is a map of the game of business. The Value Net describes all the players and analyses the elements of competition and cooperation between them. In other words, the value net locates all the various players relative to one another and identifies the interdependencies between them. The research reported here applies the concept of the value net to the reuse process.

In its simplest state, a value net can be visually represented by a diamond, with a company in the centre. The value net can be graphically portrayed as the interactions between four players, i.e. customers, suppliers, competitors and complementors and the company. In the Value Net, along the vertical dimension are the company's *customers* and *suppliers*. Resources such as raw materials and labour flow from the suppliers to the company, and products and services flow from the company to its customers. Along the horizontal dimension of the Value Net are the company's *competitors* and *complementors*. Complementors are third parties who can lend added value to a firm by supplementing its products and services. For example, a computer manufacturer views a software company as a complementor, because customers place greater value on a computer with an operating system installed than on either the computer or the software alone. The Value Net is a high level view of the key relationships that drive any company's ability to succeed sustainably (Rabbino). Figure 1 shows a schematic map

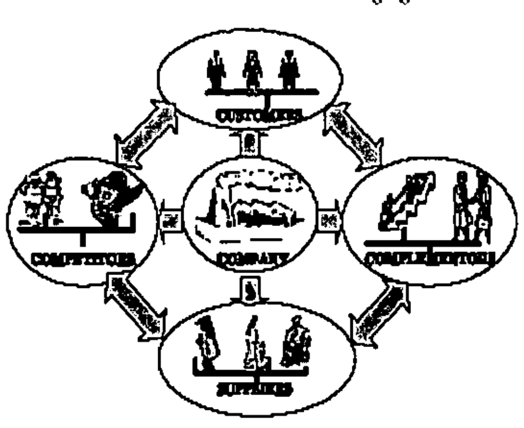


Figure 1. The Value Net (adopted from Nalebuff and Brandenburger 1996).

of the whole game or The Value Net, as presented by Nalebuff and Brandenburger (1996).

3. Applying value net theory to the reuse driven software process

The value net above is initially drawn from the company's point of view. But the concept extends beyond this single viewpoint, as there are customers' customers, suppliers' suppliers, and so on. The value net can either be extended to represent these extended values, or separate value nets can be drawn for suppliers, customers, etc. In the context of the present research, different reuse factors must be grouped because of their important interdependencies. We therefore now examine the roles of the players, customers, suppliers, complementors and competitors in forming a reuse value net.

3.1. *The suppliers and customers*

In the Value Net, suppliers and the customers appear on the vertical dimension, as the customers play symmetric roles with the suppliers. They are equal partners in creating value. Experts and Designers fulfil these roles in the Reuse Value Net. Reuse is about the production and consumption of reuse (Falchamps 1994). A producer is a creator of reusable work products, and the consumer is someone who uses them to create other software (Lim 1994). In the Reuse Value Net, system experts are the 'suppliers' who 'supply' or provide their expert knowledge for reuse. Management also plays a role as a supplier as it provides the funding, resources and additional support environments that are required to enable the reuse process. Designers have two roles, both as a 'supplier' and as a 'customer'. The designers may design a component that is subsequently available for reuse. Therefore, in this context, a designer is an expert, who uses his skills to produce a reusable product (hence designers are suppliers). A designer may also reuse a component designed by another designer or take advantage of expert knowledge on reuse (hence designers are also customers). The Organization is a customer as it wishes to gain the financial and operational benefits that can result from efficient reuse.

The value net shown in figure 2 summarizes this range of customers and suppliers.

3.2. *The competitors and complementors*

On the horizontal dimension of the value net, there is another symmetry, i.e. the competitors and the complementors (figure 3). *A competitor provides the competition to a product and to meet and overcome this competition the quality of the product must be constantly improved* (Nalebuff and Brandenburger 1996). The traditional design methods, which do not promote reuse, are the competitors to the reuse-driven process. One of the reasons for the designers using design processes, which do not involve reuse, is the Not-Invented-Here (NIH) Syndrome (Ambler 1998). The designers prefer the traditional design methods, which do not promote reuse, and hence are the competitors to the reuse-driven process. Hence, NIH competes against reuse. On the other hand, the members of the organization will be better motivated to practice reuse when there are reuse libraries/catalogues where they can easily find the reusable components. Hence the reuse libraries are the complementors in the Reuse Value Net. However, it is not sufficient to have something to reuse, since designers will not reuse existing components if they feel it is quicker or easier to 'design from scratch', than to determine whether an existing component is appropriate for their application. The reuse libraries should therefore be complemented by the knowledge of how to reuse the available components, and by techniques to simplify the reuse of these components.

3.3. *The Reuse Value Net*

The previous discussions and figures show that the factors affecting reuse can be mapped using a Reuse Value Net that is based on the concept of a value net as described in section 2. The completed Value Net for

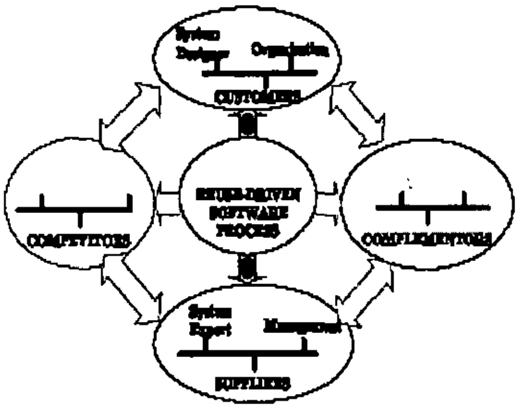


Figure 2. The suppliers and customers in the Reuse Value Net.

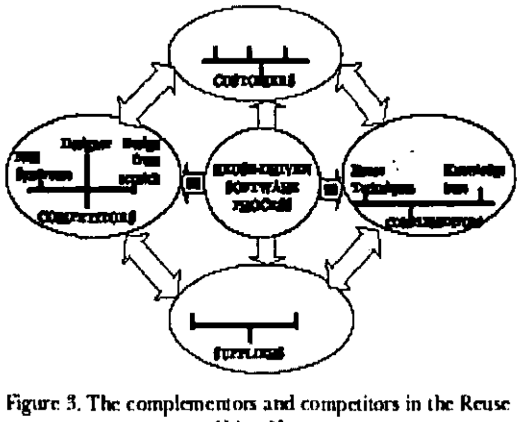


Figure 3. The complementors and competitors in the Reuse Value Net.

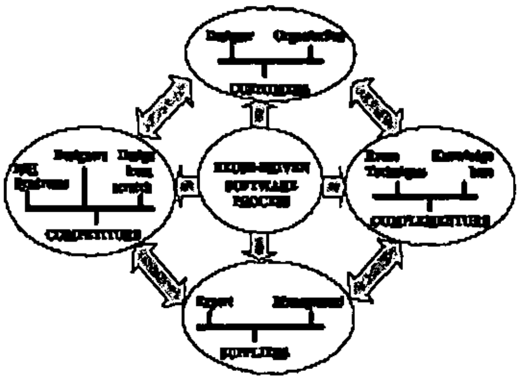


Figure 4. Reuse Value Net

Reuse-Driven Software Process can be drawn as shown in figure 4.

The Reuse Value Net in figure 4 shows that experts, designers and management are the *suppliers* of knowledge and funding to the reuse process. Similarly, the designers and organization are the *customers* as the designers are reusing the expert knowledge and the organization is getting benefit from savings that are made as a result of the reuse process. The competitors to the reuse process are alternative design methods, such as design from scratch, and preconceptions or existing prejudices of personnel involved in the process, such as the NIH syndrome, as this thinking restricts the designer, preventing him or her from taking advantage of reusable elements. Finally, the reuse library, knowledge bases, and various reuse tools and techniques are complementors as the reuse library provides the reusable components. The knowledge bases give the knowledge of what to reuse and the reuse techniques provide the knowledge of how both to reuse existing components and extend the library as existing components are adapted or new components are created.

The above explanation describes the relationships within the reuse process between the four players. However, these players should not only be examined individually, as they are related, e.g. the designers (potential customers) may be adversely influenced by Not-Invented-Syndrome (competitor). To win this customer from the competitor and convert him or her over to reuse, training (which is a complementor) can be given to the designers about the advantages of the reuse process; for example, the designers can be educated about how to reuse the reusable components. Designers will also be better motivated if a reuse library (complementor) is well organized, and the components are easy to find and reuse, and if it is straightforward to determine which reusable components meet the requirements of a particular specification. Furthermore, management can create competition among the designers, or motivate the designers initially by rewarding their efforts, both in making use of reusable components and in making new components easier to reuse. The relationships of each actor with other actors in the value net need to be studied in detail. This information can then be used to find the points of leverage. The points of leverage may be different for different companies. The reuse value net provides all the necessary information to implement a reuse process, and hence can be used as a good analysis tool.

4. Research methodology

To attempt reuse successfully, an organization must systematically address a wide range of both technical and non-technical problems. There is no guaranteed recipe for successful reuse, each organization must analyse its own needs, implement reuse measurements, define the key benefits it expects, and manage risk (Frakes and Isoda 1994). It is, however, worth the efforts, as companies that make a better job of understanding their domains and implementing systematic reuse will gain a powerful competitive advantage. Instituting reuse may require an organization to change the ways in which it conducts its business, and such changes may bring many previously hidden managerial, cultural, social and economical issues to the surface (Davis 1994). Without a reuse-driven software development process to guide them, system developers will not achieve many of the benefits of sharing and reuse. One of the main objectives of this research is to design an efficient reuse process. To achieve these objectives a Reuse agent; an intelligent software program (tool) has been designed.

Figure 5 shows the experimental environment. The experimental environment consists of a knowledge base, which is populated by knowledge about different scenarios in reuse implementation and solutions to overcome the difficulties that can arise during implementation. Before designing a reuse process for an organisation the Reuse Agent needs to collect information about the design and development environment that currently exists in the organization, and identify how developers (design engineers) and management currently view and practise reuse within this environment. To collect this information the Reuse Agent uses two questionnaires, one questionnaire seeks information about management support for reuse in the organization, and the second questionnaire seeks information about the present reuse scenario and the designer viewpoints about reuse. Each questionnaire is designed to collect information relating to the present training facilities regarding reuse and the tools to encourage or support reuse that may currently be

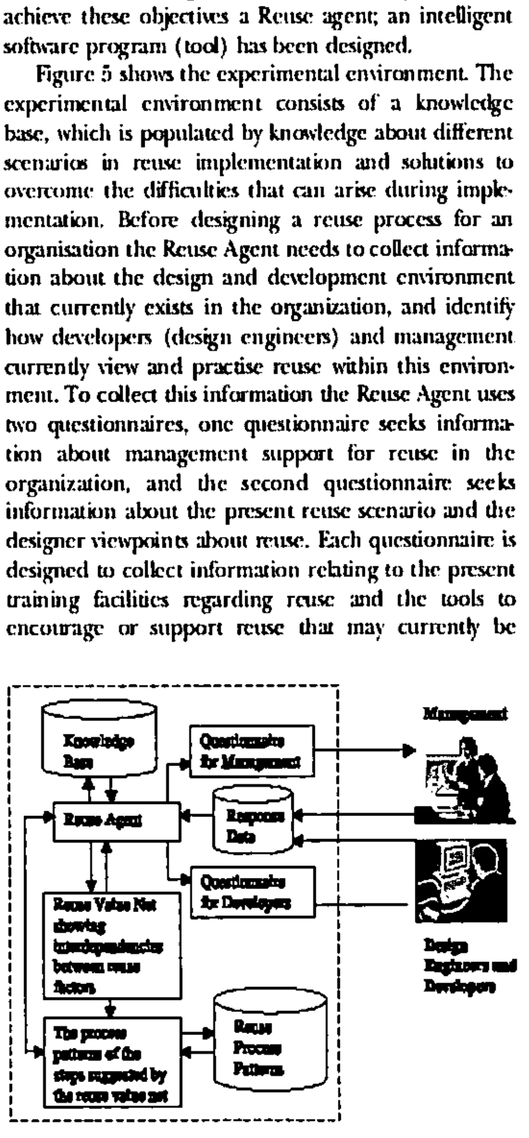

Knowledge Base
Questionnaire for Management
Management
Reuse Agent
Response Data
Questionnaire for Developers
Reuse Value Net showing interdependencies between reuse factors
Design Engineers and Developers
The process pattern of the steps suggested by the reuse value net
Reuse Process Pattern

Figure 5. Experimental set-up.

available. The questionnaire response information from management and developers is collected in a database (labelled *Response Data* in figure 5) and is then fed to the Reuse Agent. The Reuse Agent then processes the information using its knowledge base and uses it to formulate a Reuse Value Net for the organization.

The reuse value net helps to identify relationships between different reuse factors and any factors that are inhibiting the reuse implementation process. Knowledge of the interdependencies between the reuse factors, coupled with information from the knowledge base and example reuse processes help the Reuse Agent to formulate a reuse process pattern for the organization.

The reuse value net identifies the 'what', i.e. what should be done to achieve efficient reuse and the 'who', i.e. the factors and actors involved in the reuse process. However, to achieve efficient reuse, the 'when' and 'how' of practising reuse must also be known. In the current research, patterns have been identified as a means to answer these questions. Patterns capture knowledge that experts apply in solving recurring problems (Rising 2002). One common definition is that a pattern is a solution to a problem in a context. Hence, in the context of the current research, the expertise of an experienced designer is being reused when a pattern is applied. Two pattern approaches have been combined, i.e. process pattern and design pattern approaches. The combined use of process and design patterns in this approach to reuse is described in greater detail in Dani and Harding (2003). The *process patterns* are used to guide the designer about how the reuse process works, and a *design pattern* provides a scheme for refining the subsystems or components of a software system, or the relationships between them. The design pattern describes the commonly recurring structure of communicating components that solve a general design problem within a particular context. The design patterns are used to store the reusable models for future reuse.

A process pattern is a collection of general techniques, actions, and/or tasks (activities) for developing reusable designs, models or software components. A process is defined as a series of actions in which one or more inputs are used to produce one or more outputs. A pattern is a general solution to a common problem or issue, one from which a specific solution may be derived. In other words, a process pattern is a pattern that describes a proven, successful approach and/or series of actions for developing reusable components. An important feature of a process pattern is that it describes what you should do but not the exact details of how you should do it.

When applied together in an organised manner, process patterns can be used to construct a software or information reuse process for a particular organization.

Figure 5 also illustrates the use of Process Patterns along with the Reuse Value Net. The Reuse Process Patterns are stored in the reuse library. The complementors are the reusable tools, which in this case are the design pattern and process pattern catalogues. These can then be invoked in order to find the relevant reusable models for the particular reuse project. The Reuse Value Net can also be used to identify the points of leverage. The use of the Reuse Value Net and process patterns to guide developers in the reuse process is explained through an example in the next section.

5. An example to illustrate the experimental software

Company X is a global manufacturer of widgets from several product families. When a new product is designed the company uses simulation models to design suitable manufacturing cells to produce the product, and meet its throughput, cost and delivery objectives. It is a time consuming and expensive task to build and evaluate new simulation models. The company therefore wishes to introduce a Reuse Project to improve its reuse of existing information and simulation models.

The Reuse Agent is used to determine appropriate activities during the Reuse Project. Questionnaires are sent to members of management and design/modelling teams, to establish whether any reuse activities currently exist in the company. The completed questionnaires are collected and the response data stored in the database. Company X has never formally practised reuse in the past. However, the questionnaires are designed also to identify whether individual design engineers are reusing components informally; this may be by using elements of models they have previously built, or even working informally with colleagues and sharing components.

The Reuse Agent processes the response data, and in this case, the analysis of the questionnaires identifies that

(1) The designers are not currently practising reuse, they are designing systems from scratch.
(2) When they work on a new project, they generally do not try to reuse code or models from previous projects.
(3) Most of the designers have never tried to reuse code or models created by other designers.
(4) There are no formal incentive programs or mechanisms to motivate reuse either in place or planned.
(5) No reuse library or reuse catalogue currently exists in the organization.
(6) The only sharing across business units that currently exists is of software applications.

The Reuse Agent analyses these results and determines that the following problem areas need to be addressed initially:

(i) The designers are using traditional design methods. (These act as an obstacle to reuse.)
(ii) Even if the designers want to practise reuse there is nothing to reuse.
(iii) Training in reuse techniques should be provided.

These are highlighted in figure 6, which shows the value net that the Reuse Agent forms for the organization.

In the analysis for the reuse value net it was found that the design engineers in the company use traditional design methods. This is an obstacle to reuse since the engineers are likely to feel that, as what they are already doing produces satisfactory results, why should they change their approach? The adoption of any new techniques or tools inevitably causes a certain amount of pain through the learning period, before benefits are really felt. Hence, Company X will need to provide some motivation and incentives to promote the different ways of working required to make reuse effective. Reuse is not currently actively used in Company X. Answers from the questionnaires suggested that when individual designers begin a new simulation model, they may sometimes study previous models they have worked on for ideas. However, a major obstacle to reuse was identified as there are no easily available, convenient components that can be reused. To solve the identified problems, the Reuse Agent recommends the creation of a Reuse Library and Catalogue to store the reusable components, and the adoption of different reuse techniques to encourage the design engineers in the company to reuse these components. To support its recommendations, the Reuse Agent then consults the Reuse Process Patterns database (shown in figure 5) and suggests activities that need to be undertaken to progress the reuse project.

The role and behaviour of the Reuse Agent in this example are summarized in figure 7. The experimental set up (from figure 5) is shown in the top left of figure

7. As explained, the Reuse Agent initially uses its knowledge to analyse the questionnaire response data

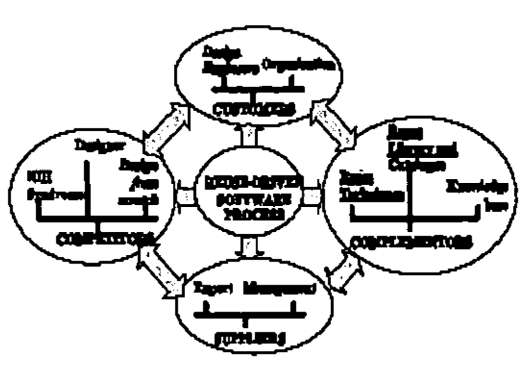
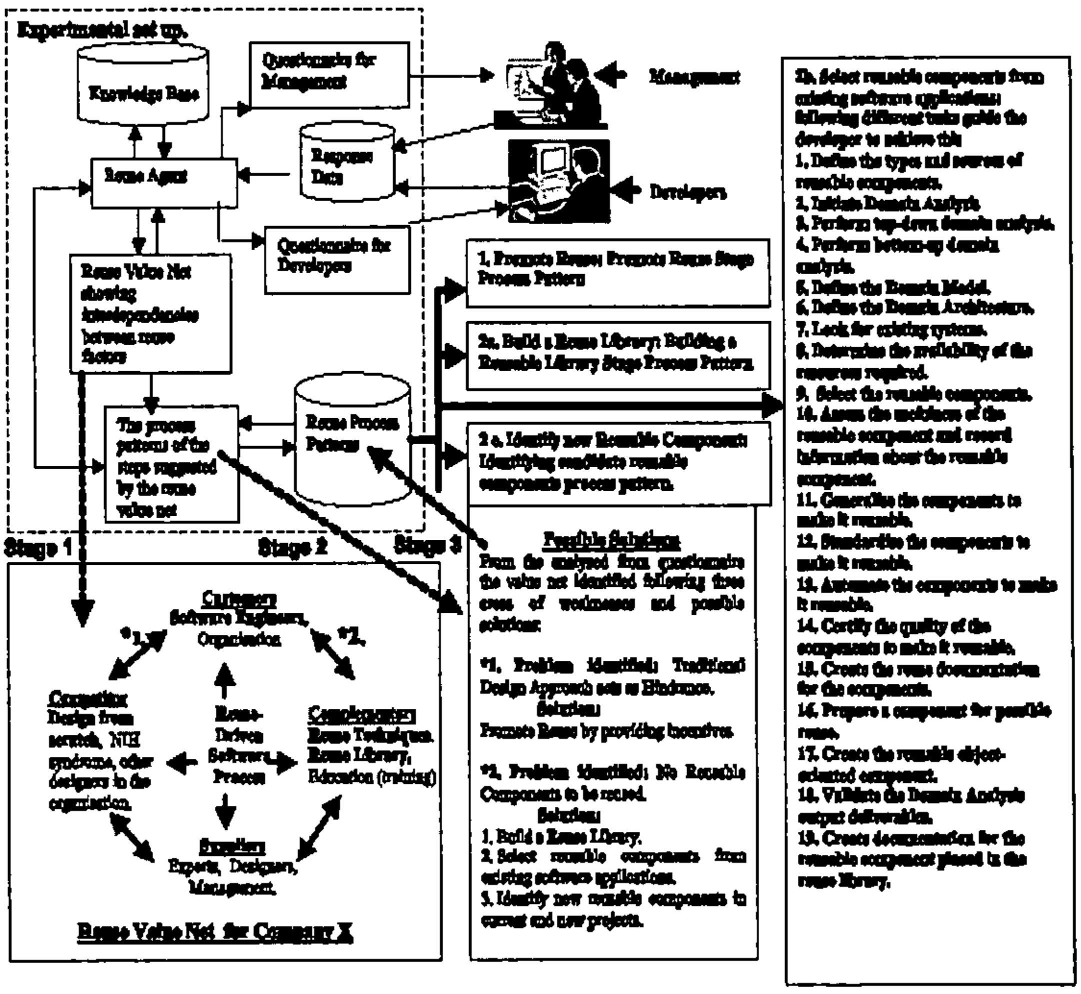


Figure 6. Reuse Value Net for example.

and produce a Reuse Value Net. The bold dotted arrow marked 'Stage 1' in figure 7 shows the Reuse Value Net that is produced for Company X in this example. The elements highlighted in bold show aspects requiring immediate attention. Typically two different types of problem may be identified through the Reuse Value Net, and these have been marked '*1' and '*2' in figure 7. Problems in the area marked '*1' are competitors, and therefore must be tackled by either decreasing their influence or by increasing the influence of reuse (to make reuse a more competitive option).

Possible solutions recommended by the Reuse Agent are also shown in figure 7 to the right of the Reuse Value Net. This is indicated by the bold dotted arrow marked 'Stage 2'. In this example, the recommended solution to problem '*1' is to promote reuse by providing incentives. As indicated by the arrows in



Experimental set up.
Knowledge Base
Questionnaire for Management
Response Data
Reuse Agent
Questionnaire for Developers
Reuse Value Net showing interdependencies between reuse factors
The process patterns of the steps suggested by the reuse value net
Reuse Process Patterns
Management
Developers
Stage 1
Stage 2
Stage 3
Customers Software Engineers, Organisation
*1,*2
*2.
Competitor Design from scratch, NIH syndrome, other designers in the organisation.
Reuse-Driven Software Process
Complementors Reuse Techniques, Reuse Library, Education (training)
Suppliers Experts, Designers, Management.
Reuse Value Net for Company X
1, Promote Reuse: Promote Reuse Stage Process Pattern
2a, Build a Reuse Library: Building a Reusable Library Stage Process Pattern.
2 c, Identify new Reusable Components Identifying candidate reusable components process pattern.

Possible Solutions

From the analysed from questionnaire the value net identified following three areas of weaknesses and possible solutions:

*1, Problem Identified: Traditional Design Approach acts as Hindrance.
Solution:
Promote Reuse by providing incentives.

*2, Problem Identified: No Reusable Components to be reused.
Solution:
1, Build a Reuse Library.
2, Select reusable components from existing software applications.
3, Identify new reusable components to current and new projects.

2b, Select reusable components from existing software applications: following different tasks guide the developer to achieve this
1, Define the types and sources of reusable components.
2, Initiate Domain Analysis
3, Perform top-down domain analysis.
4, Perform bottom-up domain analysis.
5, Define the Domain Model.
6, Define the Domain Architecture.
7, Look for existing systems.
8, Determine the availability of the resources required.
9. Select the reusable components.
10, Assess the usefulness of the reusable component and record information about the reusable component.
11, Generalise the components to make it reusable.
12, Standardise the components to make it reusable.
13, Automate the components to make it reusable.
14, Certify the quality of the components to make it reusable.
15, Create the reuse documentation for the components.
16, Prepare a component for possible reuse.
17, Create the reusable object-oriented component.
18, Validate the Domain Analysis output deliverables.
19, Create documentation for the reusable component placed in the reuse library.

Figure 7. Example application of reuse agent.

the experimental set-up figure, the Reuse Agent would also provide a process pattern showing steps that can be taken to achieve this. This is indicated by the bold dotted arrow marked 'Stage 3'. Detailed process pattern steps are not given in figure 7 for problem 1, due to space limitations. However, these steps would include, for example, activities such as setting up a bonus scheme to reward productivity increases through reuse of components and introduction of staff performance measures to assess whether reuse is being used effectively and whether staff are making it easy for others to reuse their components. The process steps suggested by the Reuse Agent are generic in form and need to be specialized by engineers within Company X to suit the requirements and environment of that individual company. The Reuse Agent can guide users through the important steps of specialization, such as the identification of suitable metrics for measurement of productivity, determination of periods of measurement, establishment of types and levels of incentives, etc. However, the actual details have to be decided by the management and users within the company, as successful applications of reuse will inevitably be different and specific to individual companies.

Problems identified in the Reuse Value net in the area marked '*2' on figure 7 relate to complementors, and typically any problems in this area indicate the absence of some key elements that are necessary to enable or facilitate reuse. They therefore must commonly be tackled by the provision of additional resources of some kind. In the example of Company X, there are three important activities that should be undertaken:

- Building a Reuse Library.
- Retrospective creation of reusable components from existing sources.
- Ongoing creation of reusable components from current and new projects (this must continue as an ongoing activity).

In figure 7, these three solutions are shown as 2.a, 2.b and 2.c.

Once again the Reuse Agent can guide users through the activities necessary to provide solutions to these identified problems by indicating useful steps to developing these necessary resources. Again, the process pattern steps suggested by the Reuse Agent are generic, and need specializing to suit the needs of the particular company. In figure 7, some example steps are provided for the solution 2b (and these will be specialized to enable the selection of reusable components from Company X's existing simulation models and information models). These steps consist of different Stage and Task Process Patterns, and this is demonstrated in further detail in the Task Process Patterns and the Stage Process Patterns shown in figure 8.

The first task shown in figure 8 is to *Determine the types and sources of reusable components*. This is the first step in the *Selecting Reusable Components Process Pattern*. If users request guidance on how to proceed with this task, the Reuse Agent can provide assistance through further process patterns. Hence the *Domain Analysis Process Pattern* (figure 8) would guide the user through an analysis of the domain, so that Company X would be encouraged to determine any detailed requirements it has for simulation modelling, and the resources and facilities that are needed to satisfy these requirements. Company X must also consider the types of simulation models that are likely to be required. This would include the key elements, resources and types of processes and structures that are used within the models and the information sources that are used to populate details of capabilities, requirements etc within the models. Having completed the domain analysis, figure 8 shows that Company X would then be guided to examine its existing systems, to identify which of the key elements already exist and then to look for similarities. If the same (or very similar) resource(s), processes or subprocesses commonly occur in models, these will be prime candidates for classification as 'reusable components'. If similar elements exist, the differences between them need to be examined to determine whether the differences are essential for the requirements of the individual cell that is being simulated, or are some of the differences purely due to individual designer preferences. Company X can be guided through these sorts of activities and decisions by the Reuse Agent, using the *Creating a Reusable Component Process Pattern* (as shown in figure 8).

The Reuse Agent can also guide design engineers and management through the necessary activities for achieving solutions 2.a and 2.c in a similar manner. The domain analysis carried out for solution 2.b is also an important requirement for both 2a and 2c. Hence, knowledge acquired during this stage will provide valuable background knowledge for the building of the Reuse Library with good catalogues and determining which elements of current projects should also be stored in the Reuse Library as reusable components. Classification and cataloguing of the reusable components in ways that make it quick and easy for design engineers to determine whether a particular component meets their requirements is an important incentive for encouraging reuse.

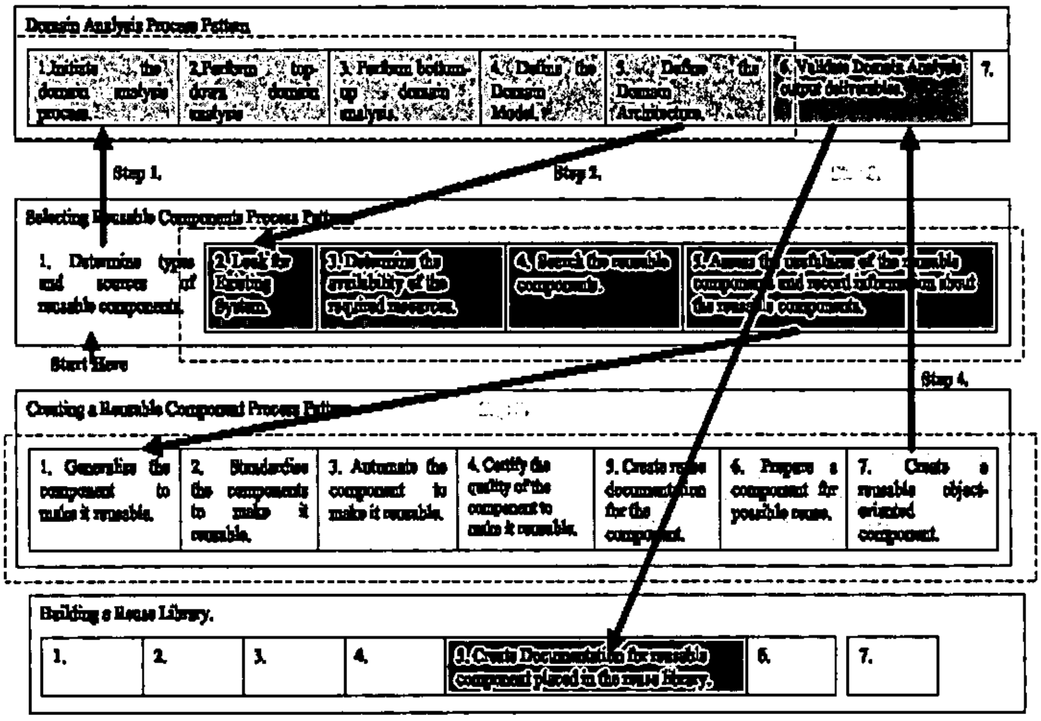

Domain Analysis Process Pattern
1. Initiate the domain analysis process.
2. Perform top-down domain analysis
3. Perform bottom-up domain analysis.
4. Define the Domain Model.
5. Define the Domain Architecture.
6. Validate Domain Analysis output deliverables.
7.
Step 1.
Step 2.
Selecting Reusable Components Process Pattern
1. Determine types and sources of reusable components.
2. Look for Existing System.
3. Determine the availability of the required resources.
4. Search the reusable components.
5. Assess the usefulness of the reusable components and record information about the reusable components.
Start Here
Step 4.
Creating a Reusable Component Process Pattern
1. Generalise the component to make it reusable.
2. Standardise the components to make it reusable.
3. Automate the component to make it reusable.
4. Certify the quality of the component to make it reusable.
5. Create reuse documentation for the component.
6. Prepare a component for possible reuse.
7. Create a reusable object-oriented component.
Building a Reuse Library.
1.
2.
3.
4.
5. Create Documentation for reusable component placed in the reuse library.
6.
7.

Figure 8. Process patterns approach for Reuse Value Net solutions.

6. Conclusions

An appropriate and efficient reuse process must be designed before it can be applied to the field of manufacturing system design. Reuse often fails because it is generally approached as an independent collection of tools and techniques, and the technical and non-technical aspects are commonly examined separately. This research proposes that all the factors affecting reuse are interdependent and therefore should be studied simultaneously. By understanding the interdependencies between various factors involved in the reuse process, an efficient reuse process can be designed. The concept of a reuse value net is proposed in this paper, and this enables the roles of the various factors affecting reuse to be studied. The value net theory has been applied to the reuse process and the various factors affecting reuse are mapped to formulate a reuse value net. Further work is required in order to study the interdependencies between the various factors in detail. The authors have identified a number of case studies and thorough experimentation is underway to demonstrate and evaluate the application of the reuse value net in achieving efficient manufacturing system design.

References

Ambler, S., 1998, *Process Patterns: Building Large-Scale Systems Using Object Technology* (Cambridge University Press).

Bhatnagar, R., and Viswanathan, S., 2000, Re-engineering global supply chains: alliances between manufacturing firms and global logistics services providers. *International Journal of Physical Distribution & Logistics Management*, 30(1), 13–34.

Busby, J. S., 1999, The problem with design reuse: an investigation into outcomes and antecedents. *Journal of Engineering Design*, 10(3), 277–296.

Davis, T., 1994, Adopting a policy of reuse. *IEEE Spectrum*, June, 44–48.

Dani, S. S., and Harding, J. A., 2003, A Pattern-driven approach to reuse in manufacturing system design. Working paper, Product Realisation Technologies Research Group, Wolfson School of Mechanical and Manufacturing Engineering, Loughborough University, Leicestershire, UK.

Fafchamps, D., 1994, Organizational factors and reuse. *IEEE Software*, September, 31–41.

Frakes, W. B., and Isoda, S., 1994, Success factors of systematic reuse. *IEEE Software*, September, 14–19.

Harding, J. A., and Popplewell, K., 1998, Simulation: an application of factory design process methodology. *Journal of Operations Research Society*, 51, 440–448.

Harding, J. A., Yu, B., and Popplewell, K., 1999, Information modelling: an integration of views of a manufacturing enterprise. *International Journal of Production Research*, 37(12), 2777–2792.

Henry, E., and Faller, B., 1995, Large-scale industrial reuse to reduce cost and cycle time. *IEEE Software*, September, 47–53.

Lim, W. C., 1994, Effects of reuse on quality, productivity, and economics. *IEEE Software*, September, 23–31.

McClure, C., 1997, *Software Reuse Techniques: Adding Reuse to the System Development Process* (Prentice Hall).

Nalebuff, B. J., and Brandenburger, A. M., 1996, *Co-opetition: 1) A Revolutionary Mindset that Combines Competition and Cooperation 2) The Game Theory Strategy that's Changing the Game of Business* (Harper Collins Business).

Rabino, H., *Applying the Principles of Co-opetition With System Dynamics Tools*. SDSG, LLC (The Strategic Decision Simulation Group), http://www.sdsg.com/

Rine, C. D., and Nada, N., 2000, Software reuse reference model: development and validation. *Journal of Information and Software Technology*, **42**, 47–65.

Rising, L., 2002, Reuse at AG Communication Systems- Patterns, http://www.agcs.com/supportv2/techpapers/patterns/papers/multi.htm (first published in *MultiUse Express*, 4(3), June 1996, p. 3).

Sivaloganathan, S., and Shahin, T. M., 1999, Design reuse: an overview. *Proceedings of the IMECH E Part B Journal of Engineering Manufacture*, **213**(7), 641–654.

Yu, B., Harding, J. A., and Popplewell, K., 2000, A reusable enterprise model. *International Journal of Operations & Production Management*, **20**(1), 50–69.

GLOSSARY OF TERMS

The definitions of important words and phrases used in this thesis are listed here, with the section number where the main or initial usage can be found.

Agent is a combination of human and software expertise, interacting with the CAE system.

Best practices are those practices that have been shown to produce superior results; selected by a systematic process; and judged as exemplary, good, or successfully demonstrated.

Data relates simply to words or numbers the meaning of which may vary and is dependent upon the context in which the data is used.

Expert is a system designer with a high degree of skill in or knowledge as the result of experience or training.

Expert system any computer program which demonstrates expert performance in a given domain.

Information is data which is structured or titled in some way so that it has a particular meaning.

Information model is a formal description of types of ideas, facts and processes which together form a model of a portion of interest of the real world and which provides an explicit set of interpretation rules.

Knowledge is information with added detail relating to how it may be used or applied.

Knowledge base is a collection of knowledge related to a specific problem domain. A knowledge engineering translate an expert's knowledge into a computer knowledge representation.

Manufacturing system is a set of machines, transportation elements, computers, storage buffers, and other items that are used together for manufacturing. People are also part of the system.

Manufacturing system design is choice of machines, buffers, transport systems, computers and communication system, operators, repaire personnel, and so forth.

Not-Invented-Here Syndrome

Object-Oriented Software Process (OOSP) is a collection of process patterns that together describe a complete process for developing, maintaining, and supporting software.

Organizational culture ' The pattern of basic assumption that the given group has invented, discovered or developed in learning to cope with its problem of external adaptation and internal integration and that have worked well enough to be considered valid and therefore to be taught to new members as the correct way to perceive, think and feel in relation to those problems.'

Pattern The description of a general solution to a common problem or issue from which a detailed solution to a specific problem may be determined. Software development patterns come in many flavors, including but not limited to analysis patterns, design patterns, and process patterns.

Phase process pattern a process pattern that depicts the interactions between the stage process patterns for a single project phase.

Process is a series of actions in which one or more inputs are used to produce one or more outputs.

Process pattern is a pattern which describes a proven, successful approach and/or series of actions for developing software.

Reuse- is simply the use of an 'asset' in different setting/context/environment/application etc. An asset could be a Business Process, design, specification, implementation, verification, documentation, etc.(hettp://www.cse.dmu.ac.uk/~nmsampat/research/subject/reuse/reuse-defn.html)

Stage process pattern – A process pattern which depicts the steps, often performed iteratively, of a single project stage.

System design- is a process of defining the hardware and software architecture, components, modules, interfaces, and data for a system to satisfy specified requirements.

Software reuse- is the practice of developing a new application/s from existing software.

Task process pattern – A process pattern that depicts the detailed steps to perform a specific task, such as detailed modeling or performing a technical review.

The Value Net- is a schematic map designed to represent all the players in the game and the interdependencies among them.

Source code

ReuseAgentApp.java

Created with JBuilder

```
package reuseagent1;

import javax.swing.UIManager;
import java.awt.*;

/**
 * Title:        Reuse Agent
 * Description:
 * Copyright:    Copyright (c) 2004
 * Company:      Loughborough University
 * @author S.Dani
 * @version 1.0
 */

public class ReuseAgentApp {
  boolean packFrame = false;

  /**Construct the application*/
  public ReuseAgentApp() {
    ReuseAgent1MainMenu frame = new ReuseAgent1MainMenu();
    //Validate frames that have preset sizes
    //Pack frames that have useful preferred size info, e.g. from
their layout
    if (packFrame) {
      frame.pack();
    }
    else {
      frame.validate();
    }
    //Center the window
    Dimension screenSize =
Toolkit.getDefaultToolkit().getScreenSize();
    Dimension frameSize = frame.getSize();
    if (frameSize.height > screenSize.height) {
      frameSize.height = screenSize.height;
    }
    if (frameSize.width > screenSize.width) {
      frameSize.width = screenSize.width;
    }
    frame.setLocation((screenSize.width - frameSize.width) / 2,
(screenSize.height - frameSize.height) / 2);
    frame.setVisible(true);
  }
  /**Main method*/
  public static void main(String[] args) {
    try {

UIManager.setLookAndFeel(UIManager.getSystemLookAndFeelClassName());
    }
    catch(Exception e) {
      e.printStackTrace();
    }
    new ReuseAgentApp();
  }
}
```

ReuseAgent1MainMenu.java

Created with JBuilder

```
package reuseagent1;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;

/**
 * Title:        Reuse Agent
 * Description:
 * Copyright:    Copyright (c) 2004
 * Company:      Loughborough University
 * @author S.Dani
 * @version 1.0
 */

public class ReuseAgent1MainMenu extends JFrame {
  JPanel contentPane;
  JMenuBar jMenuBar1 = new JMenuBar();
  JMenu jMenuFile = new JMenu();
  JMenuItem jMenuFileExit = new JMenuItem();
  JMenu jMenuHelp = new JMenu();
  JMenuItem jMenuHelpAbout = new JMenuItem();
  JLabel statusBar = new JLabel();
  BorderLayout borderLayout1 = new BorderLayout();
  JMenu jMenu1 = new JMenu();
  JMenuItem jMenuItem1 = new JMenuItem();
  JMenuItem jMenuItem2 = new JMenuItem();
  JMenu jMenu2 = new JMenu();
  JMenuItem jMenuItem3 = new JMenuItem();
  JMenuItem jMenuItem4 = new JMenuItem();

  /**Construct the frame*/
  public ReuseAgent1MainMenu() {
    enableEvents(AWTEvent.WINDOW_EVENT_MASK);
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
  /**Component initialization*/
  private void jbInit() throws Exception  {

//setIconImage(Toolkit.getDefaultToolkit().createImage(ReuseAgent1Mai
nMenu.class.getResource("[Your Icon]")));
    contentPane = (JPanel) this.getContentPane();
    contentPane.setLayout(borderLayout1);
    this.setSize(new Dimension(400, 300));
    this.setTitle("Reuse Agent");
    statusBar.setText(" ");
    jMenuFile.setText("File");
    jMenuFileExit.setText("Exit");
    jMenuFileExit.addActionListener(new ActionListener()  {
      public void actionPerformed(ActionEvent e) {
        jMenuFileExit_actionPerformed(e);
      }
    });
    jMenuHelp.setText("Help");
    jMenuHelpAbout.setText("About");
    jMenuHelpAbout.addActionListener(new ActionListener()  {
      public void actionPerformed(ActionEvent e) {
        jMenuHelpAbout_actionPerformed(e);
      }
    });
    jMenu1.setText("Reuse Support");
    jMenuItem1.setText("Formulate Reuse Value Net");
    jMenuItem1.addActionListener(new java.awt.event.ActionListener()
{
      public void actionPerformed(ActionEvent e) {
        jMenuItem1_actionPerformed(e);
      }
    });
    jMenuItem2.setText("Reuse Questionnaire");
    jMenuItem2.addActionListener(new java.awt.event.ActionListener()
{
      public void actionPerformed(ActionEvent e) {
        jMenuItem2_actionPerformed(e);
      }
    });
    jMenu2.setText("Process Pattern");
    jMenuItem3.setText("Search Pattern");
    jMenuItem3.addActionListener(new java.awt.event.ActionListener()
{
      public void actionPerformed(ActionEvent e) {
        jMenuItem3_actionPerformed(e);
      }
    });
    jMenuItem4.setText("Display Pattern Template");
    jMenuItem4.addActionListener(new java.awt.event.ActionListener()
{
      public void actionPerformed(ActionEvent e) {
        jMenuItem4_actionPerformed(e);
      }
    });
    jMenuFile.add(jMenuFileExit);
    jMenuHelp.add(jMenuHelpAbout);
    jMenuBar1.add(jMenuFile);
    jMenuBar1.add(jMenu1);
    jMenuBar1.add(jMenu2);
    jMenuBar1.add(jMenuHelp);
//    jMenuBar1.add(jMenu2);
    this.setJMenuBar(jMenuBar1);
    contentPane.add(statusBar, BorderLayout.SOUTH);
    jMenu1.add(jMenuItem1);
    jMenu1.add(jMenuItem2);
    jMenu2.add(jMenuItem3);
    jMenu2.add(jMenuItem4);
  }
  /**File | Exit action performed*/
  public void jMenuFileExit_actionPerformed(ActionEvent e) {
    System.exit(0);
  }
  /**Help | About action performed*/
  public void jMenuHelpAbout_actionPerformed(ActionEvent e) {
    ReuseAgent1MainMenu_AboutBox dlg = new
ReuseAgent1MainMenu_AboutBox(this);
```

```
    Dimension dlgSize = dlg.getPreferredSize();
    Dimension frmSize = getSize();
    Point loc = getLocation();
    dlg.setLocation((frmSize.width - dlgSize.width) / 2 + loc.x,
(frmSize.height - dlgSize.height) / 2 + loc.y);
    dlg.setModal(true);
    dlg.show();
  }
  /**Overridden so we can exit when window is closed*/
  protected void processWindowEvent(WindowEvent e) {
    super.processWindowEvent(e);
    if (e.getID() == WindowEvent.WINDOW_CLOSING) {
      jMenuFileExit_actionPerformed(null);
    }
  }

  void jMenuItem1_actionPerformed(ActionEvent e) {
    JInternalFrame frame = new JInternalFrame("Organisation's Reuse
Value Net", true, true, true, true);

    Container container = frame.getContentPane();
    FormulateReuseValueNet reusevaluenet = new
FormulateReuseValueNet();
    container.add(reusevaluenet, BorderLayout.CENTER);
    frame.pack();
    contentPane.add(frame);
    frame.setVisible(true);

  }

  void jMenuItem2_actionPerformed(ActionEvent e) {
      JInternalFrame frame = new JInternalFrame("Reuse
Questionnaire", true, true, true, true);

      Container container = frame.getContentPane();
      ReuseQuestionnaire1 reuseq = new ReuseQuestionnaire1();
      container.add(reuseq, BorderLayout.CENTER);
      frame.pack();
      contentPane.add(frame);
      frame.setVisible(true);

  }

  void jMenuItem3_actionPerformed(ActionEvent e) {
      JInternalFrame frame = new JInternalFrame("Search Pattern",
true, true, true, true);

      Container container = frame.getContentPane();
      SearchPattern searchpattern = new SearchPattern();
      container.add(searchpattern, BorderLayout.CENTER);
      frame.pack();
      contentPane.add(frame);
      frame.setVisible(true);

  }

  void jMenuItem4_actionPerformed(ActionEvent e) {
      JInternalFrame frame = new JInternalFrame("Display Pattern
Template", true, true, true, true);

      Container container = frame.getContentPane();
      DisplayPattern displaypattern = new DisplayPattern();
      container.add(displaypattern, BorderLayout.CENTER);
      frame.pack();
      contentPane.add(frame);
      frame.setVisible(true);

  }
}
```

ReuseQuestionnaire1.java

Created with JBuilder

```
package reuseagent1;

import javax.swing.*;
import com.borland.dx.sql.dataset.*;
import java.awt.event.*;
import com.borland.jbcl.layout.*;
import com.borland.dbswing.*;
import java.awt.*;
import com.borland.dx.dataset.*;

/**
 * Title:         Reuse Agent
 * Description:
 * Copyright:     Copyright (c) 2004
 * Company:       Loughborough University
 * @author
 * @version 1.0
 */

public class ReuseQuestionnaire1 extends JPanel{
 // DataMod_MasterDetail datmodMD = new DataMod_MasterDetail();
  DataMod_MasterDetail datmodMD =
DataMod_MasterDetail.getDataModule();
  JPanel jPanel1 = new JPanel();


  JTabbedPane jTabbedPane1 = new JTabbedPane();
  JPanel jPanel2 = new JPanel();
  JPanel jPanel3 = new JPanel();

  JPanel jPanel5 = new JPanel();
  JPanel jPanel6 = new JPanel();
  JPanel jPanel7 = new JPanel();
  JPanel jPanel8 = new JPanel();
  JPanel jPanel9 = new JPanel();
  JPanel jPanel10 = new JPanel();
  JPanel jPanel11 = new JPanel();
  JPanel jPanel12 = new JPanel();
  JPanel jPanel13 = new JPanel();
  JPanel jPanel14 = new JPanel();
  JPanel jPanel15 = new JPanel();
  JPanel jPanel16 = new JPanel();

  JButton jButton1 = new JButton();
  JButton jButton2 = new JButton();
  JLabel jLabel2 = new JLabel();
  JLabel jLabel3 = new JLabel();
  JLabel jLabel4 = new JLabel();
  JLabel jLabel5 = new JLabel();
JLabel jLabel6 = new JLabel();
JLabel jLabel7 = new JLabel();
JLabel jLabel8 = new JLabel();
JLabel jLabel9 = new JLabel();
JLabel jLabel10 = new JLabel();
JLabel jLabel11 = new JLabel();
JLabel jLabel12 = new JLabel();
JLabel jLabel14 = new JLabel();
JLabel jLabel15 = new JLabel();
JLabel jLabel16 = new JLabel();
JLabel jLabel17 = new JLabel();
JLabel jLabel18 = new JLabel();
JLabel jLabel19 = new JLabel();
JLabel jLabel20 = new JLabel();

QueryDataSet queryDataSet1 = new QueryDataSet();
VerticalFlowLayout verticalFlowLayout2 = new VerticalFlowLayout();

Database database1 = new Database();
JPanel jPanel4 = new JPanel();
BorderLayout borderLayout1 = new BorderLayout();
JdbNavToolBar jdbNavToolBar1 = new JdbNavToolBar();
JdbStatusLabel jdbStatusLabel1 = new JdbStatusLabel();
TableScrollPane tableScrollPane1 = new TableScrollPane();
JButton jButton3 = new JButton();
JdbTable jdbTable1 = new JdbTable();
ButtonGroup buttonGroup1 = new ButtonGroup();

ButtonGroup D13 = new ButtonGroup();
ButtonGroup D12 = new ButtonGroup();
ButtonGroup D11 = new ButtonGroup();
ButtonGroup D10 = new ButtonGroup();
ButtonGroup D1 = new ButtonGroup();
ButtonGroup D2 = new ButtonGroup();
ButtonGroup D3 = new ButtonGroup();
ButtonGroup D4 = new ButtonGroup();
ButtonGroup D5 = new ButtonGroup();
ButtonGroup D6 = new ButtonGroup();
ButtonGroup D7 = new ButtonGroup();
ButtonGroup D8 = new ButtonGroup();
ButtonGroup D9 = new ButtonGroup();
ButtonGroup M1 = new ButtonGroup();
ButtonGroup M2 = new ButtonGroup();
ButtonGroup M3 = new ButtonGroup();
ButtonGroup M4 = new ButtonGroup();
ButtonGroup M5 = new ButtonGroup();
ButtonGroup M6 = new ButtonGroup();
ButtonGroup M7 = new ButtonGroup();

JRadioButton jRadioButton40 = new JRadioButton();
JRadioButton jRadioButton39 = new JRadioButton();
JRadioButton jRadioButton38 = new JRadioButton();
JRadioButton jRadioButton37 = new JRadioButton();
JRadioButton jRadioButton36 = new JRadioButton();
JRadioButton jRadioButton35 = new JRadioButton();
JRadioButton jRadioButton34 = new JRadioButton();
JRadioButton jRadioButton33 = new JRadioButton();
JRadioButton jRadioButton32 = new JRadioButton();
JRadioButton jRadioButton31 = new JRadioButton();
JRadioButton jRadioButton30 = new JRadioButton();
JRadioButton jRadioButton29 = new JRadioButton();
```

```
  JRadioButton jRadioButton28 = new JRadioButton();
  JRadioButton jRadioButton27 = new JRadioButton();
  JRadioButton jRadioButton26 = new JRadioButton();
  JRadioButton jRadioButton25 = new JRadioButton();
  JRadioButton jRadioButton24 = new JRadioButton();
  JRadioButton jRadioButton23 = new JRadioButton();
  JRadioButton jRadioButton22 = new JRadioButton();
  JRadioButton jRadioButton21 = new JRadioButton();
  JRadioButton jRadioButton20 = new JRadioButton();
  JRadioButton jRadioButton9 = new JRadioButton();
  JRadioButton jRadioButton8 = new JRadioButton();
  JRadioButton jRadioButton7 = new JRadioButton();
  JRadioButton jRadioButton6 = new JRadioButton();
  JRadioButton jRadioButton5 = new JRadioButton();
  JRadioButton jRadioButton4 = new JRadioButton();
  JRadioButton jRadioButton3 = new JRadioButton();
  JRadioButton jRadioButton19 = new JRadioButton();
  JRadioButton jRadioButton18 = new JRadioButton();
  JRadioButton jRadioButton17 = new JRadioButton();
  JRadioButton jRadioButton16 = new JRadioButton();
  JRadioButton jRadioButton15 = new JRadioButton();
  JRadioButton jRadioButton14 = new JRadioButton();
  JRadioButton jRadioButton13 = new JRadioButton();
  JRadioButton jRadioButton12 = new JRadioButton();
  JRadioButton jRadioButton11 = new JRadioButton();
  JRadioButton jRadioButton10 = new JRadioButton();
  FlowLayout flowLayout1 = new FlowLayout();
  VerticalFlowLayout verticalFlowLayout1 = new VerticalFlowLayout();
  JLabel jLabel13 = new JLabel();
  JRadioButton jRadioButton2 = new JRadioButton();
  JRadioButton jRadioButton1 = new JRadioButton();
  JPanel jPanel17 = new JPanel();
  VerticalFlowLayout verticalFlowLayout3 = new VerticalFlowLayout();
  FlowLayout flowLayout2 = new FlowLayout();
  JLabel jLabel1 = new JLabel();
  QueryDataSet queryDataSet2 = new QueryDataSet();
  Column column1 = new Column();
  DBDisposeMonitor dBDisposeMonitor1 = new DBDisposeMonitor();


  public ReuseQuestionnaire1() {
      try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
}

  public void paint(Graphics g)
  {
    super.paint(g);
  }

  private void jbInit() throws Exception {
    jTabbedPane1.setBackground(new Color(147, 143, 255));
    jTabbedPane1.setFont(new java.awt.Font("Dialog", 1, 13));
    jTabbedPane1.setPreferredSize(new Dimension(950, 1200));
    jButton1.setText("OK");
    jButton1.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jButton1_actionPerformed(e);
      }
    });

        jRadioButton10.setText("No.");
    jRadioButton10.setBackground(new Color(151, 235, 255));
    jRadioButton11.setText("Yes.");
    jRadioButton11.setBackground(new Color(151, 235, 255));
    jRadioButton12.setText("No.");
    jRadioButton12.setBackground(new Color(151, 235, 255));
    jRadioButton13.setText("No.");
    jRadioButton13.setBackground(new Color(151, 235, 255));
    jRadioButton14.setText("Yes.");
    jRadioButton14.setBackground(new Color(151, 235, 255));
    jRadioButton15.setText("Yes.");
    jRadioButton15.setBackground(new Color(151, 235, 255));
    jRadioButton16.setText("No.");
    jRadioButton16.setBackground(new Color(151, 235, 255));
    jRadioButton17.setText("Yes.");
    jRadioButton17.setBackground(new Color(151, 235, 255));
    jRadioButton18.setText("No.");
    jRadioButton18.setBackground(new Color(151, 235, 255));
    jRadioButton19.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jRadioButton19_actionPerformed(e);
      }
    });
    jRadioButton19.setText("Yes.");
    jRadioButton19.setBackground(new Color(151, 235, 255));
    jRadioButton3.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jRadioButton3_actionPerformed(e);
      }
    });
    jRadioButton3.setText("Yes.");
    jRadioButton3.setBackground(new Color(151, 235, 255));
    jRadioButton4.setText("No.");
    jRadioButton4.setBackground(new Color(151, 235, 255));
    jRadioButton5.setText("Yes.");
    jRadioButton5.setBackground(new Color(151, 235, 255));
    jRadioButton6.setText("No.");
    jRadioButton6.setBackground(new Color(151, 235, 255));
    jRadioButton7.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jRadioButton7_actionPerformed(e);
      }
    });
    jRadioButton7.setText("Yes.");
    jRadioButton7.setBackground(new Color(151, 235, 255));
    jRadioButton8.setText("No.");
    jRadioButton8.setBackground(new Color(151, 235, 255));
    jRadioButton9.setText("Yes.");
    jRadioButton9.setBackground(new Color(151, 235, 255));
    jRadioButton20.setText("No.");
    jRadioButton20.setBackground(new Color(151, 235, 255));
    jRadioButton21.setText("Yes.");
    jRadioButton21.setBackground(new Color(151, 235, 255));
    jRadioButton22.setText("No.");
    jRadioButton22.setBackground(new Color(151, 235, 255));
    jRadioButton23.setText("Yes.");
    jRadioButton23.setBackground(new Color(151, 235, 255));
    jRadioButton24.setText("No.");
    jRadioButton24.setBackground(new Color(151, 235, 255));
    jRadioButton25.setText("Yes.");
    jRadioButton25.setBackground(new Color(151, 235, 255));
    jRadioButton25.setMinimumSize(new Dimension(10, 25));
    jRadioButton25.setPreferredSize(new Dimension(50, 23));
    jRadioButton26.setText("No.");
    jRadioButton26.setBackground(new Color(151, 235, 255));
    jRadioButton27.setText("Yes.");
    jRadioButton27.setBackground(new Color(122, 150, 255));
    jRadioButton28.setText("No.");
    jRadioButton28.setBackground(new Color(122, 150, 255));
    jRadioButton29.setText("No.");
    jRadioButton29.setBackground(new Color(122, 150, 255));
    jRadioButton30.setText("Yes.");
    jRadioButton30.setBackground(new Color(122, 150, 255));
    jRadioButton31.setText("Yes.");
    jRadioButton31.setBackground(new Color(122, 150, 255));
    jRadioButton32.setText("No.");
    jRadioButton32.setBackground(new Color(122, 150, 255));
    jRadioButton33.setText("Yes.");
    jRadioButton33.setBackground(new Color(122, 150, 255));
    jRadioButton34.setText("No.");
    jRadioButton34.setBackground(new Color(122, 150, 255));
    jRadioButton35.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jRadioButton35_actionPerformed(e);
      }
    });
    jRadioButton35.setText("Yes.");
    jRadioButton35.setBackground(new Color(122, 150, 255));
    jRadioButton36.setText("No.");
    jRadioButton36.setBackground(new Color(122, 150, 255));
    jRadioButton37.setText("No.");
    jRadioButton37.setBackground(new Color(122, 150, 255));
    jRadioButton38.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jRadioButton38_actionPerformed(e);
      }
    });
    jRadioButton38.setText("Yes.");
    jRadioButton38.setBackground(new Color(122, 150, 255));
    jRadioButton39.addActionListener(new
java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jRadioButton39_actionPerformed(e);
      }
    });
    jRadioButton39.addItemListener(new java.awt.event.ItemListener()
{
      public void itemStateChanged(ItemEvent e) {
        jRadioButton39_itemStateChanged(e);
      }
    });
    jRadioButton39.setText("Yes.");
    jRadioButton39.setBackground(new Color(122, 150, 255));
    jRadioButton40.setText("No.");
    jRadioButton40.setBackground(new Color(122, 150, 255));




    jButton2.setText("Reset");
    jPanel2.setBackground(Color.cyan);
    jPanel2.setPreferredSize(new Dimension(600, 400));
    jPanel2.setLayout(verticalFlowLayout3);
    jPanel3.setBackground(new Color(122, 150, 255));
    jPanel3.setPreferredSize(new Dimension(400, 250));
    jPanel3.setLayout(verticalFlowLayout2);
    jPanel1.setPreferredSize(new Dimension(800, 500));
    jPanel1.setToolTipText("");
    jLabel2.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel2.setText("D2. Have you ever tried to reuse codes or models
designed by other " +
    "designers?
" +
    "                 ");
    jLabel3.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel3.setText("D3. Do the designers in your organisation
commonly reuse codes or " +
    "models?
" +
    "                    ");
    jLabel4.setBackground(Color.blue);
    jLabel4.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel4.setText("D4. Have you ever tried to reuse models from
your previour project " +
    "or models from your designed by other designers?");
    jLabel5.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel5.setText("D5. Do you refer to the software components that
can be used in the " +
    "particular project that are available for reuse?     ");
    jLabel6.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel6.setText("D6. Do the designers look for reusable
components that are used in " +
    "the early phases of the life-cycle?
");
    jLabel7.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel7.setText("D7. Do you share/exchange reusable components
with other team members? " +
    "
" +
    "          ");
    jLabel8.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel8.setText("D8. Do the designers identify new candidate
reusable components? " +
    "
" +
    "                              ");
    jLabel9.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel9.setText("D9. Are the candidate reusable components
documented by the designers " +
    "for future reuse?
" +
    " ");
    jLabel10.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel10.setText("D10. Have you ever tried to reuse codes or
models designed or documented " +
    "for reuse?
" +
    "    ");
    jLabel11.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel11.setText("D11. Are there any reuse guidelines used in
your organisation?   " +
    "
" +
    "                                      ");
    jLabel12.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel12.setText("D12. Does your organisation have reuse
catalogue?                 " +
    "
" +
    "                                              ");
    jLabel14.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel14.setText("M7. ");
    jLabel15.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel15.setText("M6. Are there any mechanism in your
organisation that encourage sharing " +
    "and reuse?");
    jLabel16.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel16.setText("M5. Is there a reuse sponsor in your
organisation?");
    jLabel17.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel17.setText("M4. Is management ready to provide funding? ");
    jLabel18.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel18.setText("M3. Is there a formal incentive program in
place or planned?");
    jLabel19.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel19.setText("M2. Does your organisation have resources
commited to supporting " +
    "a reuse progam?");
    jLabel20.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel20.setText("M1. Is reuse accepted by upper management?");

    database1.setConnection(new
com.borland.dx.sql.dataset.ConnectionDescriptor("jdbc:borland:dslocal
:C:\\shilpa\\ReuseSupportTool.jds", "ensd", "", false,
"com.borland.datastore.jdbc.DataStoreDriver"));
    queryDataSet1.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database1, "select
RESPONSEDATA.QUES_ID, RESPONSEDATA.ANSWER,
REUSEKNOWLEDGEBASE.ANSWER\n" +
      "from RESPONSEDATA, REUSEKNOWLEDGEBASE\nwhere
RESPONSEDATA.QUES_ID " +
      "= REUSEKNOWLEDGEBASE.QUES_ID\n", null, true, Load.AS_NEEDED));
    jPanel4.setLayout(borderLayout1);
    jdbStatusLabel1.setText("jdbStatusLabel1");
    jdbStatusLabel1.setDataSet(queryDataSet1);
    jButton3.setMinimumSize(new Dimension(30, 27));
    jButton3.setPreferredSize(new Dimension(150, 27));
    jButton3.setText("Save Changes");
    jButton3.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jButton3_actionPerformed(e);
      }
    });
    jdbTable1.setDataSet(queryDataSet1);
    jdbNavToolBar1.setDataSet(queryDataSet1);


    jLabel13.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel13.setText("D13. Does your organisation have a reuse 
library?                " +
    "
" +
    "                                                        ");
    jRadioButton2.setText("No.");
    jRadioButton2.setBackground(new Color(151, 235, 255));
    jRadioButton1.addActionListener(new 
java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jRadioButton1_actionPerformed(e);
      }
    });
    jRadioButton1.setText("Yes.");
    jRadioButton1.setBackground(new Color(151, 235, 255));
    jPanel6.setBackground(Color.cyan);
    jPanel7.setBackground(Color.cyan);
    jPanel8.setBackground(Color.cyan);
    jPanel9.setBackground(Color.cyan);
    jPanel10.setBackground(Color.cyan);
    jPanel11.setBackground(Color.cyan);
    jPanel12.setBackground(Color.cyan);
    jPanel13.setBackground(Color.cyan);
    jPanel5.setBackground(Color.cyan);
    jPanel5.setLayout(flowLayout2);
    jPanel17.setBackground(Color.cyan);
    jPanel16.setBackground(Color.cyan);
    jPanel15.setBackground(Color.cyan);
    jPanel14.setBackground(Color.cyan);
    jLabel1.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel1.setText("D1. Do the designers in your organisation know 
about reusable components? " +
    "
" +
    "          ");
    queryDataSet2.setResolveOrder(new String[] {"RESPONSEDATA", 
"REUSEKNOWLEDGEBASE"});
    queryDataSet2.setQuery(new 
com.borland.dx.sql.dataset.QueryDescriptor(database1, "select 
REUSEKNOWLEDGEBASE.QUES_ID, REUSEKNOWLEDGEBASE.ANSWER \nfrom " +
      "REUSEKNOWLEDGEBASE, RESPONSEDATA\nwhere 
REUSEKNOWLEDGEBASE.QUES_ID " +
      "= RESPONSEDATA.QUES_ID", null, true, Load.AS_NEEDED));
    dBDisposeMonitor1.setDataAwareComponentContainer(this);
    jTabbedPane1.add(jPanel4, "RESPONSE DATA");
    jPanel4.add(jdbNavToolBar1,  BorderLayout.NORTH);
    jPanel4.add(jdbStatusLabel1,  BorderLayout.SOUTH);
    jPanel4.add(tableScrollPane1,  BorderLayout.CENTER);
    tableScrollPane1.getViewport().add(jdbTable1, null);
    jPanel4.add(jButton3,  BorderLayout.EAST);
    jTabbedPane1.add(jPanel3, "MANAGEMENT QUESTIONNAIRE");
    jPanel3.add(jLabel20, null);
    jPanel3.add(jLabel19, null);
    jPanel3.add(jLabel18, null);
    jPanel3.add(jLabel17, null);
    jPanel3.add(jLabel16, null);
    jPanel3.add(jLabel15, null);
```

```
//jPanel3.add(jLabel14, null);
jTabbedPanel.add(jPanel2, "DESIGNER QUESTIONNAIRE");
jPanel1.add(jTabbedPanel, null);
jPanel1.add(jButton1, null);
jPanel1.add(jButton2, null);

jPanel3.add(jLabel20, null);
jPanel3.add(jRadioButton39, null);
jPanel3.add(jRadioButton40, null);
jPanel3.add(jLabel19, null);
jPanel3.add(jRadioButton38, null);
jPanel3.add(jRadioButton37, null);
jPanel3.add(jLabel18, null);
jPanel3.add(jRadioButton35, null);
jPanel3.add(jRadioButton36, null);
jPanel3.add(jLabel17, null);
jPanel3.add(jRadioButton33, null);
jPanel3.add(jRadioButton34, null);
jPanel3.add(jLabel16, null);
jPanel3.add(jRadioButton31, null);
jPanel3.add(jRadioButton32, null);
jPanel3.add(jLabel15, null);
jPanel3.add(jRadioButton30, null);
jPanel3.add(jRadioButton29, null);

jPanel5.add(jLabel1, null);
jPanel5.add(jRadioButton25, null);
jPanel5.add(jRadioButton26, null);


jPanel6.add(jLabel2, null);
jPanel6.add(jRadioButton23, null);
jPanel6.add(jRadioButton24, null);

jPanel7.add(jLabel3, null);
jPanel7.add(jRadioButton21, null);
jPanel7.add(jRadioButton22, null);

jPanel8.add(jLabel4, null);
jPanel8.add(jRadioButton19, null);
jPanel8.add(jRadioButton20, null);

jPanel9.add(jLabel5, null);
jPanel9.add(jRadioButton17, null);
jPanel9.add(jRadioButton18, null);

jPanel10.add(jLabel6, null);
jPanel10.add(jRadioButton15, null);
jPanel10.add(jRadioButton16, null);

jPanel11.add(jLabel7, null);
jPanel11.add(jRadioButton14, null);
jPanel11.add(jRadioButton13, null);

jPanel12.add(jLabel8, null);
jPanel12.add(jRadioButton11, null);
jPanel12.add(jRadioButton12, null);

jPanel13.add(jLabel9, null);
jPanel13.add(jRadioButton9, null);
jPanel13.add(jRadioButton10, null);
```

```
jPanel14.add(jLabel10, null);
jPanel14.add(jRadioButton7, null);
jPanel14.add(jRadioButton8, null);

jPanel15.add(jLabel11, null);
jPanel15.add(jRadioButton5, null);
jPanel15.add(jRadioButton6, null);

jPanel16.add(jLabel12, null);
jPanel16.add(jRadioButton3, null);
jPanel16.add(jRadioButton4, null);

jPanel17.add(jLabel13, null);
jPanel17.add(jRadioButton1, null);
jPanel17.add(jRadioButton2, null);

jPanel2.add(jPanel5, null);
jPanel2.add(jPanel6, null);
jPanel2.add(jPanel7, null);
jPanel2.add(jPanel8, null);
jPanel2.add(jPanel9, null);
jPanel2.add(jPanel10, null);
jPanel2.add(jPanel11, null);
jPanel2.add(jPanel12, null);
jPanel2.add(jPanel13, null);
jPanel2.add(jPanel14, null);
jPanel2.add(jPanel15, null);
jPanel2.add(jPanel16, null);
jPanel2.add(jPanel17, null);

this.add(jTabbedPanel,null);
this.add(jButton1, null);
this.add(jButton2, null);
D12.add(jRadioButton3);
D12.add(jRadioButton4);
D11.add(jRadioButton5);
D11.add(jRadioButton6);
D10.add(jRadioButton7);
D10.add(jRadioButton8);
D9.add(jRadioButton10);
D9.add(jRadioButton9);
D8.add(jRadioButton12);
D8.add(jRadioButton11);
D7.add(jRadioButton13);
D7.add(jRadioButton14);
D6.add(jRadioButton16);
D6.add(jRadioButton15);
D1.add(jRadioButton25);
D1.add(jRadioButton26);
D2.add(jRadioButton23);
D2.add(jRadioButton24);
D3.add(jRadioButton22);
D3.add(jRadioButton21);
D4.add(jRadioButton19);
D4.add(jRadioButton20);
D5.add(jRadioButton17);
D5.add(jRadioButton18);
M1.add(jRadioButton39);
M1.add(jRadioButton40);
M2.add(jRadioButton38);
```

```
    M2.add(jRadioButton37);
    M3.add(jRadioButton35);
    M3.add(jRadioButton36);
    M4.add(jRadioButton33);
    M4.add(jRadioButton34);
    M5.add(jRadioButton31);
    M5.add(jRadioButton32);
    M6.add(jRadioButton30);
    M6.add(jRadioButton29);
    M7.add(jRadioButton27);
    M7.add(jRadioButton28);
    queryDataSet2.setMasterLink(new
com.borland.dx.dataset.MasterLinkDescriptor(queryDataSet1, new
String[] {"QUES_ID", "ANSWER"}, new String[] {"QUES_ID", "ANSWER"},
false, true, false));


  }


void jButton1_actionPerformed(ActionEvent e) {
  }

    void jButton3_actionPerformed(ActionEvent e) {
    try{
        database1.saveChanges(queryDataSet1);
        database1.saveChanges(queryDataSet2);
        System.out.println("Save changes succeeded");
    }
    catch (Exception ex){
    DBExceptionHandler.handleException(ex);
    }
  }

  void jRadioButton39_actionPerformed(ActionEvent e) {

  }
  void jRadioButton39_itemStateChanged(ItemEvent e) {

  }
  void jRadioButton38_actionPerformed(ActionEvent e) {

  }
  void jRadioButton35_actionPerformed(ActionEvent e) {

  }
  void jRadioButton7_actionPerformed(ActionEvent e) {

  }
  void jRadioButton3_actionPerformed(ActionEvent e) {

  }
  void jRadioButton19_actionPerformed(ActionEvent e) {

  }
  void jRadioButton10_actionPerformed(ActionEvent e) {

  }
  void jRadioButton1_actionPerformed(ActionEvent e) {

  }
```

```
}
```

FormulateReuseValueNet.java

Created with JBuilder

```
package reuseagent1;

import javax.swing.*;
import javax.swing.JPanel;
import java.awt.event.*;
import java.awt.*;
import com.borland.jbcl.layout.*;
import com.borland.dbswing.*;
import com.borland.dx.sql.dataset.*;

//import com.borland.jbcl.layout.*;

/**
 * Title:        Reuse Agent Demonstration
 * Description:  The purpose of this application is to demonstrate
the functionality of the reuse agent.
 * Copyright:    Copyright (c) 2004
 * Company:      Loughborough University
 * @author
 * @version 1.0
 */

public class FormulateReuseValueNet extends JPanel{
  DataModule1 datamod = DataModule1.getDataModule();

  JSplitPane jSplitPane1 = new JSplitPane();
  JPanel jPanel1 = new JPanel();
  JPanel jPanel2 = new JPanel();
  BorderLayout borderLayout1 = new BorderLayout();
  JLabel jLabel1 = new JLabel();
  JLabel jLabel2 = new JLabel();
  JLabel jLabel3 = new JLabel();
  JLabel jLabel4 = new JLabel();
  JLabel jLabel5 = new JLabel();
  JLabel jLabel6 = new JLabel();
  JLabel jLabel7 = new JLabel();
  JLabel jLabel8 = new JLabel();
  JLabel jLabel9 = new JLabel();
  JLabel jLabel10 = new JLabel();
  JLabel jLabel11 = new JLabel();
  BorderLayout borderLayout2 = new BorderLayout();
  JdbNavToolBar jdbNavToolBar1 = new JdbNavToolBar();
  JdbStatusLabel jdbStatusLabel1 = new JdbStatusLabel();
  TableScrollPane tableScrollPane1 = new TableScrollPane();
  JdbLabel jdbLabel3 = new JdbLabel();
  JdbLabel jdbLabel4 = new JdbLabel();
  JdbLabel jdbLabel5 = new JdbLabel();
  JdbLabel jdbLabel6 = new JdbLabel();
  JdbLabel jdbLabel7 = new JdbLabel();
  JdbLabel jdbLabel8 = new JdbLabel();
  JdbLabel jdbLabel9 = new JdbLabel();
  JdbLabel jdbLabel10 = new JdbLabel();
  JdbLabel jdbLabel11 = new JdbLabel();
  JPanel jPanel3 = new JPanel();
  JLabel jLabel12 = new JLabel();
```

```
JdbTextArea jdbTextArea1 = new JdbTextArea();
Database database1 = new Database();
QueryDataSet queryDataSet1 = new QueryDataSet();

public FormulateReuseValueNet() {
  try {
    jbInit();
  }
  catch(Exception e) {
    e.printStackTrace();
  }

}
private void jbInit() throws Exception {
 jSplitPane1.setOrientation(JSplitPane.HORIZONTAL_SPLIT);
 jSplitPane1.setOneTouchExpandable(true);
 jSplitPane1.setDividerLocation(600);

  jPanel1.setBackground(new Color(151, 190, 255));
  jPanel1.setPreferredSize(new Dimension(500, 600));
  jPanel1.setLayout(borderLayout1);
  jPanel2.setPreferredSize(new Dimension(500, 500));
  jPanel2.setLayout(borderLayout2);

  jdbStatusLabel1.setText("jdbStatusLabel1");
  jdbStatusLabel1.setDataSet(queryDataSet1);
  jdbNavToolBar1.setDataSet(queryDataSet1);

  jdbLabel3.setFont(new java.awt.Font("Dialog", 1, 12));
  jdbLabel3.setText("Problem_Area :");
  jdbLabel3.setBounds(new Rectangle(38, 90, 95, 17));
  jdbLabel4.setFont(new java.awt.Font("Dialog", 1, 12));
  jdbLabel4.setText("Conflict :");
  jdbLabel4.setBounds(new Rectangle(40, 196, 53, 17));
  jdbLabel5.setFont(new java.awt.Font("Dialog", 1, 12));
  jdbLabel5.setText("Player :");
  jdbLabel5.setBounds(new Rectangle(40, 145, 50, 17));
  jdbLabel6.setFont(new java.awt.Font("Dialog", 1, 12));
  jdbLabel6.setText("Reuse_Process_Pattern :");
  jdbLabel6.setBounds(new Rectangle(35, 377, 155, 17));
  jdbLabel7.setFont(new java.awt.Font("Dialog", 1, 12));
  jdbLabel7.setText("Solution :");
  jdbLabel7.setBounds(new Rectangle(40, 254, 55, 17));

  jdbLabel8.setText("jdbLabel8");
  jdbLabel8.setColumnName("PROBLEM_AREA");
  jdbLabel8.setDataSet(queryDataSet1);
  jdbLabel8.setBounds(new Rectangle(59, 119, 149, 17));
  jdbLabel9.setText("jdbLabel9");
  jdbLabel9.setColumnName("PLAYER");
  jdbLabel9.setDataSet(queryDataSet1);
  jdbLabel9.setBounds(new Rectangle(62, 171, 154, 17));
  jdbLabel10.setText("jdbLabel10");
  jdbLabel10.setColumnName("CONFLICT");
  jdbLabel10.setDataSet(queryDataSet1);
  jdbLabel10.setBounds(new Rectangle(59, 223, 291, 17));
  jdbLabel11.setText("jdbLabel11");
  jdbLabel11.setColumnName("REUSE_PROCESS_PATTERN");
  jdbLabel11.setDataSet(queryDataSet1);
  jdbLabel11.setBounds(new Rectangle(59, 411, 277, 17));
  jPanel3.setLayout(null);
```

```
    jLabel12.setFont(new java.awt.Font("Dialog", 1, 12));
    jLabel12.setText("POSSIBLE SOLUTIONS");
    jLabel12.setBounds(new Rectangle(196, 6, 158, 17));
    jdbTextArea1.setBackground(new Color(144, 152, 160));
    jdbTextArea1.setPreferredSize(new Dimension(20, 35));
    jdbTextArea1.setText("jdbTextArea1");
    jdbTextArea1.setRows(5);
    jdbTextArea1.setColumnName("SOLUTION");
    jdbTextArea1.setDataSet(queryDataSet1);
    jdbTextArea1.setBounds(new Rectangle(56, 277, 312, 92));

    jPanel3.setPreferredSize(new Dimension(400, 300));

    database1.setConnection(new
com.borland.dx.sql.dataset.ConnectionDescriptor("jdbc:borland:dslocal
:C:\\shilpa\\ReuseSupportTool.jds", "ensd", "", false,
"com.borland.datastore.jdbc.DataStoreDriver"));
    queryDataSet1.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database1, "select
REUSEKNOWLEDGEBASE.CONFLICT, REUSEKNOWLEDGEBASE.PLAYER,
REUSEKNOWLEDGEBASE.PROBLEM_AREA, " +
      "REUSEKNOWLEDGEBASE.REUSE_PROCESS_PATTERN,
REUSEKNOWLEDGEBASE.SOLUTION " +
      "from REUSEKNOWLEDGEBASE, RESPONSEDATA\nwhere
REUSEKNOWLEDGEBASE.QUES_ID " +
      "= RESPONSEDATA.QUES_ID and RESPONSEDATA.ANSWER = \'NO\'",
null, true, Load.ALL));
    this.add(jSplitPane1, null);


    JInternalFrame frame = new JInternalFrame("Reuse Value Net",
true, true, true, true);
    Container container = frame.getContentPane();
    container.setBackground(new Color(110, 140, 253));
    ReuseValueNet rvn = new ReuseValueNet();
    rvn.setBackground(new Color(151,190,255));
    container.add(rvn, BorderLayout.CENTER);
    frame.pack();

    frame.setSize(400,400);
    frame.setVisible(true);

    jSplitPane1.add(jPanel2, JSplitPane.BOTTOM);
    jPanel2.add(jdbNavToolBar1,  BorderLayout.NORTH);
    jPanel2.add(jdbStatusLabel1, BorderLayout.SOUTH);
    jPanel2.add(tableScrollPane1, BorderLayout.WEST);
    jPanel2.add(jPanel3, BorderLayout.CENTER);
    jPanel3.add(jdbLabel3, null);
    jPanel3.add(jdbLabel5, null);
    jPanel3.add(jdbLabel9, null);
    jPanel3.add(jdbLabel8, null);
    jPanel3.add(jdbLabel4, null);
    jPanel3.add(jLabel12, null);
    jPanel3.add(jdbLabel10, null);
    jPanel3.add(jdbLabel7, null);
    jPanel3.add(jdbTextArea1, null);
    jPanel3.add(jdbLabel11, null);
    jPanel3.add(jdbLabel6, null);

    jSplitPane1.add(jPanel1, JSplitPane.TOP);
    jPanel1.add(frame,  BorderLayout.CENTER);
```

```
  }

}// end class FormulateReuseValueNet2

 class ReuseValueNet extends JPanel{
  DataModule1 datamod = DataModule1.getDataModule();
  JdbLabel jdbLabel_Player = new JdbLabel();


   public ReuseValueNet()
   {
   }

   // display Strings in different fonts and colors
   public void paint( Graphics g )
   {
      // call superclass's paint method
      super.paint( g );

       g.setColor(Color.black);

      // set current font to Ser
      //if (Times), bold, 12pt
      // and draw a string

        g.setFont( new Font( "Serif", Font.BOLD, 16 ) );
        g.drawString( " Customers", 220, 100 );

        g.setColor(Color.red);
        g.setFont( new Font( "Serif", Font.BOLD, 16 ) );
        g.drawString( "System Designers, ", 160, 120 );
        g.setColor(Color.black);

        g.setFont( new Font( "Serif", Font.PLAIN, 16 ) );
        g.drawString( " Organisation", 285, 120 );

        g.setFont( new Font( "Serif", Font.BOLD, 16 ) );
        g.drawString( " Competitors", 5, 220 );

        g.setFont( new Font( "Serif", Font.PLAIN, 16 ) );
        g.drawString( " Design from scratch, ", 5, 240 );

        g.setFont( new Font( "Serif", Font.PLAIN, 16 ) );
        g.drawString( " NIH-Syndrome, ", 5, 260 );

        g.setFont( new Font( "Serif", Font.PLAIN, 16 ) );
        g.drawString( " other Designers", 5, 280 );

        g.setFont( new Font( "Serif", Font.BOLD, 16 ) );
        g.drawString( " Complementors", 430, 220 );


        g.setColor(Color.blue);
        g.setFont( new Font( "Serif", Font.BOLD, 16 ) );
        g.drawString( " education(training) ", 425, 300 );
        g.setColor(Color.black);

        g.setFont( new Font( "Serif", Font.BOLD, 16 ) );
        g.drawString( " Suppliers", 220, 400 );
        g.setFont( new Font( "Serif", Font.PLAIN, 16 ) );
        g.drawString( " System Experts, ", 170, 420 );

        g.setFont( new Font( "Serif", Font.PLAIN, 16 ) );
        g.drawString( " Management", 280, 420 );

        g.setFont( new Font( "Serif", Font.BOLD, 16 ) );
        g.drawString( " Reuse-Driven MSD", 205, 250 );


      g.drawLine(260, 270, 260, 380);
      g.drawLine(260, 140, 260, 230);
      g.drawLine(100, 245, 195, 245);
      g.drawLine(345, 245, 420, 245);

      g.drawLine(110, 265, 205, 380);
      g.drawLine(340, 140, 420, 210);
      g.drawLine(110, 215, 180, 140);
      g.drawLine(345, 380, 420, 270);


      g.setFont( new Font( "Serif", Font.PLAIN, 16 ) );
      g.drawString( " Reuse tools and tech., ", 425, 240 );

      g.setColor(Color.blue);
      g.setFont( new Font( "Serif", Font.BOLD, 16 ) );
      g.drawString( " Knowledge Bases, ", 425, 260 );

      g.setFont( new Font( "Serif", Font.BOLD, 16 ) );
      g.drawString( " Reuse Library, ", 425, 280 );

      g.setColor(Color.black);

    }

}
```

SearchPattern.java

Created with JBuilder

```
package reuseagent1;

import javax.swing.*;
import java.awt.*;
import com.borland.dbswing.*;
import com.borland.dx.sql.dataset.*;
import com.borland.dx.dataset.*;
import java.awt.event.*;



/**
 * Title:        Reuse Agent
 * Description:
 * Copyright:    Copyright (c) 2004
 * Company:      Loughborough University
 * @author
 * @version 1.0
 */
```

```
public class SearchPattern extends JPanel{
  Database database1 = new Database();
  QueryDataSet queryDataSet1 = new QueryDataSet();
  BorderLayout borderLayout1 = new BorderLayout();
  JdbNavToolBar jdbNavToolBar1 = new JdbNavToolBar();
  JdbStatusLabel jdbStatusLabel1 = new JdbStatusLabel();
  JPanel jPanel1 = new JPanel();
  JdbLabel jdbLabel1 = new JdbLabel();
  JdbLabel jdbLabel2 = new JdbLabel();
  JdbLabel jdbLabel3 = new JdbLabel();
  JdbLabel jdbLabel4 = new JdbLabel();
  JdbLabel jdbLabel5 = new JdbLabel();
  JdbTextArea jdbTextArea1 = new JdbTextArea();
  JdbLabel jdbLabel6 = new JdbLabel();
  ParameterRow parameterRow1 = new ParameterRow();
  JLabel jLabel1 = new JLabel();
  JList jList2 = new JList();
  JdbList jdbList1 = new JdbList();
  JdbComboBox jdbComboBox1 = new JdbComboBox();
  JButton jButton1 = new JButton();
  JTextField jTextField1 = new JTextField();
  Column column1 = new Column();
  JdbComboBox jdbComboBox2 = new JdbComboBox();

  public SearchPattern() {
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
  private void jbInit() throws Exception {
    this.setLayout(borderLayout1);
    queryDataSet1.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database1, "select * from
PROCESSPATTERNCATALOGUE\nwhere PATTERN_NAME = :pattern_name\n" +
      " ", parameterRow1, true, Load.AS_NEEDED));
    database1.setConnection(new
com.borland.dx.sql.dataset.ConnectionDescriptor("jdbc:borland:dslocal
:C:\\shilpa\\ReuseSupportTool.jds", "ensd", "", false,
"com.borland.datastore.jdbc.DataStoreDriver"));
    jdbStatusLabel1.setText("jdbStatusLabel1");
    jdbStatusLabel1.setDataSet(queryDataSet1);
    jdbNavToolBar1.setDataSet(queryDataSet1);
    jPanel1.setLayout(null);
    jdbLabel1.setFont(new java.awt.Font("Dialog", 1, 15));
    jdbLabel1.setText("Pattern_Name :");
    jdbLabel1.setBounds(new Rectangle(18, 107, 111, 17));
    jdbLabel2.setFont(new java.awt.Font("Dialog", 1, 15));
    jdbLabel2.setText("Pattern_Type :");
    jdbLabel2.setBounds(new Rectangle(24, 143, 108, 17));
    jdbLabel3.setFont(new java.awt.Font("Dialog", 1, 15));
    jdbLabel3.setText("Purpose :");
    jdbLabel3.setBounds(new Rectangle(58, 178, 76, 17));
    jdbLabel4.setFont(new java.awt.Font("Dialog", 0, 14));
    jdbLabel4.setText("jdbLabel4");
    jdbLabel4.setColumnName("PATTERN_NAME");
    jdbLabel4.setDataSet(queryDataSet1);
    jdbLabel4.setBounds(new Rectangle(145, 110, 344, 17));
    jdbLabel5.setFont(new java.awt.Font("Dialog", 0, 14));
    jdbLabel5.setText("jdbLabel5");
    jdbLabel5.setColumnName("PATTERN_TYPE");
    jdbLabel5.setDataSet(queryDataSet1);
    jdbLabel5.setBounds(new Rectangle(145, 144, 180, 17));
    jdbTextArea1.setBackground(new Color(144, 152, 160));
    jdbTextArea1.setFont(new java.awt.Font("Dialog", 0, 14));
    jdbTextArea1.setPreferredSize(new Dimension(40, 17));
    jdbTextArea1.setText("jdbTextArea1");
    jdbTextArea1.setColumnName("PURPOSE");
    jdbTextArea1.setDataSet(queryDataSet1);
    jdbTextArea1.setBounds(new Rectangle(141, 182, 229, 74));
    jdbLabel6.setFont(new java.awt.Font("Dialog", 1, 16));
    jdbLabel6.setText("SEARCH RESULT");
    jdbLabel6.setBounds(new Rectangle(85, 66, 165, 17));
    jLabel1.setFont(new java.awt.Font("Dialog", 1, 14));
    jLabel1.setText("Select Pattern :");
    jLabel1.setBounds(new Rectangle(31, 26, 116, 17));

    jdbComboBox1.setItems(new String[] {"Build a reuse library",
"Create reusable component", "Select reusable component"});
    jdbComboBox1.setBounds(new Rectangle(173, 22, 195, 21));
    jButton1.setFont(new java.awt.Font("Dialog", 1, 12));
    jButton1.setText("Search");
    jButton1.setBounds(new Rectangle(340, 23, 79, 27));
    jButton1.addActionListener(new java.awt.event.ActionListener() {
      public void actionPerformed(ActionEvent e) {
        jButton1_actionPerformed(e);
      }
    });
    jTextField1.setText("jTextField1");
    jTextField1.setBounds(new Rectangle(161, 25, 123, 21));
    column1.setColumnName("pattern_name");
    column1.setDataType(com.borland.dx.dataset.Variant.STRING);
    column1.setServerColumnName("NewColumn1");
    column1.setSqlType(0);
    parameterRow1.setColumns(new Column[] {column1});
    jdbComboBox2.setToolTipText("");
    jdbComboBox2.setItems(new String[] {"Select reusable components",
"Create reusable component", "Determine the types of components to
search for",
        "Determine the source of reusable components", "Look for
existing reusable systems", "Determine the availability of
resources",
        "Search reuse library and catalogue", "Search for reusable
components", "Selected reusable component report",
        "Generalise component", "Standardise component", "Automate
component", "Certify component", "Prepare component",
        "Build reuse library", "Create reuse documentation for the
component", "Define the types of components to be stored",
        "Plan for growing reuse library", "Select components",
"Create reusable component documentation", "Design physical storage
for the library",
        "Design logical structure of the library", "Define the
classification scheme", "Select reuse support tool",
        "Set up a reuse catalogue", "Define reuse library personnel
support"});
    jdbComboBox2.setBounds(new Rectangle(147, 24, 177, 21));
    this.add(jdbNavToolBar1, BorderLayout.NORTH);
    this.add(jdbStatusLabel1, BorderLayout.SOUTH);
    this.add(jPanel1, BorderLayout.CENTER);
    jPanel1.add(jdbLabel3, null);
    jPanel1.add(jdbLabel2, null);
    jPanel1.add(jdbLabel1, null);
    jPanel1.add(jdbLabel6, null);
    jPanel1.add(jLabel1, null);
    jPanel1.add(jList2, null);
    jPanel1.add(jdbList1, null);
    jPanel1.add(jdbTextArea1, null);
    jPanel1.add(jdbLabel5, null);
    jPanel1.add(jdbLabel4, null);
    jPanel1.add(jdbComboBox2, null);
    jPanel1.add(jButton1, null);
  }

  void jButton1_actionPerformed(ActionEvent e) {
    try {
            parameterRow1.setString("pattern_name",
(String)(jdbComboBox2.getSelectedItem()));
            queryDataSet1.refresh();
    }
    catch (Exception ex){
      ex.printStackTrace();
    }
  }
}
```

DisplayPattern.java

Created with JBuilder

```
package reuseagent1;

import java.awt.*;
import java.awt.event.*;
import javax.swing.*;
import javax.swing.JOptionPane;

/**
 * Title:        Reuse Agent
 * Description:
 * Copyright:    Copyright (c) 2004
 * Company:      Loughborough University
 * @author
 * @version 1.0
 */


class DisplayPattern extends JPanel {
   private ImageIcon imageIcon;
   String patternName;
   // load image
   public DisplayPattern()
   {
      patternName = JOptionPane.showInputDialog("Enter Pattern
Name");

      imageIcon = new ImageIcon( "e:/Reuse Process Pattern
Library/"+patternName+".png" );
    }

   // display imageIcon on panel
   public void paintComponent( Graphics g )
   {
      // call superclass paintComponent method
      super.paintComponent( g );

      // display icon
      imageIcon.paintIcon( this, g, 0, 0 );
   }

   // return image dimensions
   public Dimension getPreferredSize()
   {
      return new Dimension( imageIcon.getIconWidth(),
         imageIcon.getIconHeight() );
   }

}  // end class MyJPanel
```

DataModule1.java

Created with JBuilder

```
package reuseagent1;

import java.awt.*;
import java.awt.event.*;
import com.borland.dx.dataset.*;
import com.borland.dx.sql.dataset.*;
import com.borland.dbswing.*;
/**
 * Title:        Reuse Agent
 * Description:
 * Copyright:    Copyright (c) 2004
 * Company:      Loughborough University
 * @author S.Dani
 * @version 1.0
 */

public class DataModule1 implements DataModule {
  private static DataModule1 myDM;
  Database database1 = new Database();
  QueryDataSet reuseknowledgebase = new QueryDataSet();
  JdbStatusLabel jdbStatusLabel1 = new JdbStatusLabel();
  public static DataModule1 getDataModule() {
    if (myDM == null) {
      myDM = new DataModule1();
    }
    return myDM;
  }

  public DataModule1() {
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
  private void jbInit() throws Exception {
    jdbStatusLabel1.setText("jdbStatusLabel1");
    jdbStatusLabel1.setDataSet(reuseknowledgebase);
    reuseknowledgebase.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database1, "SELECT
REUSEKNOWLEDGEBASE.ANSWER,REUSEKNOWLEDGEBASE.PROBLEM_AREA,REUSEKNOWLE
DGEBASE.PLAYER,REUSEKNOWL" +

"EDGEBASE.CONFLICT,REUSEKNOWLEDGEBASE.REUSE_PROCESS_PATTERN,REUSEKNOW
LEDGEBASE.SOLUTION " +
      "FROM REUSEKNOWLEDGEBASE WHERE REUSEKNOWLEDGEBASE.ANSWER=\'NO\'
GROUP " +
      "BY REUSEKNOWLEDGEBASE.CONFLICT ORDER BY
REUSEKNOWLEDGEBASE.PLAYER", null, true, Load.ALL));
    database1.setConnection(new
com.borland.dx.sql.dataset.ConnectionDescriptor("jdbc:borland:dslocal
:C:\\shilpa\\ReuseSupportTool.jds", "ensd", "", false,
"com.borland.datastore.jdbc.DataStoreDriver"));
  }
  public Database getDatabase1() {
    return database1;
  }
  public QueryDataSet getReuseknowledgebase() {
    return reuseknowledgebase;
  }
}
```

DataMod_MasterDetail.java

Created with JBuilder

```
package reuseagent1;

import java.awt.*;
import java.awt.event.*;
import com.borland.dx.dataset.*;
import com.borland.dx.sql.dataset.*;

/**
 * Title:        Reuse Agent
 * Description:
 * Copyright:    Copyright (c) 2004
 * Company:      Loughborough University
 * @author
 * @version 1.0
 */

public class DataMod_MasterDetail implements DataModule {
  private static DataMod_MasterDetail myDM;
  Database database1 = new Database();
  QueryDataSet responsedata = new QueryDataSet();
  Database database2 = new Database();
  QueryDataSet queryDataSet1 = new QueryDataSet();
  ParameterRow parameterRow1 = new ParameterRow();
  Column column1 = new Column();
  public static DataMod_MasterDetail getDataModule() {
    if (myDM == null) {
      myDM = new DataMod_MasterDetail();
    }
    return myDM;
  }

  public DataMod_MasterDetail() {
    try {
      jbInit();
    }
    catch(Exception e) {
      e.printStackTrace();
    }
  }
  private void jbInit() throws Exception {
    column1.setColumnName("ANSWER");
    column1.setDataType(com.borland.dx.dataset.Variant.STRING);
    column1.setServerColumnName("NewColumn1");
    column1.setSqlType(0);
    queryDataSet1.setQuery(new
com.borland.dx.sql.dataset.QueryDescriptor(database2, "select * from
REUSEKNOWLEDGEBASE where REUSEKNOWLEDGEBASE.ANSWER " +
    "\nwhere REUSEKNOWLEDGEBASE.ANSWER = :ANSWER",responsedata, true,
Load.AS_NEEDED));
    database2.setConnection(new
com.borland.dx.sql.dataset.ConnectionDescriptor("jdbc:borland:dslocal
:C:\\shilpa\\ReuseSupportTool.jds", "ensd", "", false,
"com.borland.datastore.jdbc.DataStoreDriver"));

    database1.setConnection(new
com.borland.dx.sql.dataset.ConnectionDescriptor("jdbc:borland:dslocal
:C:\\shilpa\\ReuseSupportTool.jds", "ensd", "", false,
"com.borland.datastore.jdbc.DataStoreDriver"));

    queryDataSet1.setMasterLink(new
com.borland.dx.dataset.MasterLinkDescriptor(responsedata, new
String[] {"ANSWER"}, new String[] {"ANSWER"}, true, false, false));
    parameterRow1.setColumns(new Column[] {column1});
  }

  public Database getDatabase1() {
    return database1;
  }

  public QueryDataSet getResponsedata() {
    return responsedata;
  }

  public com.borland.dx.sql.dataset.Database getDatabase2() {
    return database2;
  }

  public com.borland.dx.sql.dataset.QueryDataSet getQueryDataSet1() {
    return queryDataSet1;
  }

  public com.borland.dx.dataset.ParameterRow getParameterRow1() {
    return parameterRow1;
  }
}
```