AN ARCHITECTURE TO SUPPORT VIRTUAL CONCURRENT ENGINEERING

by

Martin Hanneghan B.Sc. (Hons)

A thesis submitted in partial fulfilment of the requirements of Liverpool John Moores University for the degree of Doctor of Philosophy

Liverpool John Moores University
School of Engineering

June, 1998

The following table and figures have been omitted on request of the university –

Table 1.1 (p.4)

Fig 2.3 (p.20)

Fig 2.4 (p.27)

Fig 2.5 (p.30)

Fig 3.1 (p.37)

Fig 3.2 (p.39)

Fig 3.3 (p.41)

Fig 3.4 (p.42)

Fig 3.5 (p.43)

Fig 3.6 (p.45)

Fig 4.3 (p.61)

Fig 4.4 (p.62)

Fig 4.5 (p.63)

Fig 8.1 (p.111)

Fig 8.2 (p.112)

ABSTRACT

The continual pressure on manufacturers to be globally competitive by increasing productivity while reducing costs and development time has been the driving force behind the philosophy of Concurrent Engineering (CE). CE aims to systematically integrate all aspects of the entire product life cycle to enable competitive advantage. Ironically, contemporary CE support tools have only served to intensify the problems associated with this integration due to incompatibilities from equipment and tool heterogeneity, lack of support for the operation, management and co-ordination of globally dispersed teams and the lack of globally available information sharing practices that aid concurrent development. This thesis investigates methodologies and technologies that are suitable for application in information technology-based support environments for Concurrent Engineering.

In an effort to solve these problems, this thesis draws on aspects from many diverse fields and presents a new architecture that unifies and integrates these multiple disciplines to provide a robust, computer-based environment for supporting a Concurrent Engineering strategy. At the core of this new architecture is a viewpoint-based reference model for Concurrent Engineering (CE-RM) that examines and supports the multiple views of all aspects in a CE project and which provides a framework for the development of computer systems to support CE. A computer systems middleware architecture (called CONCERT — CONCurrent Engineering suppoRT) and an object-oriented information system (IS) model have been devised using this framework. CONCERT enables global participation and interaction while the IS model captures the information requirements of organisations employing a CE strategy. A prototype support environment based on the CONCERT architecture has been built using the Java language and is described with two case studies that serve to show the validity and feasibility of the architecture. This thesis also highlights how the practice of Concurrent Engineering has many parallels with other problem domains and explores how these complementary domains can be supported using the proposed architecture.

ACKNOWLEDGMENTS

I am truly indebted to a number of people, without whom this thesis would not exist.

Firstly, I would like to thank my two supervisors: Madjid Merabti and Gary Colquhoun, whose excellent guidance throughout this research has been invaluable. Their encouragement and support to go forth and publish my ideas has been a great impetus at all times. In addition, their direction and enthusiasm have certainly helped me develop my research and writing skills. In the dual role of supervisor and friend they have been a constant source of inspiration, support and shoulder to lean on. This thesis is a testament to their talent.

I am also indebted to the academic, technician and research staff (both past and present) from the School of Computing and Mathematical Sciences at Liverpool John Moores University for their support, encouragement and hospitality throughout this research.

Finally, my sincere thanks go to my family and my fiancée, Joanne. Their constant interest in my efforts, and faith in my ability, has proved a continual source of inspiration to me.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACRONYMS

BPR	Business Process Reengineering
CAD	Computer-Aided Design
CAE	Computer-Aided Engineering
CAM	Computer-Aided Manufacturing
CE	Concurrent Engineering
CE-RM	Concurrent Engineering Reference Model
CGI	Common Gateway Interface
CIM	Computer Integrated Manufacturing
CIM-OSA	Open Systems Architecture for Computer Integrated Manufacturing
CONCERT	CONCurrent Engineering suppORT
CORBA	Common Object Request Broker Architecture
CSCW	Computer-Supported Co-operative Working
DBMS	Database Management System
DCE	Distributed Computing Environment
DFA	Design For Assembly
DFM	Design For Manufacture
FMS	Flexible Manufacturing Systems
GUI	Graphical User Interface
HTML	HyperText Mark-up Language
HTTP	HyperText Transfer Protocol
IDL	Interface Definition Language
IGES	Initial Graphics Exchange Standard
IIOP	Internet Inter-ORB Protocol
IPD	Integrated Product Development
JIT	Just-In-Time
JVM	Java Virtual Machine
LAN	Local Area Network
MAP	Manufacturing Automation Protocol
MRP	Material Requirements Planning
MRPII	Manufacturing Resource Planning
NIIIP	National Industrial Information Infrastructure Protocols
ODBMS	Object Database Management System
ODP	Open Distributed Processing
OMA	Object Management Architecture
OMG	Object Management Group
OPT	Optimised Production Technology
OQL	Object Query Language
ORB	Object Request Broker
RMI	Remote Method Invocation
RM-ODP	Reference Model for Open Distributed Processing
RPC	Remote Procedure Call
SE	Simultaneous Engineering
SQL	Structured Query Language
STEP	Standard for The Exchange of Product model data
TCP/IP	Transmission Control Protocol / Internet Protocol
TOP	Technical and Office Protocol
TQM	Total Quality Management
URL	Uniform Resource Locator
WWW	World-Wide Web

Chapter 1

"People can have the Model T in any colour—so long as it's black."
Henry Ford, automotive pioneer, describing the extent of customer choice and involvement in the production of his Model T motor car in 1908.

1. Introduction

1.1 Preamble

This research has its origin in the now global initiative to dramatically improve the product development process in the area of product development and manufacturing known as Concurrent Engineering (CE). Unlike our ancestors in the quotation above, modern-day consumers have a much greater say in the goods they buy. Contrast the statement above with Ford's current practice (Hobby 1997) of bespoke manufacturing in which vehicles are assembled with the help of computer-generated images depicting virtual motorists. Modern manufacturing organisations have to be able to accommodate consumers' needs more efficiently in order to retain or capture market share.

This chapter provides a context for this research describing the market forces that have led to the adoption of Concurrent Engineering and in particular, the problems facing organisations employing a CE strategy. The objectives of the work are defined and the methodology is explained. The chapter concludes by summarising the novel outcomes of this work and provides a structure for the remainder of the thesis.

1.2 Competition and the global market

Today, just as in the past (Smith 1997) manufacturing organisations must contend with, and aim to survive longer than, their competitors. Competition exists in various forms: organisations selling similar product lines, aggressive pricing or mass production stealing substantial market share. In the past, organisations have only had to endure competition from peers in their own country. However with the advent of cheaper telecommunications and improved transportation routes between countries[1] and even continents, organisations are now faced with competition from other highly developed countries that share technical

[1] An obvious example of this can be seen in the Channel Tunnel linking Britain to France.

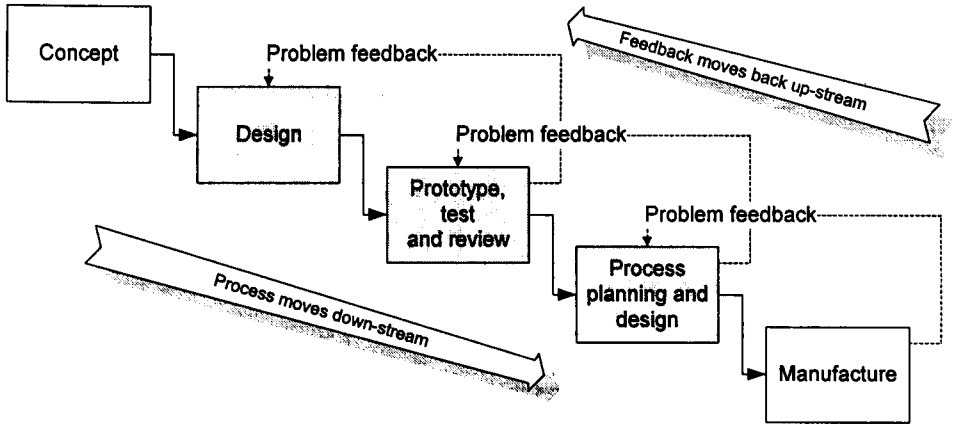


Figure 1-1. A traditional sequential development process.

know-how and are aided by a healthy economy. This is in addition to the threat from third-world countries that have a large, cheap labour force that can be exploited by large multinational organisations and their highly flexible manufacturing plants. Competitive advantage in modern organisations can therefore lie in what (Drucker 1991) describes as 'working smarter,' i.e. making better use of available resources.

Prasad (p.8-9, 1996) lists a number of factors that have led to the need for governments as well as industries to change their way of working. These include decreasing defence budgets, an evolving, highly competitive marketplace, recent technological advances and closely aligned world economics and trade structures. One important area of research into improving the competitiveness of manufacturing organisations has concentrated on the product development process and how this can be better applied.

1.3 The product development process

1.3.1 Sequential development

Traditionally, the development of manufactured products has followed the sequential process[2] as shown in Figure 1-1. The analogy of this in the field of Software Engineering, is what is known as the classic life cycle or 'waterfall model' (Pressman 1997; Royce 1970; Sommerville 1996) because of the way the product development cascades from one phase to another. Communication occurs only when a particular stage in the process is completed and this "hand-shaking" process (Evans 1988) iterates until all parties are satisfied with the result. Changes are inevitable during the design and development of any product. In a sequential development process, down-stream problems are fed back up to the previous development phase to be corrected in an iterative fashion through the disciplined use of

[2] The nature and number of tasks shown in Figure 1-1 are hypothetical and are used simply to explain the concept.

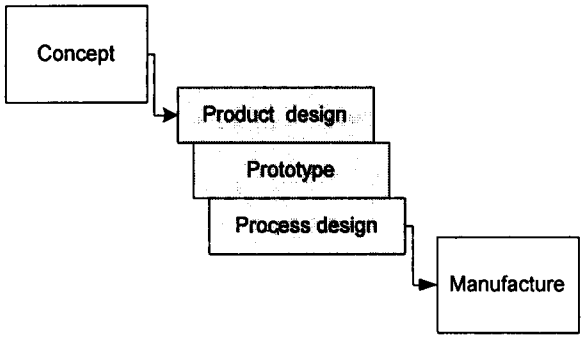


Figure 1-2. Concurrent engineering development.

Engineering Changes. The cost of this reworking can amount to a substantial proportion of the total development cost. Gatenby and Foo (1990) estimate that around 80 to 90 percent of the total life cycle cost is determined during the design phase even though the design phase itself may only account for around 5 to 10 percent of the total product development cost.

1.3.2 Concurrent Engineering

Concurrent Engineering differs from a traditional sequential development process in that it attempts to run as many downstream tasks as possible in parallel (see Figure 1-2). This requires that each process has almost immediate feedback from down-stream processes and similarly those down-stream processes begin as soon as possible using early and accurate information from up-stream processes. Nickerson (1990) describes the difference between sequential and concurrent engineering as "...not what is done but when it is done."

The philosophy of Concurrent Engineering (CE) has existed in various forms since the beginning of the twentieth century (Smith 1997). It is well documented that the pioneers of the automotive industry such as Henry Ford, Karl Benz and Ransom Olds applied principles which today would be characterised as Concurrent Engineering (Jo et al. 1991). In 1908, Ford's Model T motor car was a prime example of radically different manufacturing techniques that required not only a redesigned product but also a redesigned manufacturing process. This combined product and process engineering is the essence of CE. The Model T was relatively inexpensive to both manufacture and own, and thus enabled the average person to own a motor car. Ford sold around 15 million cars before the Model T was discontinued in 1927. The techniques pioneered by Ford were soon adopted by many other industries. (For a more detailed survey of CE history the reader is referred to Smith 1997).

CE is a manufacturing philosophy as opposed to a technology (Jo et al. 1993) which has emerged as one of the many new philosophies that is changing the way that UK and World-Wide manufacturers organise and control manufacturing in an attempt to remain globally

competitive. Other manufacturing strategies include Kanban and Just-In-Time (JIT) (Gupta and Brennan 1995; Yavuz and Satir 1995), Flexible Manufacturing Systems (FMS) (Wu 1995), and Optimised Production Technology (OPT) (Yenradee 1994). Some best-practice approaches that have been applied to industries other than manufacturing include Business Process Reengineering (BPR) (Hammer and Champy 1993) and Total Quality Management (TQM) (Johnson and Kleiner 1995).

The philosophy of concurrent engineering has been successfully applied to a number of different industries. Case studies have been reported in the automotive industry, Government agencies such as the defence industry, the aerospace industry, equipment manufacturing and the electronics industry amongst others (Coupland 1992; Zhang and Zhang 1995). Gatenby et al. (1994) report some dramatic results on the benefits that can be achieved by applying a well disciplined Concurrent Engineering strategy. These results are shown in Table 1-1.

1.4 Definitions of Concurrent Engineering

There have been a number of definitions of Concurrent Engineering cited in recent literature. They do however, all share a common theme stressing the need for team co-operation and parallel development throughout the entire product life cycle. Some common definitions of CE are given below.

> "...a systematic approach to the integrated concurrent design of products and their related processes, including manufacture and support. This approach is intended to cause the developers, from the outset to consider all elements of the product life cycle from conception through disposal, including quality, cost, scheduling and user requirements," (Winner et al. 1988).

> "...the concurrent development of project design functions, with open and interactive communication existing between all team members for the purpose of reducing lead time from concept to production launch," (McKnight and Jackson 1989).

"...a systematic approach to integrated product development that emphasises response to customer expectations and embodies team values of co-operation, trust and sharing in such a manner that decision making proceeds with large intervals of parallel working by all life cycle perspectives early in the process, synchronised by comparatively brief exchanges to produce consensus," (Cleetus 1992).

"...the delivery of better, cheaper, faster products to market, by a lean way of working, using multi-discipline teams, right first time methods and parallel processing activities to continuously consider all constraints," (Lettice et al. 1995).

"...an holistic methodology for the co-ordination of distributed, heterogeneous expertise to achieve cost effective, market-driven products in minimum time scales," (Harding and Popplewell 1996).

The process of Concurrent Engineering has been described using many synonyms. These synonyms include *engineering* such as Simultaneous Engineering (SE), life cycle engineering, parallel engineering and producibility engineering; *integration* such as Integrated Product Development (IPD), concurrent product and process design, integrated and co-operative design and design fusion; and *design for X* such as design for production, design for assembly and design for manufacturability.

1.5 Concurrent Engineering approaches

There have been many different approaches to Concurrent Engineering developed in the last decade (Dowlatshahi 1994). These range from pure technology-based programmes which use tools such as information systems, software systems and artificial intelligence through semi-technologically based programmes which aim to integrate Computer-Aided Design (CAD) and Computer-Aided Manufacturing (CAM) with more traditional approaches such as organisational and cultural changes and techniques such as Design for Manufacturabilty (DFM) and Design for Assembly (DFA). Molina et al. (1995a) suggest four 'enabling technologies' that can be used to support CE which include modelling methodologies, computer-aided decision support, information architectures and frameworks for CE environment development.

This research focuses on the technology-based approach to CE which looks at how computer systems can be made to inter-operate throughout the entire product life cycle and how they can be used to better support the Concurrent Engineering process. In the perspective of Molina et al. (1995a) this thesis considers information system architectures and frameworks for CE.

1.6 Problems associated with technology-based CE approaches

Practitioners of Concurrent Engineering have encountered a number of problems. These have emerged as a result of the new working practices that CE requires such as information sharing, wide geographically located teams and the problems associated with integrating inherently heterogeneous systems.

1.6.1 Sharing information effectively

Co-operative teams need to share information effectively in order to reduce data wastage. Lewis et al. (1994) reports how sharing incompatible data among groups commonly requires data re-entry or translation which leads to data redundancy, data duplication, data transcription errors and data that is second-hand. This is largely due to the reliance on legacy computer systems that are not capable of sharing information (Nickerson 1990) either because they were not built with this feature in mind or because the data they contain is not in a standardised format. These "islands of automation" (Koonce 1995) can seriously hinder the CE process.

1.6.2 Widely dispersed teams

Companies employing CE come in all shapes and sizes. Using conventional means, factors such as the ease with which team members can contact each other and with which meetings can be arranged are all affected by the geographic dispersion of the organisation. This has become increasingly significant as more and more companies employ CE teams with members not only at different sites but also in different countries spanning different time zones (Scrivener et al. 1995). The resultant difficulty involved in managing such a widely dispersed non-collocated team is therefore higher than for collocated teams (Harding and Popplewell 1996; Powell 1996) (see Figure 1-3.)

1.6.3 Heterogeneity

Large computer networks such as those used in modern manufacturing organisations are inherently heterogeneous due to a number of factors. These include:

- *The rapid pace of technological change*—the average processing power of a microprocessor doubles roughly every 18 months according to Moore's Law[3]. This leads to organisations upgrading hardware constantly in an attempt to remain competitive (for a survey of the heterogeneity of systems in CE, see Kannan et al. 1992).

[3] The observation by semiconductor engineer Gordon Moore in 1964, that the amount of information storable on a given amount of silicon has roughly doubled every year since the technology was invented. This relation held until the late 1970s at which point the doubling period slowed to about eighteen months.

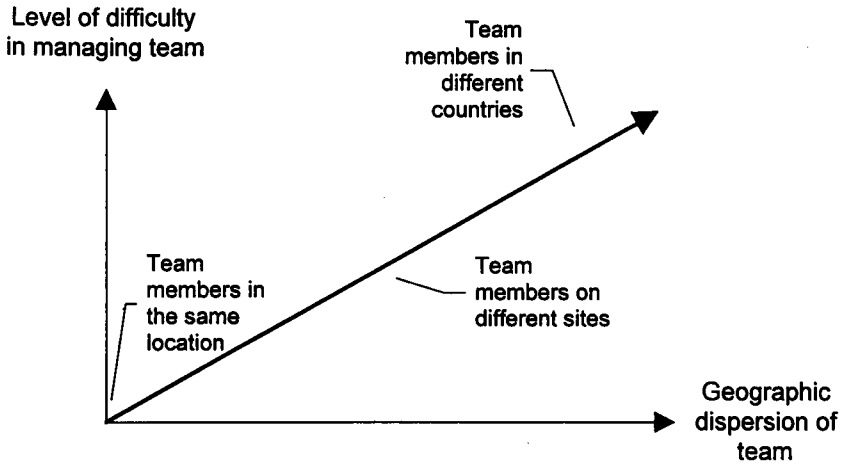


Figure 1-3. The effect of geographic location on CE teams.

- *Reliance on legacy systems*—for example a company that relies heavily on a mainframe application to handle MRPII will be reluctant to scrap it. Also, organisations may have spent large sums of money on existing systems that must be utilised at least until the investment has paid off.

- *Engineering trade-offs* (Vinoski 1997)—the fact that there is usually more than one acceptable solution to a complex engineering problem can lead to different departments within an organisation choosing different solutions to similar problems.

Farooqui et al. (1995) also categorises types of heterogeneity to include equipment, operating systems, computational (programming or database) languages, software applications and authorities (where interaction between autonomous ownership domains is required).

1.7 Definitions of keywords in the context of this research

The following terms will be referred to throughout this thesis, and are therefore defined below in the context of this research.

Architecture

In the context of software engineering, architecture is the formal structure that can be used to integrate all the elements necessary to develop a complete software system. The architecture is the combination of system and data models along with definition of the procedures that link each of these components.

Environment

An environment is the realisation of an architecture and supporting systems. In the case of realising a computer system architecture, the environment would be the combination of software and hardware required to implement the architecture.

Cross-functional team

A cross-functional team is a team composed of representative members from as many aspects of the entire product life cycle as possible. In contrast to, say, a design team that is made up purely of designers, a cross-functional team might include members from design, production planning, production, marketing, sales, finance, maintenance, quality control and disposal to name but a few.

Virtual working

Virtual working is hereby defined as the use of computer-based communication systems to facilitate normal day-to-day working practices for non-collocated workers.

1.8 Scope of the research

This research is concerned with facilitating the Concurrent Engineering process from an information and collaboration perspective, from initial product conception through to disposal. It focuses on the problems of virtual team working and the underlying infrastructure required for supporting virtual Concurrent Engineering.

The field of Computer Supported Co-operative Working (CSCW) is an interesting research area that impinges on technology-based companies practising CE. This research acknowledges the contribution of CSCW on CE and incorporates many of the characteristics of CSCW in solving some of the problems encountered by CE practitioners. It does not however, focus on the mechanisms used or the social implications of using CSCW technology.

The research that surrounds the area of generic product information models for enabling CE such as STEP (STandard for the Exchange of Product model data; ISO 1992b) is not the focus of this research, although this work does consider the application of such information models.

1.9 Aims and objectives

The research aims to answer a number of questions posed by CE practitioners. These include:

1. What aspects of current computer technology are useful to Concurrent Engineering?
2. What new technologies are needed to support CE?
3. What frameworks exist for supporting CE?
4. What components are useful building blocks for next-generation frameworks?

The aim of this research is to present a global computing and information system architecture that will facilitate the day-to-day running of Concurrent Engineering projects. By providing a means to efficiently undertake CE, it is believed that the dramatic reductions in product development time and cost, along with improvements in product quality that are attributed to CE, can at least be sustained and even improved upon. To achieve this aim the following objectives will be pursued:

a) Acquire an understanding of current CE product development processes.
b) Review current computer-based systems used in manufacturing.
c) Develop a computer systems architecture that can be applied to manufacturing organisations globally.
d) Develop an information systems model that can be applied to CE projects.
e) Effectively demonstrate the complete system in a live application.

1.10 Research methodology

To achieve the above objectives, the research methodology outlined below will be adopted:

a) Compile a state-of-the-art review of techniques that are used by CE practitioners.
b) Investigate information systems that have been applied to manufacturing and / or CE problems.
c) Develop a state-of-the-art review of recent technological advances that can be used to facilitate CE.
d) Investigate information systems and computer systems modelling techniques. Database approaches, data modelling methods and systems building will be analysed in the context of the unique requirements of CE.
e) Develop a computer systems architecture and information systems model that can be used to facilitate CE projects.
f) Build a prototype CE support environment based on these models. These models will be realised in a live software application suite.
g) Validate the environment by application in a number of case studies.

1.11 Contribution to knowledge through this research

The novel aspects of the work described in this thesis are:

1. A reference model for Concurrent Engineering (CE-RM) that can be used as a foundation in the design and building of distributed computer-based support environments has been devised (Hanneghan et al. 1998), (discussed in Chapter 4).
2. A computer systems architecture (the CONCERT architecture) that fulfils the requirements of the CE-RM and which can be used to facilitate a Concurrent Engineering strategy has been developed (Hanneghan et al. 1996a) and implemented (Hanneghan et al. 1997), (discussed in Chapter 5).
3. An information system model has been developed that can be used to capture necessary information associated with CE projects (Hanneghan et al. 1995), (discussed in Chapter 6).
4. This work is the result of a synthesis of a number of disparate research fields including:
   a) Concurrent Engineering.
   b) Global Manufacturing.
   c) Inter-operable computer systems.
   d) Computer-Supported Co-operative Working (CSCW).
   e) Data and database management (discussed in Chapter 2).
5. A global perspective has been assumed in analysing the CE process. This has taken into account the fact that modern organisations need to co-operate with other organisations that may be in different countries. Organisations may themselves be composed of a number of departments that span an entire continent.
6. The system has been designed as a generic tool that can be applied to many diverse problem domains, not just Concurrent Engineering. As such, the system does not enforce any particular product information model standard on a project[4], instead leaving that decision to the Project Manager. This means that the system can been more easily integrated into existing projects without causing product specifications to be re-worked, (discussed in Chapter 8).
7. The work has produced an open environment on which to base future research. This openness is a result of the flexibility and extensibility of the design that is uniformly modelled using object-oriented methods, (discussed in Chapter 9).

[4] It is, however, the opinion of the author that the STEP protocol is an obvious choice to enforce for manufacturing-based projects.

8. The implementation of the design is a novel use of the Internet (Hanneghan et al. 1996b; Hanneghan et al. 1996c) which can be safely applied to an Internet environment in all its current guises: i.e. the Internet, Intranets and Extranets, (discussed in Chapter 7).

1.12 Summary

This introductory chapter has described the reasons why organisations have been compelled to look for improvements in working practices due to increasing global competitiveness. This chapter has described the notion of a Concurrent Engineering strategy in relation to the traditional sequential engineering process. It has also given a number of definitions that have been used to describe Concurrent Engineering by its practitioners. This chapter concludes with an up-to-date definition of CE which has been formulated whilst undertaking this research (Hanneghan et al. 1997) and which will serve as the contextual basis for the remainder of the thesis (see Figure 1-4). It is based on the respected opinions of leading CE practitioners and personal observations of CE practice. Key points in the definition are shown emboldened (in a similar vein to that used by Cleetus 1992) to emphasise their importance and highlight the gaps in current CE research.

"Concurrent Engineering is a ***systematic approach to parallel development of all product life cycle activities****, from initial conception through design, planning, production and disposal. It is* ***an enriched communication infrastructure*** *which is* ***unconstrained by geographical location*** *that encourages right-first-time methods through* ***cross-functional team working and consensus.*** *"*

Figure 1-4. Definition of Concurrent Engineering.

1.13 Thesis structure

This thesis is structured into nine Chapters. Chapter two describes the background of the basis for current computer-based support systems looking at the role of distributed computing environments and CSCW in addition to recent advances in product information modelling and virtual working. To fully complete this survey, this Chapter also takes into account a number of recent technology advances that have impacted the work of global manufacturing organisations: initiatives such as the Internet and World-Wide Web and the prevalence of distributed object-based systems.

Chapter three presents a detailed survey of current state-of-the-art in CE support systems and describes requirements for CE support environments from the perspective of CE practitioners and leading academics. It concludes by presenting a start-of-the-art list of requirements for CE support systems noting key areas of research that need to be addressed.

Chapters four, five and six present the design of a new architecture to facilitate virtual Concurrent Engineering teams. Chapter four introduces this design through a viewpoint model that takes into consideration the various facets of CE projects and synthesises these into a single coherent model. From this viewpoint model, a high-level computer systems architecture which describes the various components needed to address the issues raised is the subject of Chapter five and an information system object model which is comprised of a detailed system model and an information systems model follows in Chapter six. These detailed models describe the inner workings of the entire system via a set of rules, constraints and policies. The design is consistently modelled throughout using an object-oriented methodology.

Chapter seven demonstrates a physical implementation of the architecture described in Chapters four, five and six and discusses the decisions made in implementing the system in the way proposed. An evaluation of the implemented system is given in Chapter eight which looks at its application in two case studies and this Chapter concludes with a comment on how the proposed system can also be applied to other problem domains in addition to Concurrent Engineering.

The thesis concludes with Chapter nine, which summarises the work done, and draws some general conclusions. A number of interesting points are also raised here for further work and further research based on the contribution of this research to the field of Concurrent Engineering.

Appendix A provides an overview of the Fusion object modelling notation used to describe the models developed in this thesis with Appendix B describing the system interfaces and Appendix C describing the information system object model. For completeness, a list of international journal and conference publications that have been made during the course of this research is given in Appendix D.

Chapter 2

"... it is safe to assume that anyone with any knowledge will have to acquire new knowledge every four or five years or become obsolete." Peter Drucker, speaking in Harvard Business Review, September-October 1992.

2. Background and influencing factors on this research

2.1 Introduction

This chapter introduces a number of factors that have influenced this research. Although the driving force for the research was the support of Concurrent Engineering, it became apparent that recent advances in information technology and working practices would play a key part in defining any new model or method for CE support. This chapter also discusses a number of other factors that now need to be considered for development of future environments to avoid this research becoming obsolete (see Drucker's comment above).

2.2 Computer Integrated Manufacturing

Computer Integrated Manufacturing (CIM) today is based on the concepts proposed by Harrington (1973) which aim to integrate data processing operations within a manufacturing environment for the entire product life cycle. A number of applications that can be used to help achieve this aim have been developed since this concept was introduced, Figure 2-1 shows a taxonomy of some of them.

This wide variety of applications, each of which usually has its own unique set of data requirements, has given rise to interoperability problems. In the manufacture of some product, for example, a CAD application may represent the product as a set of features (holes, blocks, etc.) while a stress analysis application may represent the same product's geometry as a mesh while a CAM system might represent the product in yet another format as a tool path.

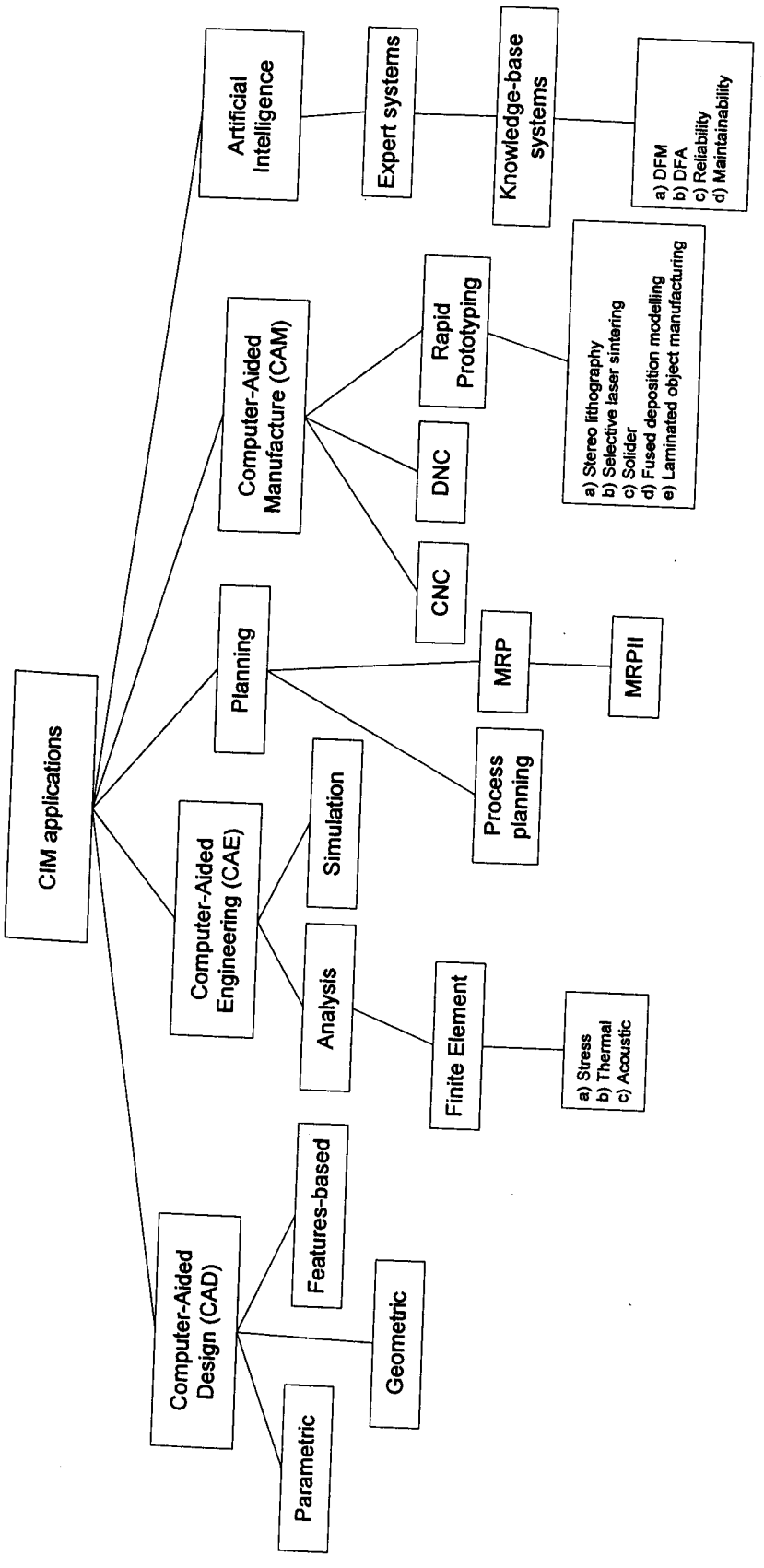


Figure 2-1. Taxonomy of CIM applications currently in use.

In addition to this, management information systems such as MRPII (Manufacturing Resource Planning) systems require product information such as lead times and due dates of component parts. It can be seen then that each CIM application can potentially have a different view of the same product data, highlighting the product's polymorphic characteristics.

A common form of implementing a common product information model between all CIM applications is via data translation. Specially written software that performs the translation between one format and another are logically configured between incompatible applications in the CIM environment (see Figure 2-2). This method can work (with varying degrees of success) but can also cause data to be lost or misinterpreted since the semantics of the data is rarely captured (Trapp et al. 1992). A number of recent efforts have begun to address the problems of multiple information models, most notably is the STEP protocol.

2.2.1 Product information models

The Standard for The Exchange of Product model data (STEP) protocol is an extensive product information model which covers the entire life cycle of a product and is an internationally recognised standard (ISO 1992b). STEP provides an object-oriented language called EXPRESS (ISO 1993) which can be used to define entities. This emphasis on object-orientation allows the capturing of the polymorphic characteristics of product data. STEP also provides a means for data translation between STEP and legacy formats via the STEP Data Access Interface (SDAI).

Other related product information model initiatives include:

- *IGES*—Initial Graphics Exchange Standard (Smith and Wellington 1986) is a format jointly developed by Boeing, General Electric and the USAF and is widely used. However, it is not suitable for solid modelling or assembly operations.
- *DXF*—DXF (Drawing interchange Format) has its origins on PC-based CAD systems e.g. AutoCAD (Autodesk Inc. 1997) and become a *de facto* standard for smaller companies.

Figure 2-2. Data translation in CIM environments.

- *CADEX, NEUTRABAS and IMPPACT*—CADEX (CAD geometry EXchange), NEUTRABAS (NEUTRal product definition dataBASe for large multi-functional systems) and IMPPACT (Integrated Modelling of Products and Processes using Advanced Computer Technologies) are projects initiated by the European Strategic Programme for Research and development in Information Technology (ESPRIT) with the goal of supporting the STEP protocol.

Many of these and other formats are currently in use by companies although most practitioners of CE believe that the STEP protocol will be dominant in the near future.

2.3 Distributed computing

Distributed computing has been defined as "a collection of autonomous computers linked by a computer network and equipped with distribution software" (Coulouris et al. 1994). This statement raises a number of important points that have importance in CE support environments:

1. *Users are separate*—Each user has their own personal workstation capable of running their own set of locally available application programs as well as those that are centrally available (for example of a company mainframe).

2. *There is autonomous operation*—Users are no longer tied to sharing a central computer where they compete for computing resources.

3. *There is a reliance on a network infrastructure*—The workstation can operate in conjunction with other computers via network communications.

Distributed computing has a number of goals: separation and transparency (ANSA 1989) and heterogeneity. Separation describes the physical characteristics of a distributed system; each user maintains a separate existence in the system. The effect of this is that workstation faults or errors are contained without disrupting the entire network. Separation also allows for the parallel execution of tasks, whereby some large task can be split into smaller sub-tasks that can be executed on a number of separate workstations simultaneously.

Transparency is defined as the concealment of separation from either the application programmer or end-system user. The ODP (Open Distributed Processing) engineering model (Farooqui et al. 1995) identifies a number of types of transparency that are important in distributed systems while ANSA proposes a few additions to this list (ANSA 1989). These transparencies are described in Table 2-1.

Heterogeneity in distributed computing systems is evident in the diverse range of architectures, programming and database languages, operating systems and application programs (Farooqui et al. 1995; Vinoski 1997). Access transparency provides a means for heterogeneous systems to inter-operate.

2.3.1 Manufacturing networks

The Manufacturing Automation Protocol (MAP) and Technical and Office Protocol (TOP) have been proposed as standards for distributed manufacturing networks (Valenzano et al. 1992). MAP was developed by General Motors to tackle integration problems experienced due to the wide heterogeneity in the computer equipment used within its networks. It has been defined by Goscinski (pg. 62, 1991) as "...a local network and associated communication protocols for programmable controllers, robots and terminals within a plant (factory)."

2.3.2 Inter-operable systems

Research into inter-operable systems emerged as a result of the problems that became evident when heterogeneous systems needed to work together. These problems arose from the incompatibilities of competing systems in areas such as data representation, network transport protocol and programming language support. This inter-operability also now needed to span organisational boundaries, not just plant wide LANs which in the view of Nickerson (1990) failed to solve the underlying problems.

2.3.2.1 Homogeneous language-based systems

Recently, homogeneous language-based inter-operable systems such as the Remote Method Invocation (RMI) mechanism built into the Java language (Arnold and Gosling 1996) have become popular. This approach effectively turns the problem of heterogeneity on its head by enforcing the use of a single language (Java in this case) which runs inside a portable Virtual Machine core. The Java Virtual Machine (JVM) is made available on multiple computer platforms within a distributed environment and application programs are only ever written in Java. In this type of environment the applications are homogeneous while the JVM is heterogeneous.

Table 2-1. Transparencies important in distributed computing.

Transparency	*Effect for distributed computing*
Access	Clients are unaware of the invocation mechanisms at the server interface (i.e. whether they are local or remote).
Concurrency	Hides from the clients the fact that there may be several other clients concurrently accessing the server.
Failure	Clients are unaware of the failure of the server and its subsequent reactivation.
Federation	Clients need not be aware that the system spans many organisational boundaries.
Location	Clients are unaware of the physical location of the server.
Migration	Clients are unaware that the server may have been dynamically relocated.
Performance	Allows the system to be optimised to improve performance as system load varies.
Replication	Clients need not be aware that there may be multiple replicated instances of data.
Resource	Clients are unaware of the deactivation and reactivation of the server.
Scaling	Allows the system to expand and decrease in scale without change to system structure or application programs.

2.3.2.2 Heterogeneous language-based systems

Heterogeneous language-based systems attempt to allow more than one language (and / or computer architecture) to inter-operate. This is usually done through the use of some intermediate language which participating systems can use to describe their interfaces to other services and clients. This language translates any proprietary data formats into one common format that is used by all participants. A number of standards for this type of inter-

operable system have been proposed including the Object Management Architecture, Distributed Computing Environment (DCE) and the Remote Procedure Call (RPC).

The Object Management Architecture

The OMG (Object Management Group) is a consortium of more than 700 organisations working together to create standards to address the challenge of developing network applications. At the core of OMG's work is the CORBA (Common Object Request Broker Architecture) standard that defines a communication infrastructure allowing a heterogeneous distributed collection of objects to collaborate transparently. CORBA is part of the Object Management Architecture Reference Model (shown in Figure 2-3) which consists of the following components:

- *Object Request Broker*—This enables objects to transparently make and receive requests and responses in a distributed environment. It is the foundation for building applications from distributed objects and for interoperability between applications in heterogeneous and homogeneous environments.

- *Object Services*—These are a collection of services (interfaces and objects) that support basic functions for using and implementing objects and which are always independent of application domains. CORBA defines fourteen such object services (Object Management Group 1997) which cover aspects such as Naming, Events, Life Cycle, Persistence, Transactions, Concurrency Control, Relationships, Externalisation, Queries, Licensing, Properties, Time, Security and Object Trading.

- *Common Facilities*—these are a collection of services that many applications may share, but which are not as fundamental as the Object Services. For instance, system management and electronic mail are classified as common facilities (Object Management Group 1995b).

- *Application Objects*—these are products of a single vendor or in-house development group which controls their interfaces. Application Objects correspond to the traditional notion of applications, so they are not standardised by OMG. Instead, Application Objects constitute the uppermost layer of the Reference Model.

CORBA makes use of an intermediate language called IDL (Interface Definition Language) which is used to describe object interfaces in a single common format. Its structure is very similar to the C++ programming language (Stroustrup 1991) and the OMG has produced mappings from IDL to many popular languages such as C, C++, Java and Smalltalk. A number of commercial and research-based computer systems have been successfully built using the Object Management Architecture.

Other initiatives

The Remote Procedure Call (RPC) (Birrell and Nelson 1984; Coulouris et al. 1994) is a technique which allows software application programs to call procedures or functions which reside in software on distributed machines as simply as calling procedures local to the application. It does this by packaging (also known as marshalling) any parameters associated with the procedure call and transmitting these over the network where they are un-marshalled at the receiving end. The remote procedure then executes using these parameters and any results are passed back to the calling application in a similar fashion. The information passed in an RPC is translated into a portable common format known as XDR (eXternal Data Representation). The most common implementation of RPC is found in the BSD UNIX operating system.

DCE (Open Software Foundation 1993) is a suite of integrated software services that uses RPC. It also includes a number of key services:

- *Security service*—This authenticates the identities of users, authorises access to resources on a distributed network and provides user and server account management.
- *Directory service*—This provides a single naming model throughout the distributed environment.
- *Time service*—This synchronises the system clocks throughout the network.
- *Threads service*—This provides applications with multiple threads of execution capability.
- *Distributed file service*—This provides access to files across a network.

2.4 Database models

The product development process typically generates huge amounts of information, which brings with it the requirement of secure, reliable, long-term storage. Database systems are commonly used to store and manage this type of information and a number of database models currently exist. Early database models included the hierarchical and network models. However, more advanced and generally easier to use models such as the relational and object-oriented models have largely superseded these. These last two models are therefore described in greater detail in the following sub-sections.

2.4.1 Relational databases

The relational database model has its origins in set theory and was devised by E.F. Codd in the 1970s (Codd 1970). Entities to be stored in a relational database are usually normalised to remove data duplication and redundancy before being stored in tables. These tables have columns or attributes and rows or tuples. The data in these tables can be manipulated using a relational algebra of which SQL (Structured Query Language) is the best known example. SQL has been the subject of many standardisation efforts that have resulted in the now universally accepted SQL92 standard. Examples of relational databases include Ingres, DB2 and Oracle 7.0.

2.4.2 Object databases

Object databases (also known as object-oriented databases and Object Database Management Systems or ODBMS) are defined by the Object Database Management Group (ODMG) as: "...a DBMS that integrates database capabilities with object-oriented programming language capabilities" (Cattell et al. 1997); in other words, the objects in the database appear transparently as programming language objects.

There are a number of reasons why object databases are more amenable to Concurrent Engineering projects than relational databases. These include:

- *Better support for abstract data types (ADTs)*—Object databases can support diverse media such as video, audio and proprietary binary information such as CAD models as well as providing a means to support abstract data types which allows new data types to be defined in the future. Concurrent Engineering data covers the entire product life cycle and therefore includes many arbitrary data types.

- *Better support for highly complex interconnected objects*—Object-oriented languages such as Smalltalk (Goldberg 1984) and Java (Arnold and Gosling 1996) provide data structures that allow collections of objects to be closely interconnected and aggregated which can result in complex tree-like structures. Object databases can store these complex objects '*as is*', whereas relational databases must first break the structure into normalised entities before it can be stored.

- *More fine-grained operations*—Locking can take place at the individual object level as opposed to the relational table or page[5] level resulting in more responsive transactions for users.

- *Better support for distributed environments*—Object databases map more naturally into distributed environments (Wade 1995).

Early research into object databases (Kim and Lochovsky 1989) has resulted in a number of commercial products including Itasca which grew from a research project called ORION (Kim et al. 1989), GemStone (Bretl et al. 1989), ObjectStore, O_2, Versant and Poet.

2.4.3 Object-relational databases

Object-relational databases have the relational model as their foundation. They typically extend this, however, to include support for object-oriented concepts such as subtypes and inheritance, polymorphism, object identifiers and the ability to store abstract data types and their behaviour (Stonebraker and Kemnitz 1991). Standardisation efforts[6] to unify the operation of object-relational databases has been slow, but recent efforts aim to ratify the SQL3 query language which is a superset of SQL92 with standard object extensions. Examples of object-relational databases include Postgres (Stonebraker et al. 1990), Illustra and Oracle 8.0.

---

[5] A page is a subset of a some data (i.e. a table) and is usually of a fixed size, e.g. 64 kilobytes.

[6] Both ANSI (X3H2) and ISO (ISO/IEC JTC1/SC21/WG3) have standardisation committees for recommending object extensions to SQL.

2.5 Computer-Supported Co-operative Working

Computer-Support Co-operative Working (CSCW) is the term used to describe collaboration between multiple team members using computers. CSCW emerged in the mid-1980s and is now a recognised interdisciplinary research field in its own right (Spurr et al. 1994). 'Groupware' is the term being given to computing products which are beginning to provide practical examples of the tools and techniques involved in CSCW (Golfin and Jackson 1994) and whose purpose is to 'support the collaborative activities of organisational workgroups' (Mandviwalla and Olfman 1994). Certain groupware applications are now becoming common in organisations such as e-mail and calendar applications (these are described in more detail in Table 2-2.) A more detailed review of CSCW can be found in Greenberg (1991a).

Typical examples of groupware applications and where they are used are shown in Table 2-3. This figure is composed of both a time and space dimension: a group of people can work at the same time or collaborate over some period of time. Furthermore these people can either be in the same location (e.g. the same building) or physically distributed (e.g. spread throughout an entire country). The time-scale of the product development process also dictates the level of collaboration: shorter time-scales may require almost instantaneous (synchronous) communication between team members while longer time-scales may allow delayed (asynchronous) feedback. Table 2-4 gives examples of typical product development processes and the time-scales involved.

The importance of CSCW should not be understated; results of recent studies have shown that electronic brainstorming is superior to traditional verbal brainstorming for large groups involved in sharing ideas (Aiken et al. 1994). This is highly applicable to Concurrent Engineering where multiple team members aim to achieve consensus. In addition to this, dramatic cost savings can be made by utilising electronic conferencing tools; an article in Computing (Massey 1996) describes how one company saved its executives 750,000 miles of business travel a year through the use of videoconferencing. However for groupware to proactively support a CE process, the specific needs of a decentralised virtual team need to be considered. Mandviwalla and Olfman (1994) report that current groupware tools are mostly 'leader' driven. As will be described in the next section, virtual teams rarely involve a central leader, instead relying on group consensus for control.

Table 2-2. Types of CSCW application.

Application	*Description*
Electronic mail (e-mail)	E-mail is used to send electronic messages to individuals or groups of individuals. Recent enhancements such as support for MIME (Multipurpose Internet Mail Extensions) types means that e-mail can be used to send binary data not just text.
Bulletin board	Bulletin boards are used as electronic notice boards where users can read and post electronic notices. An example of this can be found in the Lotus Notes product (Lotus Corporation 1993).
Group scheduler / calendar	Group schedulers allow managers to schedule meetings electronically. Each user has an individual calendar that they maintain with their own appointments. The group scheduler can interrogate a user's calendar to determine if a meeting is possible at a given time. In the event of a conflict, the scheduler can find the next available meeting date when all respective participants are free. An example is the Clockwise application (Hayes 1992).
Workflow system	Workflow tools aid in the monitoring and managing of state and stage-oriented operations. An example is DEC LinkWorks (Smith 1994).
Project management	Project management tools can be used to plan and monitor the progress of a project. The tool can be used to quickly calculate the effect of any changes to a schedule.
Collaborative authoring	Collaborative authoring occurs when more than one author prepares a document either in real time or asynchronously. These tools facilitate collaborative authoring by allowing multiple users to share a common resource while preserving any changes that are made e.g. by allowing non-destructive editing. An example can be found in CoDraft (Kirsche et al. 1993).
Screen sharing	Screen sharing is a technique used when multiple users need to view changes to a document in real-time. A number of users effectively 'share' a common screen in which the effects of changes by any user can be seen immediately. An example is Share (Greenberg 1991b).
Meeting support system	Meeting support systems are used to facilitate recording and participation in meetings. An example can be seen in the Collaborative Management Room (Nunamaker et al. 1991).

Table 2-2 (continued)

Application	*Description*
Desktop conferencing	This is a technique for allowing multiple users to participate in a 'conference' electronically. Typically each user can view and send text-based comments to all other users involved in a given conference. An example is the MONET application (Srinivas et al. 1992).
Videoconferencing	A logical extension to desktop conferencing is videoconferencing that includes the ability to use real-time video and audio. Typically this is used to relay live visual displays of the participants. An example of this is ISDN PC (Jager et al. 1993).

Table 2-3. Typical uses of groupware applications.

		Mode of working	
		Simultaneous	Collaborative
Location	Local	• Face-to-face meetings • Group decision support systems • Meeting support systems	• Local area network • Electronic mail • Common information model repository • Collaborative authoring tools • Project management • Workflow management • Bulletin boards
	Distributed	• Wide area network • Common information model repository • Desktop conferencing • Videoconferencing • Collaborative authoring tools • Screen sharing • Real-time document management	• Wide area network • Electronic mail • Common information model repository • Project management • Workflow management • Collaborative authoring tools • Bulletin boards

Table 2-4. How the product life cycle dictates types of communication.

Product	*Life cycle time-scale*	*Example*	*Communication*
Daily newspaper	very short	Less than 24 hours	synchronous
Fashion clothing	short	Weeks	synchronous / asynchronous
Computer software	medium to long	Months to years	asynchronous
Aerospace	long to very long	Many person-years	asynchronous

2.6 Virtual teams

Trapp et al. (1992) defines the virtual team as "...a geographically scattered team of experts who are connected to a computer network and communicate through multiple media and share information using groupware and other shared information models." Virtual CE teams are composed of cross-functional members from the entire life cycle of the product. In order for these team members to be able to effectively co-operate and share information, they must each conform to a common product information model. An example structure of a CE virtual team is shown in Figure 2-4. In addition to the team members shown in this diagram, customers and suppliers also need to be considered as part of the virtual team.

2.6.1 Problems associated with virtual team working

A number of problems associated with team working over large distances have been highlighted by Scrivener et al. (1995). These problems include cross-cultural differences and cross-time zone differences in addition to the problems related to technology and work practices highlighted earlier.

Cultural differences exist when teams are composed of members from more than one country. An extreme example of this is the collaboration of an English speaking organisation and a Japanese speaking organisation which requires translators to act as intermediaries between the two parties. Product and project information may be stored in one common language that may lose valuable details in translation, or in a combination of both languages which would require translations. It has been shown that even when team members come from countries which share a common language, e.g. England and Australia,

cultural differences can still exist. For a detailed investigation of these problems the reader is referred to Scrivener (1995).

Time zone differences are a consequence of the wide geographic dispersions that are possible in virtual teams. Problems include finding a suitable time of day to arrange meetings or group collaborations. When team members are situated on opposite sides of the globe, normal working hours may not coincide. For example, the time in Japan is nine hours ahead of London so while one office is working, it is highly likely that the other is closed and its workers sleeping.

2.6.2 Practical issues for virtual CE teams

Prasad (p. 211, 1996) lists six common pitfalls in Concurrent Engineering projects three of which relate to the virtual team working paradigm. These are the risk of wasted efforts, the build-up of errors and 'concurrent chaos'. Wasted effort occurs when team members perform the same task in parallel unaware of each other's actions. Error build-up occurs when team members engage in more simultaneity that necessary, taking valuable time away from key activities such as error correction. Concurrent chaos becomes evident when incomplete information is released early to other team members in order to improve the concurrency of the overall project that instead intensifies the error build-up problem.

It follows then that successful virtual team working requires a systematic approach in order to be most effective. The first, and possibly the most important step, is the formation of the CE team. This is the collective unit that will work together to achieve a common goal. In the

context of this research it is considered that teams that may be made up of non-collocated team members—so called "Virtual Teams". In order for virtual CE teams to be effective there are a number of issues that need to be addressed, accepted and adhered to by all members from senior executive to shop floor worker. These are described in detail below.

2.6.2.1 Team structure

The virtual CE team (henceforth referred to as 'the Team') is a cross-functional team. In order to capture aspects from all stages of the product life cycle it must be representative of all the requisite stages. This requires building the Team from actual stakeholders in the product, examples include personnel from disciplines such as: executive management, product design, marketing, sales, production, maintenance, purchasing, disposal, materials, finance, and support. One important team member who must also be considered is the eventual customer. In this team building exercise, it is sometimes helpful if one particular team member acts as the 'project champion' i.e. someone who makes it their goal to ensure the success of the project and help the project gain momentum.

2.6.2.2 Consensus

An important factor in improving product quality is to reduce the level of scrap or reworking needed. This 'waste' arises when communication between Team members breaks down or is simply ineffective. The Team should aim to adopt a consensus approach to decision-making whereby all interested parties can state their approval or disapproval on a particular aspect. This is very different from traditional sequential engineering processes where the product passes through various production stages with little or no comment from down-stream employees. One approach to achieving consensus is through regular product development reviews that involve all team members and allow each member to state their views. This could be in the form of regular conferences or disciplined group voting available as CSCW software tools.

2.6.2.3 Level playing field

Computer-based communication breaks down hierarchical barriers and can provide an environment where all voices are equal. However most computer-based communication methods are radically different from traditional methods such as face-to-face communication and telephone calls and as such require new skills to be learned to achieve their full potential. Cutkosky et al. (1996) cites one such example during the Madefast project of a technically competent technician who was reluctant to speak up during group conferences or respond to broadcast solicitations for feedback. In this case it was deemed more effective to consult the technician directly, thereby abandoning electronic methods. A virtual team support environment should record contact information of all team members

such as their address, fax and telephone numbers to facilitate this form of communication also.

2.6.2.4 Anticipating problems caused by dramatic change

Henry Ford was a pioneer of new manufacturing processes in the early twentieth century. His assembly line approach greatly increased productivity (and company profits) but just six years after production of his revolutionary Model T car began, labour turnover was at forty to sixty percent. This was largely due to the monotony and repetition of assembly line procedures. Workers were not accustomed to this method of working; the change was just too dramatic.[7] This stresses the need for improved working practices perhaps in the form of new pay structures (Ford increased stability in the workforce by doubling the industry standard daily wage) or bonus schemes such as performance-related pay.

Smith (1997) argues that CE is the result of a summary of best practice in product development rather than the adoption of a radically new set of ideas. However, this research contends that when an organisation versed in the more traditional sequential product development process is faced with implementing CE, the transition is significant. For the degree of change to be considered insignificant, Smith's theory of best practice coalescence assumes that an organisation has already undergone a period where it was committed to some other product and process improvement philosophy.

2.7 The Internet

The Internet began life in 1969 when the US Department of Defense commissioned ARPANET for computer networking research (Baran 1995). ARPANET produced a means of linking military networks together. This 'network of networks' initially covered only military networks in the United States but was quickly expanded to include defence-related corporations and research institutions. In the 1980s these interconnected networks spread widely to reach universities and other organisations throughout the world. Now, it is possible for the Internet to reach almost every household that has the necessary computer hardware and software and a telephone line. The actual number of computers connected to the Internet is roughly doubling each year (see Figure 2-5).

[7] This concept has proved too expensive in present day motor vehicle manufacturing with manufacturers instead making the transition to robot assembly to handle mundane tasks.

Both large and small companies can access this global inter-network relatively easily and cheaply and there is no discrimination against who can join the Internet, as it is not owned by any single organisation. This may be of special importance to companies in third world and developing countries that wish to use the Internet to assist a Concurrent Engineering strategy. Ellsworth and Ellsworth (1994) provide ten reasons why businesses are using the Internet and it is clear that these reasons have a lot in common with the goals of a CE strategy (Table 2-5 describes these reasons in relation to a CE strategy.)

2.7.1 Internet terminology

The term 'Internet' has been associated with many meanings in published literature. This section provides some definitions of the Internet terminology that will be used in the remainder of this thesis.

Internet—The Internet is a global infrastructure that links networks of networks, i.e. it is an inter-network.

Intranet—An Intranet is a local and usually private network that is based on Internet technologies (see §2.7.2). Intranets are typically used to provide a range of information services for employees and departments within an organisation.

Extranet—The term Extranet refers to any Intranet site that can be accessed securely by an

external Internet site. This is used for example to enable organisations to share information with business partners.

2.7.2 Internet technologies

A number of useful Internet technologies have been developed and standardised such as FTP (File Transfer Protocol) which allows users to send and receive files from any connected site (Postel and Reynolds 1985), NNTP (Network News Transfer Protocol) a standard for the stream-based transmission of news (Kantor and Lapsley 1986) and Gopher which is a distributed document search and retrieval protocol (Anklesaria et al. 1993). These technologies are now widely used by the Internet community. However, possibly the most well known Internet technology of recent years is the World-Wide Web.

2.7.2.1 The World-Wide Web

The World-Wide Web (WWW) began life in March 1989 when Tim Berners-Lee of the European Particle Physics Laboratory in Geneva (CERN) proposed the project as a means of transporting research and ideas effectively throughout the organisation. It initially outlined the use of a simple system of networked computers that would use a hypertext protocol to communicate and transmit documents (Berners Lee et al. 1994). The technologies that underpin the WWW include a standard for describing hypertext documents (HTML) HyperText Markup Language (Raggett 1997); a special transport protocol for hypertext navigation (HTTP) HyperText Transport Protocol (Fielding et al. 1997); and a means for server-side applications to interact with client request (CGI) Common Gateway Interface (McCool 1994).

The WWW is a distributed architecture on which to base simple client-server applications and a number of novel WWW-based applications are currently being used and developed to exploit this. Examples include on-line insurance quotations, credit ratings, stocks and share prices, home shopping malls and even the purchasing of a new car.

The WWW can be considered as a medium for delivering requested information. In a CE environment, multiple members of a virtual team both produce and consume information: designers and engineers create new designs or proposals and submit them into a repository where they can be viewed, commented upon and reviewed or modified by other team members. These team members also need to communicate with each other to share and discuss ideas and problems (Siemieniuch and Sinclair 1994). For these reasons, a number of researchers have explored the use of the WWW as a medium for supporting CE (these efforts are described in detail in the following chapter.)

Table 2-5. Why organisations are using the Internet
(Source: Ellsworth and Ellsworth 1994).

Reason for using the Internet	*Concurrent Engineering analogy*
Communication (internal and external)	Effective communication is needed both internally (i.e. between team members) and externally (i.e. between the organisation and its external suppliers).
Corporate logistics	A CE strategy requires systematic planning and organisation.
Levelling the playing field—globalisation	This is desirable for global organisations employing CE and smaller organisations wishing to compete with global competitors.
Gaining and maintaining competitive advantage	This is a primary goal of CE.
Cost containment	This is a primary goal of CE.
Collaboration and development	CE is intrinsically a collaborative development process.
Information retrieval and utilisation	Sharing information is a key aspect of a CE process.
Marketing and sales	These tasks are inherent in the product life cycle.
Transmission of data	Virtual team working always requires some method of data transmission.
Creating a corporate presence	Publicising the organisation to external suppliers, customers and markets.

2.7.3 Recent developments

This section discussed some of the key developments that have taken place in the field of Internet-based applications. Internet-aware applications are now being developed using the technologies described below with the aim of linking enterprise-wide systems.

2.7.3.1 Internet programming languages

A number of programming languages have been developed recently that take advantage of Internet connectivity allowing applications to be developed that can span distributed computer networks. Software applications written in traditional programming languages such as C (Kernighan and Ritchie 1988), C++ (Stroustrup 1991) and COBOL are typically compiled to run on a specific hardware architecture. A key feature of Internet programming

languages is that they are interpreted on the client machine to ensure high portability of the language. These include:

- *ECMA-262*—This is a standard devised by the European Computer Manufacturers Association (ECMA). ECMA is a European-based association for standardising information and communications systems. The standard recently approved is based on joint submissions from Microsoft and Netscape and is similar to the JavaScript language with a few notable enhancements.

- *Java*—The Java language (Arnold and Gosling 1996) is described as an "...object-oriented...robust and secure...architecture neutral and portable...high performance...interpreted, threaded and dynamic" environment (Gosling and McGilton 1995). Java programs are compiled into a portable intermediate bytecode that is interpreted by a Virtual Machine on the client.

- *JavaScript*—JavaScript (Netscape Communications Corporation 1996) is a scripting language based on the Java language that allows the scripting of events, objects and user actions within a World-Wide Web browser.

- *Limbo*—Limbo is a programming language developed to support the Inferno network operating system (Lucent Technologies 1996). Limbo programs are compiled into a portable intermediate bytecode that is interpreted by a Virtual Machine on the client.

- *Tcl/Tk*—Tcl/Tk is an Internet scripting language from SunLabs (Ousterhout 1994). This interpreted language is composed of two components: Tcl (tool command language) which is the core language and Tk (toolkit) which is a graphical user interface toolkit.

- *VBScript*—Visual Basic Scripting Edition (also known as VBScript) is a subset of the Microsoft Visual Basic language (Microsoft Corporation 1997). It is implemented as a fast, portable, lightweight interpreter for use in World-Wide Web browsers and other applications.

- *Weblets*—Weblets are self-contained applications which can be either downloaded when needed or run locally from a client disk (Eòlas Technologies 1995). The programming language used for these Weblets, Spynergy, is based on Tcl/Tk and is a high-level, interpreted language.

For a further discussion of scripting languages, the reader is referred Ousterhout (1997).

2.7.3.2 *Internet Inter-ORB Protocol (IIOP)*

With the growing popularity amongst businesses to use CORBA-based Object Request Brokers in addition to Internet-based applications, the OMG has devised a means for ORBs to communicate via the Internet. This communication protocol is known as IIOP (Internet Inter-ORB Protocol) and uses the standard TCP/IP transmission protocols. IIOP helps leverage applications which span many organisations by using the Internet as a common medium.

2.8 Summary

This chapter has introduced a number of key technologies that have influenced this research. These factors include:

- existing CIM systems and the problems of integration between CIM applications
- the provision of information sharing and long-term storage
- the role that distributed computing has had on defining the way organisations inter-operate, notably in the areas of CSCW tools and techniques, distributed object-based computing environments and the impact of the Internet, and,
- how organisational structures suitable for CE can be captured and managed by applying a virtual team concept.

Aspects from the technologies described here have been shown to be beneficial to the building and operation of technology-based CE support environments. However, a number of issues and requirements that need to be addressed are raised if these technologies are to aid the Concurrent Engineering process. This is the subject of the next chapter, which also looks at a number of existing CE support systems.

Chapter 3

3. Concurrent Engineering support environments

3.1 Introduction

This chapter provides a review of current computer systems that have been used to support a Concurrent Engineering strategy. The chapter begins with a survey of requirements that have been proposed and requested by researchers and practitioners in the field of Concurrent Engineering. It then provides a survey of existing CE support tools and systems from a broad spectrum of disciplines. The chapter concludes by evaluating these existing support environments in the light of user requirements and presents a state-of-the-art requirements definition for Concurrent Engineering support environments.

3.2 State-of-the-art survey of requirements for CE support

The requirements for CE support systems presented here fall into three categories: barriers to building new systems, architectural requirements and software requirements. In considering possible barriers and exploring the means to overcome them, requirements for building future support environments can be obtained.

3.2.1 Barriers

The barriers considered here as those aspects which impede the development or operation of Concurrent Engineering. Barriers exist because of reliance on legacy applications and data, non-conformance to standards and lack of adequate supporting infrastructure. It has been proposed by Schmitz and Desa (1993) that an ideal CE development method cannot be feasibly implemented without a computer environment. There are many researchers and academics who support this view, but it must also be noted that any single approach to CE cannot be considered a panacea and ideally should be used in conjunction with other complementary approaches, such as organisational and cultural changes. A comparison of approaches to CE can be found in Dowlatshahi (1994). It is however, unarguable that a well designed and robust computer environment can help support and manage the information processing requirements associated with a Concurrent Engineering project (Adams et al. 1995).

Londono et al. (1992) describe a number of barriers to Concurrent Engineering. These include organisational issues such as:

a) The need for an organisation to "...subscribe to a open environment where co-operation is assured."
b) Training managers to embrace computer technology that is being used to support the CE process.
c) Tools for assessment and tracking of progress.

Additionally, technological barriers such as the ability to "...scale up easily from small projects with a few people to large, complex projects with a great number of people" and providing tools to help achieve "common visibility" among team members are described.

Trapp (1992) identifies a number of technological barriers that hinder CE development. These are:

a) A lack of viable software architectures for CE.
b) A lack of effective and transparent mechanisms to share product data and support tools that reside on heterogeneous hardware platforms in multiple geographic locations.
c) A lack of compatible formats and representations that enable inter-operability among multiple CAD and CAE programs.
d) A lack of standards and mechanisms for sharing product, process and organisation data.
e) A lack of communication and co-ordination tools which support interaction, negotiation and conflict resolution among geographically dispersed teams.
f) A lack of mechanisms to capture design intent and product development history.

Sobolewski and Erkes (1995) also describe a number of barriers from the perspective of agile manufacturing including lack of network-based services, lack of computer communications facilities, the physical separation of user and data and restrictive security procedures. They proceed to supply a number of requirements for support systems, which specify that the system must provide:

a) Sharable design and manufacturing services.
b) A locator for the service taxonomy.
c) Understandable and organised information presentation facilities.
d) Access to distributed information from distributed computer workstations.
e) Management of security, persistence and maintenance of the information and underlying services.

3.2.2 Architectural requirements

Architectural requirements help define the infrastructure upon which support environments can be built. Nickerson (1990) suggests three 'foundations' to successful Concurrent Engineering as:

a) Proper organisational structure.
b) Effective product and team communication.
c) Efficient use of technology.

In Nickerson's model, these foundations are analogous to the foundations in house building: technology supports communication, communication supports organisation and organisation supports Concurrent Engineering. In contrast to Schmitz and Desa (1993), Nickerson states that CE can still be practised even without the technology component.

Molina et al. (1995a) has compiled a list of requirements for Concurrent Engineering support architectures. These state that an architecture should:

a) Identify, co-ordinate and communicate between the different perspectives involved in [Concurrent Engineering], whether represented by groups of people or by applications.
b) Provide information and knowledge sources that are able to represent evolving expertise and product designs, that can be readily modified and that are easily accessible.
c) Monitor the history of the design process so as to enable future design procedures to capture best practice and to maintain accountability.
d) Control and configure the various system elements in a way that is transparent to the user and ensures system integration.
e) Provide an interactive, multimedia interface for the system user.

The National Industrial Information Infrastructure Protocols (NIIIP) Consortium is a team

of organisations that has entered into a co-operative development agreement with the U.S. Government to develop open industry software protocols in an effort to make it possible for manufacturers and their suppliers to effectively inter-operate as if they were part of the same enterprise (Goldschmidt 1996). The vision of the NIIIP Consortium is: "... to make U.S. industrial enterprises more globally competitive through a new form of collaborative computing that supports the formation of 'virtual enterprises'," (NIIIP Consortium 1995). The Consortium considers virtual enterprises to be teams who are linked using technology to achieve a common purpose, share skills and costs with each participant contributing their core competence; a description which maps well to the definition of CE. It identifies four technology requirements for Industrial Virtual Enterprises (see Figure 3-1):

a) Common communication protocols.
b) A uniform object technology base for system and application interoperability.
c) Common information model specification and exchange.
d) Co-operative management of integrated virtual enterprise processes.

With the backing of a number of major US manufacturers, the NIIIP architecture clearly holds some significance for the building of future CE support architectures.

3.2.3 Software requirements

The building of CE support environments is a complex software engineering task (Hanneghan et al. 1998). Software-based support tools for CE have their own unique set of requirements. Sohlenius (1992) specifies three fundamental functional requirements for Concurrent Engineering software tools. These are:

a) *Perspectives*—This refers to each of the different life cycle aspects such as function, structure, manufacturing and maintenance.
b) *Stages*—This refers to the current stage of decision-making activities, e.g. from concept through to detailed analysis and production.
c) *Participants*—This considers the number of people required to develop a given product.

These requirements are shown in the form of a cube in Figure 3-2. Each of the three requirements makes up an axis of the cube. If the cube is considered to have a volume of one, the ideal software tool to support CE would be found at the point (1,1,1). Sohlenius argues that current tools exist primarily at the point (0,0,0) in this diagram, i.e. supporting only single users, for single perspectives at single stages in the life cycle.

Smith (1988), cited in Sohlenius (1992), states that for Concurrent Engineering, group productivity tools must meet the following requirements:

a) Integration of complementary engineering expertise.
b) Co-operation of multiple competing perspectives.
c) Communication of upstream and downstream concerns.
d) Co-ordination of group problem solving activities.

Cutkosky et al. (1993) reiterates these points by describing a need for tools: "...that will help the team members share knowledge and keep track of each others' needs, constraints, decisions and assumptions."

3.3 CE support systems and tools

This section provides a survey of fourteen existing support systems and tools that are being used to support Concurrent Engineering. These systems are presented in alphabetical order.

3.3.1 BSCW shared workspace system

The BSCW (Basic Support for Co-operative Work) shared workspace system (Bentley et al. 1996; Bentley et al. 1995) is a WWW-based environment for sharing documents between

distributed collaborative team members. The documents are contained in 'workspaces' to which can be assigned basic version and access control parameters. These workspaces are small repositories in which users can upload and download information, hold threaded discussions and obtain information on the activities of other colleagues using that workspace.

Access control allows restricted access to workspaces by known members who must authenticate themselves using a username and password combination. BSCW also monitors 'events' that occur in workspaces such as the adding of a new document, the deletion or modification of existing documents or even the reading of a document. These details of these events can be delivered to the user when a user enters a workspace or when requested[8]. The BSCW software[9] is written in the Python scripting language and uses a standard HTTP server application running on UNIX operating systems.

3.3.2 CE-Toolkit

The CE-Toolkit (Erkes et al. 1996; Lewis et al. 1994) was developed as part of the DARPA Initiative in Concurrent Engineering (DICE) programme. It is an Internet-based environment which makes extensive use of bespoke software 'wrappers' to allow legacy applications to be integrated via the World-Wide Web. These wrappers are created using the Tcl/Tk language (Ousterhout 1994) and provide a user interface for the legacy applications so that they can be accessed remotely via the WWW. The tools include simple publishing applications such as handbooks and catalogues and various analysis applications.

3.3.3 CM and PCB

The Communications Manager (CM) (Kannan et al. 1992) and Project Co-ordination Board (PCB) (Londono et al. 1992) came about from work undertaken as part of the DICE programme by the Concurrent Engineering Research Centre at West Virginia University. The PCB is described as: "...being developed with the aim of supporting co-ordination of product development by the virtual team" while the CM is described as: "...a framework for distributing messages, objects and CPU cycles in a heterogeneous networked environment." In combination, these two technologies can be used to help support virtual CE teams.

The PCB provides facilities for common visibility and reaching of consensus via the detection and notification of conflicts, workflow management and tracking of design progress. It does this by means of a 'common workspace' which models product structure (a

[8] This information is provided asynchronously, i.e. after the event has happened.

[9] The software is currently up to version 3.0 at time of writing (for more details the reader is referred to the BSCW World-Wide Web site at http://bscw.gmd.de/).

cross-functional view of product data), organisation structure (a representation of the virtual team structure along with roles and duties of team members) and activities structure (details of product development life cycle activities). The PCB uses a window-based GUI that has been built using the X Window system (Scheifler and Gettys 1992).

3.3.4 CoConut

The CoConut (Computer support for Concurrent design using STEP) environment has been developed by the Fraunhofer-Institut für Graphische Datenverarbeitung using an object-oriented methodology. The goal of the CoConut environment is: "...the integration of existing technologies in CAD modelling, design, analysis, simulation, networked co-operation and virtual reality," (Kress 1996). The architecture, shown in Figure 3-3, is composed of three main components: a communication system, a data management system (which uses a distributed object-oriented database) and kernel applications. A previous incarnation of the CoConut environment (Jasnoch et al. 1994) also included a separate user interface system that offered a generic look and feel to applications within the environment.

CoConut supports the sharing of product information using the STEP protocol but also provides a means for translating IGES files into STEP format as well as allowing the storage of arbitrary native application file formats. The environment also incorporates CSCW applications to enable synchronous and asynchronous communication. A standard front-end application called the 'Cockpit' is used by team members to access the functionality of the environment.

3.3.5 *COMBINE*

The COMBINE architecture (Merabti and Carew 1994) is described by its authors as a "framework for co-operation". It is a distributed computer systems architecture that makes use of a central Collaboration Management Object (CMO). Components such as Application Entities and Service Providers populate a Concurrent Engineering environment and perform operations via the CMO. They do this by way of a message passing mechanism that is handled by a Communications Object local to each entity (see Figure 3-4). The CMO acts as a collaborative working facilitator and scheduler for entities. It also handles the access control function of the environment and communication routing via each entity's Communications Object.

3.3.6 *DMMS*

The Design Management and Manufacturing System (DMMS) (Bounab et al. 1993) is described as a "...concurrent engineering manufacturing architecture based on PCTE" (Portable Common Tool Environment). At the core of DMMS is a common data repository which is based on PCTE but which has been extended to a NIAM (Nijssen Information Analysis Method) representation. DMMS claims to integrate design, manufacturing (simulation and maintenance) and management (of data and the manufacturing process) by means of this repository. It does this by wrapping common domains of tools using a 'reference model' particular to those tools. This reference model allows data sharing to take place between these tools. In addition to this, a central 'DMMS reference model' is employed to allow data to be shared across tool domains. However, the authors note that the

"[repository model] covers partially the needs of CIM modelling" due to the repository only modelling the data that is shared between tools.

3.3.7 I-CARE

The I-CARE (Interactive-Computer Aided Reliability) system (Santos and Cardoso 1993) is based on a concurrent design methodology consisting of engineering analysis, reliability, maintainability and producibility design. The environment integrates commercial software packages such as PATRAN and ANSYS with bespoke design sensitivity analysis software (see Figure 3-5). For data storage, the environment uses a relational-database although access to the database is via an object-oriented wrapper. For ease of use, a Graphical User Interface based on the Motif windowing system is used to provide a menu-driven front-end application. This allows access to all the tools within the environment. Some of the tools supported include: structural modelling, finite element pre- and post-processing, visualisation, Finite Element Analysis (FEA), sensitivity analysis, reliability analysis, deterministic and reliability-based optimisation.

3.3.8 iDCSS

The iDCSS (Integrated Design Collaboration Support System) is a support system for co-ordination during a CE process (Klein 1996). The iDCSS architecture contains three core services: dependency capture, process enactment and exception management. The dependency capture service "records the dependencies among the process and product decisions". The process enactment service is used to deliver documents among participating agents. The exception management service attempts to detect and avoid conflicts that occur during process definition and enactment. In addition to these three services there is a central repository for storing product, process and organisational information. Software agents can interact with the architecture via special 'assistants' which are software wrappers designed

to create an abstraction from the architecture.

3.3.9 ITED (Texas Instruments)

A Concurrent Engineering design environment for a printed wiring board DFM system has been developed by Texas Instruments (Amundsen and Hutchison 1990). This system, called ITED (Integrated Tools for Electronic Design), consists of the following components:

- a number of CAD / CAE tools including symbol editor, library browser, simulator, timing analyser, schematic editor, DFM tools, auto-router and layout editor
- a knowledge-base inference engine which can be applied to solve design problems
- a common product representation that supports all stages of printed wiring board design.

The key to this system is the common information model and object-oriented data representation that supports hierarchical configuration and multiple levels of data abstraction. This implies that any tool in the environment can access any parameter in the development process, with changes being propagated to all other tools. When a change is made, the inference engine will undertake any necessary calculations to ensure the integrity of the design. The knowledge base was built by interviewing a team of experts from a number of relevant fields and translating this expertise into rules and code. This knowledge base can be updated and amended as and when needed. ITED is implemented in the Common LISP language with the inference engine written in PROLOG and the software runs on Texas Instruments Explorer machines.

3.3.10 Madefast

Madefast is an ARPA-sponsored project to demonstrate technology developed under the MADE (Manufacturing Automation and Design Engineering) program (Cutkosky et al. 1996). Madefast makes extensive use of the World-Wide Web for collaborating and archiving product model information. Tools within this environment are made accessible via the WWW and include design, analysis, simulation and rapid prototyping. These tools can be run as batch jobs or interactively. Special WWW-based software for synchronous and asynchronous communication are also provided in addition to a HTML authoring system. Documentation produced using the environment is in HTML format while product models have no prescribed format relying instead on data translation between tools.

3.3.11 MOSES

The Model Oriented Simultaneous Engineering System (MOSES) architecture (Molina et al. 1995b) is based on two information models, a product model and a manufacturing model linked via an integration environment (see Figure 3-6.) MOSES uses an object-oriented database to store both of these information models. The integration environment provides support for interactions and communication between applications that may require support from translators or special software wrappers. The engineering moderator is "...a specialist manager or co-ordinating program who role is to drive concurrency within the MOSES system" (Harding and Popplewell 1996). It is used to provide conflict notification and negotiation. The designers claim that MOSES has been designed in a modular fashion so that additional application domains or 'environments' can be catered for as and when is needed.

3.3.12 PACT

The Palo Alto Collaborative Testbed (PACT) is a prototype framework for integrating existing Concurrent Engineering systems (Cutkosky et al. 1993). At the core of this framework are agents and facilitators. Agents are the services (systems) that can be plugged into the framework while facilitators act as the interface between local and remote agents by providing a message passing and agent monitoring service. Agents communicate with each other using Knowledge Query and Manipulation Language (KQML) and Knowledge Interchange Format (KIF) on the assumption of an explicit information model. PACT has been implemented on a distributed messaging architecture that uses the TCP/IP transport protocol thereby allowing collaboration over the Internet.

3.3.13 Shastra

Shastra is described by its designers (Anupam and Bajaj 1994) as an "...extensible, collaborative, distributed, geometric design and scientific manipulation environment." The core of the Shastra architecture is composed of two layers or 'substrates': a collaboration substrate and a distribution substrate. Interaction with these substrate comes from three interfaces: a GUI interface for use by applications, an ASCII interface for shell-like front-ends and a network interface for use by Shastra's own proprietary network protocol. The environment is extensible via 'Toolkits' which adhere to the Shastra Application Architecture specifications. These toolkits provide functionality to users of the environment.

3.3.14 WISE

The Web Integrated Software Environment (Callahan et al. 1995) is an Internet-based project management and metrics tool. The WISE project is sponsored through a co-operative research agreement between the NASA Software IV&V Facility and the Concurrent Engineering Research Centre (CERC) at West Virginia University. It is an automated change management system that can be used as a communication tool to provide feedback on CE projects. At the core of WISE is a relational database that can be accessed via the World-Wide Web using standard HTML-based forms. Sitting between the database system and the user is a set of software applications that translate information sent by users to the database and vice versa

3.4 Evaluation of current CE support systems

The barriers to CE highlighted in section 3.2.1 may well be a considerable cause of the problems currently experienced in CE support systems, but what of the symptoms that manifest as a result of these problems? This section provides an evaluation of existing CE support systems to draw attention to these symptoms so that future environments can avoid reproducing them.

Global co-operation between partner companies engaged in CE requires some means for linking each partners' computer networks so that information can be shared across organisation boundaries. The lack of support for inter-networking can restrict the interoperability of CE organisations—I-CARE (Santos and Cardoso 1993), ITED (Amundsen and Hutchison 1990) and iDCSS (Klein 1996) all suffer from this symptom.

The common product information model is the cornerstone of successful virtual team working. Proprietary, non-standard information models may hinder an environment's ability to be fully integrated with existing or new applications, e.g. by making translation necessary—I-CARE (Santos and Cardoso 1993) integrates commercially available software

packages which invariably use proprietary data formats; PACT (Cutkosky et al. 1993), Madefast (Cutkosky et al. 1996), BSCW (Bentley et al. 1996) and COMBINE (Merabti and Carew 1994) require integrated applications to explicitly agree on a known format or apply translation procedures; DMMS (Bounab et al. 1993), iDCSS (Klein 1996) and ITED (Amundsen and Hutchison 1990) provide proprietary data formats.

Many existing support systems suffer from being tied to a single problem domain thereby limiting their usefulness as a generic support tool for CE—I-CARE (Santos and Cardoso 1993) has only been applied to civil and mechanical engineering; CoConut (Kress 1996) is only applicable to CAD development (although the authors claim that the concept of CoConut can be applied to other areas); Shastra (Anupam and Bajaj 1994) is tied to the domain of scientific design; (Amundsen and Hutchison 1990) only supports printed wiring board design; WISE (Callahan et al. 1995) specifically addresses software development.

Environments must cater for the entire team, not just single users. Failure to adequately address the needs of multiple users can reduce the visibility of group interactions such as changes to a shared document—the DMMS architecture (Bounab et al. 1993) only models shared data between tools with no consideration given to the users of the environment; similarly, iDCSS (Klein 1996) omits team structure from its architecture. Another hindrance to current support environments is that they are only applicable to portions of a product's life cycle—current environments typically only support the design phase (e.g. Amundsen and Hutchison 1990; Callahan et al. 1995; Kress 1996), the design and analysis phases (e.g. Cutkosky et al. 1993; Santos and Cardoso 1993) or design and manufacturing phases (e.g. Erkes et al. 1996); systems such as BSCW (Bentley et al. 1996) only provides basic support for collaboration while WISE (Callahan et al. 1995) only supports project management.

A number of environments differ in the database model used to store the product information model. Again, this can cause interoperability problems when two different environments need to be joined for collaboration purposes—some environments (e.g. Bounab et al. 1993; Callahan et al. 1995; Santos and Cardoso 1993) favour the relational model while others prefer the object-oriented model (e.g. Kress 1996; Molina et al. 1995b); others such as PACT (Cutkosky et al. 1993) and COMBINE (Merabti and Carew 1994) omit the notion of a logically central database leaving data management to individual applications; Madefast (Cutkosky et al. 1996) and the CE-toolkit (Erkes et al. 1996) choose to use multiple data repositories at a multiple participant sites.

Environments based on open standards generally tend to provide more flexibility and extensibility to extend existing functionality with new tools. Object-oriented environments

- *The lack of security measures*—This is evident in both the transmission of data and authentication of users (BCS Security Committee 1995; Erkes et al. 1996).

- *The lack of service guarantees*—This leads to poor reliability (Erkes et al. 1996).

- *The lack of quality of service (QoS) guarantees in HTTP*—This prevents real-time dynamic data, such as that produced during virtual prototyping and videoconferencing, from being used effectively. In addition to this, there is a scarcity of protocols and tools for sharing and collaboration using HTTP (Hanneghan et al. 1997).

The most widely used method of attempting to overcome the limitations of the WWW is to provide non-standard 'add-in' software that must be used by all team members. This only serves to hinder to openness of the system as a whole by tying users to specific and sometime proprietary solutions. The WWW can perform a useful secondary role however, for example, Wingrove et al. (1997) describe the development of a concurrent engineering handbook using HTML. This handbook is a continuous document of the process and results of a given CE project and imposes a rigid structure on the documentation. This handbook concept does not lend itself to highly dynamic CE environments but can be of use as a means for recording and browsing past project histories. Project histories are made up of a large number of 'after-the-fact' static documents.

3.5 Summary

In the previous chapter, a number of technological considerations were introduced that impact on how computer-based support systems are built. A number of additional issues have been raised in this chapter for the building of the next generation of CE support systems. These issues have arisen from the requirement to overcome the current barriers to, and limitations of, existing CE support environments. This chapter concludes with a state-of-the-art summary of requirements for next generation CE support systems.

This research proposes that support environments should provide at least the following functionality:

a) Provide a communication sub-system for effective participation (both asynchronous and synchronous) between all people involved in a CE project. This includes facilities for controlling and tracking the project tasks and resources including people. Efforts in the field of CSCW (see chapter 2) are a critical success factor to this requirement.

b) Provide the ability to treat geographically dispersed resources as though they were local. Virtual CE teams may share resources in different physical sites. This facility must be an intrinsic part of any support environment.

c) Provide centralised, accountable data management facilities that can be used to store any artefact produced during the entire product life cycle.

d) Enable transparent access to all the necessary software tools and applications needed throughout the development process e.g. CAD, CAM, CAE and project administration tools. Seamless integration from any location is the goal here.

e) Support a consistent user interface even across multiple computer platforms and architectures.

f) Include the ability for the environment to be extended to include further tools and technologies as and when they become available. This brings with it the requirement for adoption of open standards.

g) Be sufficiently generic and portable between computer platforms to enjoy widespread adoption and improve future maintainability. Again, this requires adherence to open standards.

h) Cater for the large number of legacy applications already in use in order to protect current investments. Data format translation will remain a 'necessary evil' until the STEP community can persuade software developers otherwise.

i) Have facilities for inter-networking since this is the foundation of cross organisation virtual team working.

With these factors in mind, the following three chapters propose and develop an architecture that can be used to help facilitate the building of the next generation of support environments for CE. As a basis for this new architecture and to re-examine the issues raised here in greater detail, a robust reference model derived using the technique of viewpoint analysis is described in the following chapter.

Chapter 4

"When we mean to build,
We first survey the plot, then draw the model"
William Shakespeare (1564–1616) Henry IV, Part Two, *I:3*

4. A reference model for building CE support systems

4.1 Introduction

Requirements analysis is a complex task. All too often, computer systems are built and delivered that do not fulfil users' requirements. In software engineering, viewpoint analysis (Finkelstein et al. 1992) has been proposed as a means of successfully capturing user requirements for complex systems. Viewpoint analysis is a process that produces a synthesis of requirements from a number of disparate and distinct perspectives. This chapter describes such a viewpoint analysis that has been applied to the field of Concurrent Engineering. A viewpoint reference model is proposed which draws on the experiences of a number of existing models from fields such as manufacturing, information systems and distributed computing. This new reference model, the CE-RM, has been designed using object-oriented methods and has been used as a basis for building a Concurrent Engineering support environment which is the subject of the following two chapters. The CE-RM is discussed and compared with a number of existing models that have been proposed to capture requirements in the manufacturing and / or engineering domain in order to validate the viewpoint analysis approach.

4.2 Background

In the design of systems, a viewpoint is the self-contained and complete perspective of a stakeholder (person or system) that is directly affected by the system under investigation. From an object-oriented perspective, Kotonya and Sommerville (1992) describe a viewpoint as "an external entity that interacts with the system being analysed, but one which can exist without the presence of the system." In modelling the requirements of any new system, multiple ideas and perspectives must be analysed and resolved. Invariably, conflicts of opinion and omissions cause requirements specifications to incompletely capture a system's proposed behaviour (Gruia-Catalin 1985). Finkelstein et al. (1992) endorse viewpoint analysis as a means for formulating requirements definitions for large and complex systems.

In Concurrent Engineering projects, we find what has been termed "the multiple perspective problem" (Easterbrook et al. 1994) with many actors, sundry representation schemes, diverse domain knowledge and differing development strategies. Smith (1988), cited in Sohlenius (1992), also states that co-operation of multiple competing perspectives and the integration of complementary engineering expertise are requirements for systems that intend to support Concurrent Engineering. This research contends that viewpoint analysis can therefore be used to help design the complex environments necessary to support Concurrent Engineering.

Over the last decade the concept of manufacturing 'architectures' has evolved to provide structure for the analysis and design of manufacturing enterprises. Early examples include the European ESPRIT phase-I CIM project (Yeomans 1987) which provided flowcharts and text descriptions of the generic activities that comprise the machining sector of manufacturing. It is claimed to provide 'a European Computer Integrated Manufacturing (CIM) architecture against which IT vendors could fashion CIM products'. Subsequently much of the European ESPRIT phase-II research was aimed at developing open-systems CIM architectures and communications to support multi-vendor environments. In the USA the ICAM architecture (US Air Force 1981) was taking a hierarchical architectural approach to do the same thing. The development of the ISO Manufacturing Automation Protocol (MAP) grew from a proposal (ISO 1986) to 'create a multi-dimensional, open ended reference architecture and provide a basis for long-range planning and standardisation through the identification of interfaces and their characteristics, electrical, mechanical, man-machine, information, procedural language, etc.'. Large manufacturing system vendors have also proposed architectures (IBM 1987) as frameworks to develop computer-based manufacturing.

The scope of the proposed architectures is limited, in the case of ISO to 'discrete parts manufacture'. The ESPRIT CIM project was specifically aimed at mechanical engineering and machining operations 'because there are more manufacturing organisations within Europe involved in machining operations than any other single type of manufacturing and machining represents the largest market for CIM system vendors' (Yeomans 1987). The GRAI architecture (Doumeingts et al. 1992) is restricted to a 'production management system'. Subsequently many 'architectures' have been proposed (Colquhoun et al. 1996; Davis and Jones 1989; Graefe and Thomson 1989; Jorysz and Vernadat 1990; Klittich 1990; Los et al. 1992; Scheer 1992; Weston 1995; Zachman 1987) for manufacturing enterprise applications.

A common thread to these architectures is that they use graphical models to represent the various aspects of manufacturing such as processes or functions and the logic or sequence of information flow (documents, verbal or data) that link and control them. Such modelling methods are characterised by a formal syntax and structured diagramming techniques and are based on concepts from General Systems Theory and software development methods. In practice this means that models can describe a complex manufacturing system (consisting of people, machines, material, products, data, etc.) in easily understood, related elements using a series of diagrams. A model can then be used as a common understanding of a complex situation, for gaining insight, for system design or as the basis of quantitative analysis. However, in the models described above it would appear that no single model sufficiently addresses all the needs of Concurrent Engineering, forcing the designers of CE support environments to use multiple, separate, and sometimes contrasting models to achieve their goal. When using multiple models, there is the risk that information from a particular model might be missed or misinterpreted in a following model when that information is transcribed. The challenge then, is to produce a model that can address all the needs specific to Concurrent Engineering yet which is also flexible enough to be extended to other domains with relative ease.

The specific needs of CE are highlighted in the author's definition of Concurrent Engineering given in chapter 1 and include: enriched communication between team members, cross-functional team-working and distributed access to resources, such as product information and software applications, to multiple team members at multiple locations. The bounds of these functions stretches from initial concept of a product through to its disposal so any model designed to support CE must be applicable for all stages in the product life cycle.

An important and related effort has been targeted at the general area of Distributed Systems and has also produced a reference model. The Reference Model for Open Distributed Processing (RM-ODP) (ISO 1992a) is an architectural framework which defines five viewpoints that can be used to identify and integrate open distributed processing standards. Each of the five viewpoints: *Enterprise*, *Information*, *Computation*, *Engineering* and *Technology* is a complete and self-contained perspective on distributed systems in a terminology appropriate to a particular interested party and focuses on a unique aspect of the system under investigation (see Table 4-1.)

Table 4-1. Viewpoints specified in the RM-ODP.

Viewpoint	*Focus*
Enterprise	Purpose, scope and policies for the system
Information	The semantics of information and information processing activities in the system
Computational	Functional decomposition into structures suitable for distribution
Engineering	Functions to support distribution in the system
Technology	The actual choice of technology to be used in the final system

The RM-ODP is an object-based framework in that it loosely uses the concept of an object to describe each viewpoint. Although sometimes described as such (see for example Farooqui et al. 1995), the RM-ODP cannot really be considered as an object-oriented model since it does not support features such as abstraction, inheritance, aggregation and polymorphism that are present in true object-orientated models. This widely accepted model has been shown to be useful in the design and implementation of complex distributed systems, see for example the ANSAware software product (ANSA 1989) and (Coulson 1993).

4.3 Identification of viewpoints to support CE

A reference model describes a complex system in terms of its functions and information content. In order to present a new reference model for CE, viewpoint *partitions* are described in object-oriented terms so that object-oriented analysis techniques such as inheritance can be used to define super and sub-types of viewpoint. This brings the flexibility to devise separate self-contained viewpoints that can reuse, i.e. inherit, common aspects or attributes. Object-orientation also provides aggregation to facilitate the building of more complex viewpoints by defining smaller, more manageable viewpoints and then combining or synthesising these to produce the whole. This approach has been influenced by Kotonya and Sommerville (1992).

In order to illustrate the viewpoints required to support CE, the following scenario taken from product development in the aerospace industry (Cutkosky et al. 1996) will be used.

4.3.1 *Scenario*

In the design and manufacture of a strategically important product, such as an air-to-air missile, it is necessary for cross-functional team members to collaborate. This is illustrated as an event flow diagram in Figure 4-1. For strategic alliances between different countries e.g. those associated with NATO (the North Atlantic Treaty Organisation), non-collocation is an important issue that also needs to be addressed. Any project begins by defining the tasks that need to be performed. Multiple team members then concurrently perform these tasks. This involves the creation of some new item, the group discussion of this item, followed by either the acceptance of the item or a re-iteration of the process in which the item is modified (i.e. a new version is produced) until the group arrives at consensus.

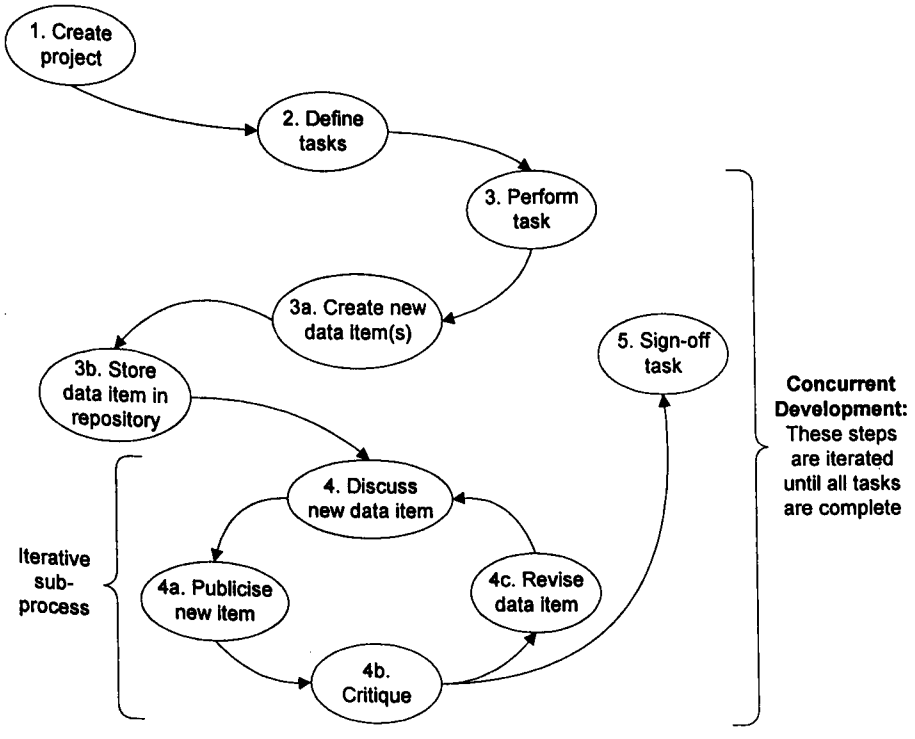


Figure 4-1. Typical event flow for concurrent development.

Constraining a project such as this will be a strict set of security guidelines that specify how information should be transmitted in addition to restricting the channels in which communication should take place.

4.3.2 *A Concurrent Engineering Reference Model (CE-RM)*

The viewpoints proposed in order to support CE are described below in relation to the RM-ODP and are shown graphically in Figure 4-2; the CE-RM (Concurrent Engineering Reference Model). Whereas the RM-ODP has only five distinct viewpoints, the CE-RM further sub-divides or partitions these five viewpoints to cater specifically for the needs of

4.3.1 *Scenario*

In the design and manufacture of a strategically important product, such as an air-to-air missile, it is necessary for cross-functional team members to collaborate. This is illustrated as an event flow diagram in Figure 4-1. For strategic alliances between different countries e.g. those associated with NATO (the North Atlantic Treaty Organisation), non-collocation is an important issue that also needs to be addressed. Any project begins by defining the tasks that need to be performed. Multiple team members then concurrently perform these tasks. This involves the creation of some new item, the group discussion of this item, followed by either the acceptance of the item or a re-iteration of the process in which the item is modified (i.e. a new version is produced) until the group arrives at consensus.

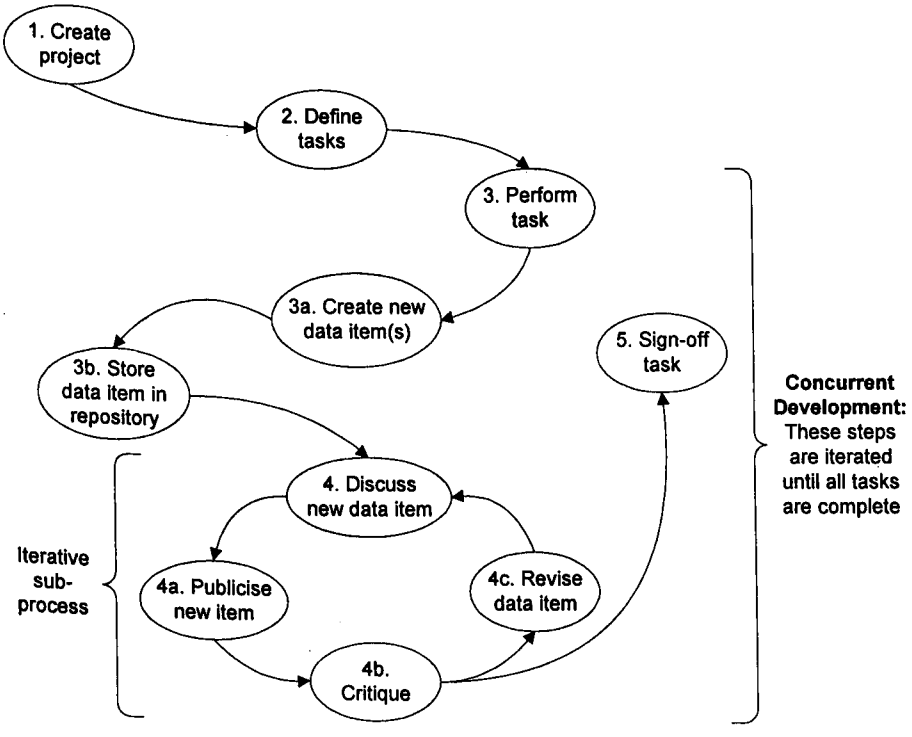


Figure 4-1. Typical event flow for concurrent development.

Constraining a project such as this will be a strict set of security guidelines that specify how information should be transmitted in addition to restricting the channels in which communication should take place.

4.3.2 *A Concurrent Engineering Reference Model (CE-RM)*

The viewpoints proposed in order to support CE are described below in relation to the RM-ODP and are shown graphically in Figure 4-2; the CE-RM (Concurrent Engineering Reference Model). Whereas the RM-ODP has only five distinct viewpoints, the CE-RM further sub-divides or partitions these five viewpoints to cater specifically for the needs of

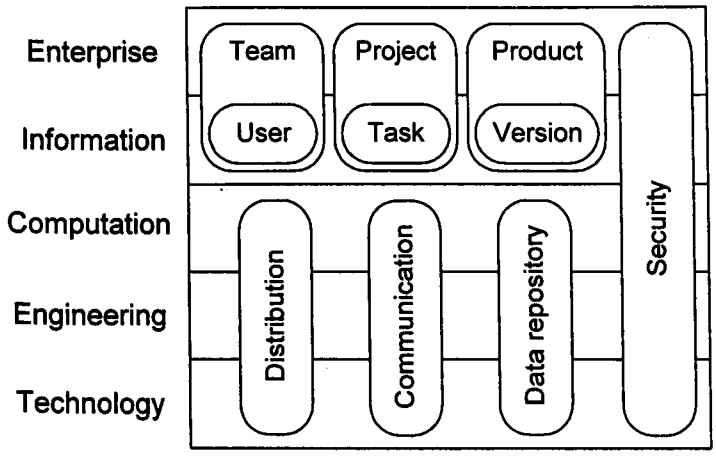


Figure 4-2. The Concurrent Engineering Reference Model (CE-RM).

Concurrent Engineering projects by remodelling the RM-ODP viewpoints using the technique of object-orientation. In the CE-RM, the five RM-ODP base viewpoints (i.e. enterprise, information, computation, engineering and technology) are retained as the fundamental base viewpoint super-types. These are then further analysed to determine sub-types of these viewpoints which have more specific properties concerned with the problem domain, i.e. Concurrent Engineering. By doing this, a number of aggregate viewpoint objects are also introduced. Viewpoints can be classified into *functional* and *non-functional* viewpoints (Mullery 1979). Non-functional viewpoints define constraints that must be considered in the final system design.

At the *Enterprise* level, the following viewpoints have been identified:

- TEAM,
- PROJECT,
- PRODUCT and
- SECURITY.

Security is an important aspect of any virtual enterprise undertaking a CE project and is considered as a non-functional viewpoint, i.e. it is a constraint that must be imposed in the final system design. The SECURITY viewpoint at the Enterprise level focuses on enterprise-wide security policies and thus determines the level of security mandated in lower levels of the model. The SECURITY viewpoint will be seen to be pervasive throughout the subsequent lower levels of the model. Security issues are extremely important to organisations

employing CE in highly competitive or highly sensitive markets, for example to prevent competitors acquiring detailed product information.

The TEAM viewpoint is concerned with the collection of participants (team members) involved in a CE project. These team members operate with a high level of cohesion and therefore this viewpoint is used to capture the perspective of this collective team unit. This viewpoint accommodates not only physically collocated teams but also non-collocated teams and teams whose structure changes dynamically, so-called *Virtual Teams* (Trapp et al. 1992). The characteristics of virtual teams are captured using the inherent object-oriented capabilities of our model that supports polymorphism and inheritance. This allows teams to be built from well-tested reusable constructs while retaining their own unique characteristics.

The PROJECT viewpoint is concerned with not only the current CE project but also with past projects. One desirable aspect of Concurrent Engineering is to be able to reuse project histories, i.e. design and manufacturing information, data and procedures from previous projects, in order to benefit the current project. Design or manufacturing problems from previous projects needs to be avoided while solutions to problems in previous projects need to be promoted.

The PRODUCT viewpoint takes into consideration the needs of the product through each phase of its life cycle. An end product is composed of many sub-components. The PRODUCT viewpoint considers these components and what happens to them during their individual lifetimes.

At the *Information* level, the following viewpoints, which are aggregate components[11] of the respective viewpoints identified at the Enterprise level, are identified:

- USER,
- TASK,
- VERSION and
- SECURITY.

[11] The aggregation of viewpoint object types is denoted in the CE-RM by the aggregate or whole structure surrounding the part structure.

The USER viewpoint is concerned with capturing the requirements of individual team members. This is important because within a project there may be tasks or sub-tasks that require the services of only one person, i.e. the team member acts autonomously with no collaboration from their co-workers. Autonomy is also a characteristic of distributed systems where multiple autonomous agents simultaneously perform some role while interacting with others whenever necessary.

The TASK viewpoint considers the fact that a project can be broken down and partitioned. Large, complex projects are divided into smaller, more manageable tasks and sub-tasks. This viewpoint and its super-type considers the system from a project management perspective. The PROJECT and TASK viewpoints help manage the complexity of CE project management which can be hindered by the fact that some team members could be located in different countries.

The VERSION viewpoint and its associated super-type, PRODUCT, takes into account how a final product is created via a process of revision, e.g. an individual component may have been developed through many versions before a final agreement is met as to its specification. This viewpoint needs to be considered since it would be undesirable for a designer, for example, to work on an out-of-date version of a product specification which could lead to a loss of data integrity and increased data redundancy (Amundsen and Hutchison 1990).

The SECURITY viewpoint is needed at the Information level to ensure that information does not get into the wrong hands or even to ensure that a particular team member cannot erroneously change information without adequate authority. For example, a possibly disastrous situation could arise if sales personnel were allowed to alter a process plan. As a non-functional viewpoint, the SECURITY viewpoint at this level specifies access control constraints.

The *Computation* level consists of the following viewpoints:

- DISTRIBUTION,
- COMMUNICATION
- DATA REPOSITORY, and
- SECURITY.

Successful CE requires distributed processing; the DISTRIBUTION viewpoint takes into consideration the fact that users and resources internal and external to the system are geographically distributed. It considers the system as being composed of a number of distributed nodes. The COMMUNICATION viewpoint considers the system from the point of view that CE team members need to communicate in order to perform collaborative work and to behave as a virtual team. This communication can be in two forms: *synchronous* (i.e. taking place at the same time, for example a telephone conversation) or *asynchronous* (i.e. not at the same time, for example, mailing a letter). The DATA REPOSITORY viewpoint is concerned with the notions that information sharing, data integrity and the prevention of data redundancy are key aspects of CE data management (Hanneghan et al. 1995). Concurrent Engineering is concerned with the wide publication and consensus of information throughout a project. This viewpoint is influenced by the need to access and publish data that is centrally available to all team members. The relevance of the SECURITY viewpoint here relates to the need to prevent unauthorised viewing of data as it is passed around the network by way of cryptography constraints.

At the *Engineering* level, the four viewpoints of DISTRIBUTION, COMMUNICATION, DATA REPOSITORY and SECURITY are again present. At this point in the model, the functions required to support these viewpoints are specified. Security considerations at this level relate to the need to control distributed access to the system and prevent unauthorised access.

At the *Technology* level, the four viewpoints DISTRIBUTION, COMMUNICATION, DATA REPOSITORY and SECURITY again permeate through the model. Each of these viewpoints considers the technological choices for implementing the system. For example, in the DATA REPOSITORY viewpoint a system designer may consider relational databases, object databases or object-relational databases as reasonable choices for implementation. Similarly, in the DISTRIBUTION viewpoint, the Object Management Group's Common Object Request Broker Architecture (CORBA) (Object Management Group 1995a) or Open Software Foundation's Distributed Computing Environment (DCE) (Open Software Foundation 1993) models may be considered as possible choices for implementation. For the SECURITY viewpoint, implementation choices could consider data encryption standards such as DES or MD5.

4.3.3 Evaluation of the CE-RM

A principal part of the CE-RM is that it specifies viewpoints spanning the five ODP dimensions. This means that each viewpoint pervades down through successive levels to fully specify its function. For example, *Security* can be seen to pervade through and affect

the organisational, information, computational, engineering and technology viewpoints specified in the CE-RM. Another feature, as a result of object-oriented modelling, is that characteristics of viewpoints are inherited by viewpoint sub-classes. A major example of this can be seen in the model as a whole, which is itself, a sub-class of the *Concurrent Engineering* viewpoint (i.e. the perspective of CE as specified by its definition or the term). Therefore, characteristics such as concurrency, process improvement, quality enhancement and development time reductions are passed on to the various sub-classes of the model.

Returning to the scenario described in §4.3.1, the viewpoints defined in the CE-RM can now be put in context. In a particular *Project* a team member (*User*) performs various *Tasks*. In order for a *User* to work on some *Version* of a *Product* specification, the physical location of the *User* must be taken into account and what resources that *User* has access to. This is handled by the *Distribution* component in conjunction with the constraints laid down by both the *Security* component and the *Data repository* component. Determining where the actual data object is located and whether any other *Team* member is currently using the same object (the *Data repository* component) or is likely to be affected by the operation (by way of the *Communication* component in co-operation with the *Distribution* component) is also necessary. Affected parties (*Users*) will need to be notified (using the *Communication* component). Once a data item is produced, the CE Team then engage in discussion (using the *Communication* and *Distribution* components) in an effort to achieve consensus about the item's definition. It is essential that this entire process be secure and free from tampering by outside agents (the role of the *Security* component).

4.4 Comparison of the CE-RM with related work

Zhang and Alting (1992) describe a functional model of manufacturing enterprises that comprises three views. These views are: *Management*, *Engineering* and *Fabrication*. The Management functions consider the tasks of planning, analysing and controlling, while the Engineering functions consider definition, formulation and evaluation tasks. Finally, the Fabrication functions consider organisation, execution and distribution.

Although not described as such, these views can be considered in object-oriented terms as aggregate viewpoint objects. For example, the Management viewpoint object has primary operations to *Plan*, *Analyse* and *Control* while its attributes (which themselves are lower-level viewpoint objects) include *Sales and marketing*, *Finance and accounting*, *Production planning and control*, *Policy-making / strategic planning* and *Personnel*. To illustrate this example, Figure 4-3 shows the viewpoint object definition.

Similar view-based models can be found in the Architecture of Integrated Information Systems (ARIS) (Scheer 1992) and Zachman's framework (Zachman 1987). There are four views in ARIS: a Data view, a Control view, a Function view and an Organisation view where each view is supported by a specific tool in the ARIS TOOLSET (Scheer 1996). Zachman uses three major perspectives to describe enterprise-overlapping systems: Data (what the system deals with,) Process (how the system works) and Network (where flows and connections exist). Each of these views consists of six representations: Scope and Objective, Business, Information system, Technology, Machine code and Actual system. The combination of these views and representations creates a two-dimensional matrix where each cell represents the intersection of an architectural representation from a certain perspective.

The major limitation of these models when applied in a CE context is that they do not explicitly support team interactions throughout the enterprise. Nor do they adequately address the notion that an enterprise may exist virtually with departments situated at different physical sites. It should also be noted that these models are strictly of a functional nature, i.e. there is no attempt to address non-functional viewpoints. A number of researchers have stated that non-functional or indirect viewpoints often have significant influence within an organisation (for example Kotonya and Sommerville 1996). We agree with this and consider the omission of non-functional viewpoints to be a limitation of models such as those described above in the context of this research.

An analogy to viewpoints are proposed by Harding and Popplewell (1996) as a means for requirements capture in the MOSES (Molina et al. 1995b) CAE architecture. The viewpoints considered make up a three-by-three matrix as shown in Figure 4-4. Along the *x*-

axis of this matrix are the three levels of co-operation that can be achieved by team members involved in a CE project. It considers the individual member, the team and the organisation. Along the *y*-axis, it considers three dimensions that should be addressed to support CE: distribution, heterogeneity and autonomy.

The *autonomy* dimension is analogous to the COMMUNICATION viewpoint of the CE-RM discussed earlier in that they both specify support for synchronous and asynchronous communication and decision making within CE teams. The three levels of co-operation are also captured in the CE-RM. We explicitly define individual user and team viewpoints and implicitly define an organisation viewpoint by encapsulating the *Enterprise* viewpoint. We believe that the object-oriented sub-classing in the CE-RM makes for more robust viewpoints since the designer is forced to think first in terms of the five fundamental RM-ODP viewpoints and then to look at these in more detail to extract useful sub-classes.

One limitation of this matrix model however, is that it does not explicitly specify non-functional requirements or viewpoints. The advantage of viewpoint analysis is that these non-functional viewpoints are given equal consideration in an effort to improve the quality of the requirements capture phase of system development. Again we believe that the omission of non-functional viewpoints is a weakness that needs to be addressed if this model is to be used to build robust support environments for CE.

The development of the Open System Architecture for Computer Integrated Manufacturing (CIM-OSA) is a result of ESPRIT Project 688, AMICE (1989). It defines an integrated methodology to support all phases of a CIM system life cycle from requirement specification through system design and implementation to operation and maintenance. The CIM-OSA framework can therefore be used to model organisations employing Concurrent Engineering. The CIM-OSA modelling framework or 'CIM-OSA cube' shown in Figure 4-5, uses the three axes of a cube to represent the concept of the architecture:

a) The instantiation process is a design principle going from generic to particular and re-uses previous solutions.

b) The derivation process is a design principle which forces analysts to adopt a structured approach to system design and implementation, going from requirements definition to design specification and full implementation description.

c) The generation process is a design principle that encourages users to think about the total enterprise in terms of function, information, resources, and organisations.

CIM-OSA takes into account a number of well-accepted ideas and principles such as

functional decomposition and activity modelling used in SADT (Ross 1977) or IDEF0 (US Air Force 1981) and the entity-relationship model (Chen 1976). In common with other approaches, CIM-OSA uses different views to describe a manufacturing system.

The techniques of object-orientation also cater for instantiation through the use of abstract or generic objects which can be used to instantiate more specific sub-classes of these objects. Object-orientation has also been shown to uniformly model requirements analysis (Coad and Yourdon 1991a; Firesmith 1993), design (Booch 1991; Coad and Yourdon 1991b) and implementation (Goldberg 1984; Gosling and McGilton 1995; Stroustrup 1991). It can be seen then, that by consistently using object-oriented methods we can also achieve derivation. This derivation is possible because the same object models and modelling techniques can be applied to requirements analysis through design and implementation: the system is derived by progressively enhancing the model at each stage.

4.5 Summary

Requirements analysis for complex systems can be a difficult task. The design and implementation of systems to support Concurrent Engineering (CE) projects involving virtual teams is one such example of this. Previous research in this field has largely concentrated on creating architectures and reference models for Computer-Aided Manufacturing and Engineering domains operating a purely sequential development process which do not take specific CE needs such as enriched communication, information sharing and distributed working into consideration. Additionally, these existing architectures and reference models typically specify modelling methods that are different at each stage in a project. For example, models such as the CIM-OSA model which freely allows the developer to use a tool such Data Flow Diagrams (DFDs) for information systems modelling yet use IDEF0 for physical system modelling. These multiple models can lead to loss of information and clarity when transforming models from one stage to another.

To help address the shortcomings of these existing models, the technique of viewpoint analysis has been applied in an effort to produce a more realistic and practical model. By uniformly applying object-oriented techniques and drawing on existing work from the manufacturing / engineering field and other related fields such as open distributed processing, a new specific reference model for Concurrent Engineering, the CE-RM, has been devised in which requirements capture is only part of a wider framework. By adopting the object-oriented paradigm a consistent model can be used for specifying system requirements, system design and implementing the actual system. This has saved effort in development due to the fact that no transformation of the model is needed between stages and also because objects can be reused by subsequent stages without re-definition. This also

improves the integrity of information, as there is less chance for information to be misinterpreted or lost as a consequence of the model translation process. Object-orientation also copes well when modelling distributed resources since objects communicate uniformly using a message-passing paradigm regardless of whether those objects exist locally or remotely at different physical sites. It is the belief of the author that the technique described in this chapter for partitioning viewpoints into viewpoint objects can be applied to other problem domains, not just Concurrent Engineering.

To validate the viewpoints identified in the CE-RM, a computer systems architectural model and an information system object model have been developed and these topics are the subject of the following two chapters. The relation between these three chapters is shown in Figure 4-6.

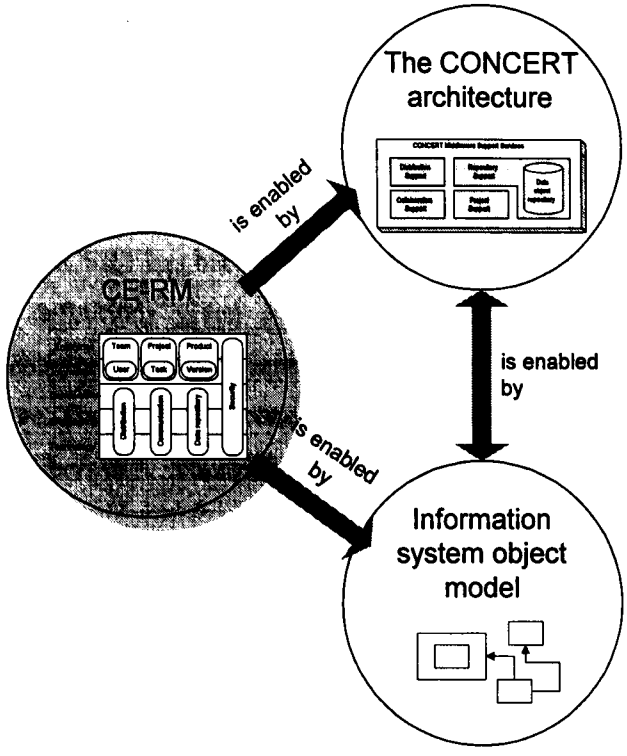


Figure 4-6. Relationship between CE-RM and work introduced in the following chapters.

Chapter 5

concert n & v. meaning n. 1 a musical performance of usu. several separate compositions. 2 agreement, accordance, harmony. 3 a combination of voices or sounds. meaning v. tr. arrange (by mutual agreement or co-ordination).

***in concert** idiom. acting jointly and accordantly.*

(Concise Oxford Dictionary)

***CONCERT** acronym[12]. CONCurrent Engineering suppoRT*

5. A computer systems architecture to support virtual CE teams

5.1 Introduction

This chapter describes a computer system architecture developed to support virtual concurrent engineering teams and which is based on the CE-RM presented in the previous chapter. The name CONCERT has been coined for this architecture as it shares a number of meanings with the definitions given above. Concurrent engineering usually involves '...several separate compositions' and requires '...agreement, accordance' and '...harmony' in order to be effective. The process of concurrent engineering can be seen to be the work of a number of team members or '...combination of voices' '...acting jointly and accordantly.'

The discussion of the CONCERT architecture in this chapter will adopt a bottom-up approach by firstly describing the policies and considerations that have been adopted by the components of the architecture before going on to describe the actual components themselves. Finally, the integration of the architecture with a real world environment, the CONCERT environment is described. The relationship between the CONCERT architecture described in this chapter and the CE-RM in the previous chapter is once again shown in Figure 5-1.

[12] I apologise to any lexicologists for associating the word **'concert'** with such a poor acronym for the goal of this research.

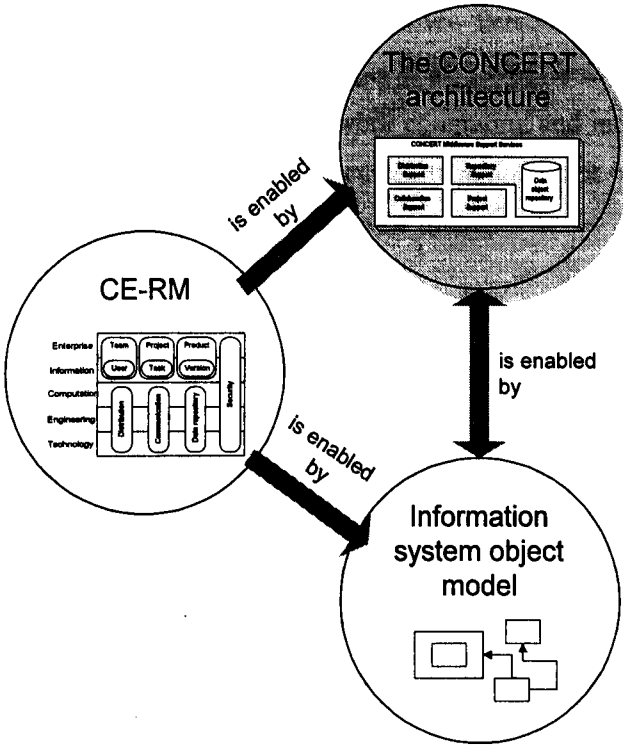


Figure 5-1. The CONCERT architecture overview.

5.2 Objectives of new architecture

The following objectives were pursued in the design of the CONCERT architecture:

- *Wide availability*—The architecture should be capable of being implemented in such a way that existing engineering systems could employ the technology without the need for heavy investment in additional resources.
- *Scalability*—The architecture should be capable of being used in both small and large organisations.
- *Extensibility*—The architecture should allow future expansion with relative ease.
- *Generality*—The architecture should use open standards to enable it to be used in conjunction with existing systems.
- *Architecture neutrality*—The architecture should be capable of being used unchanged on a number of popular hardware platforms.

The notions of *wide availability* and *architecture neutrality* stem from the fact that within a given project workgroup, there will be a broad variety of computer systems in use, ranging from (possibly) mainframes to workstations to high-powered PCs and dumb-terminals. Recent advances in miniaturisation and mobile communications technology has meant that

portable hand-held devices (sometimes known as PDAs—Personal Digital Assistants) are becoming commonplace within virtual teams (Finger et al. 1996; Gessler and Kotulla 1995). In addition to this, an industry shift towards network computing and indeed the generic NC or Network Computer (Apple et al. 1996) that is currently taking place means that these two notions will become increasingly important in the near future.

5.3 Policies and considerations for support services

The following section discusses the policy decisions that have been formalised to govern the new architecture. A number of factors have influenced the decisions that have been taken in designing the CONCERT architecture. These factors relate to the complex nature of concurrent engineering projects and include conflict resolution, data translation, group consensus, long transactions, project management strategies, security requirements and versioning. The architectural decisions made as a result of these factors are described in the following sub-sections as formal policies for designers of computer systems to support CE.

5.3.1 Conflict resolution

It is widely considered that fully automated resolution of design conflicts is undesirable, and in some cases unfeasible, (see for example Bahler et al. 1994a; Bahler et al. 1994b; Harding and Popplewell 1996). With these factors in mind, a *mediator* metaphor has been chosen with regard to solving potential conflicts between team members (Goldstein 1994; Hori et al. 1996). When a conflict arises, the CONCERT support services should assist the parties involved to attempt to resolve the problem using negotiation and mediation.

5.3.2 Data formats and translation

In order to support data sharing between the large number of legacy applications (with legacy data formats) that currently exist in CE environments there is a need for data translation functionality that can be easily accessible by all tools within the CE support environment. This translation should be applicable to all objects stored in the data object repository and made centrally available to ensure consistency among users of the environment. To support this view, the data translation policy of the CONCERT architecture prescribes that:

I. All data translation tools be registered with a CONCERT support service.

II. These translation tools must declare an interface which describes which format they can translate from and which format they can translate to, and be able to take as input an arbitrary number of data bytes and return an arbitrary number of translated data bytes.

An example could be a translation tool that takes a *word processor document* and outputs a *HTML document.*

5.3.3 Group consensus

Successful CE methodologies should encourage team working and promote consensus amongst the team members. This ensures that whole life cycle issues are addressed at all stages. The goal of the CONCERT architecture is to achieve increased awareness and better communication between team members so that they can make empowered decisions that take into account the entire product life cycle. A method that can be utilised to help achieve this is through event and message channelling. This allows events such as key presses, mouse movements and changes to user interface windows or data fields to be 'channelled' to interested parties. This is also known as the 'publish and subscribe' metaphor: users subscribe to shared application channels and publish information to that channel. The information is then re-distributed to all subscribed listeners on that channel thereby allowing group collaboration. The CONCERT architecture therefore prescribes the use of publish and subscribe facilities.

5.3.4 Long transactions and concurrency control

Another factor that can affect information sharing within a CE organisation is that a particular process can use information for a long period of time. An example of this could be a design engineer who works on a CAD design for a whole day at a time. This can prevent other team members using the same information and thus hinder concurrency. This problem of long transactions exists in most, if not all, CE projects. Andrews and Krieger (1993) state that a: "semantic concurrency model combined with notification locks is an effective mechanism that supports concurrent collaborative work environments."

The concurrency control policy determines how multiple team members can access the same object simultaneously. There are two possible actions that a team member can perform on objects: they can view (*read*) the object where the intent *is not* modify the object's contents, or they can edit (*write*) an object where the intent *is* to modify the actual object. Issues that arise here are:

a) If a team member is reading an object and another team member changes that object, the first team member will be working with an out-of-date version.
b) If two team members edit (write) an object at the same time, there will be multiple versions of that object which may need to be reconciled into a composite object at some later stage.

With these considerations in mind, the CONCERT environment stipulates the following constraints. For a team member to work on an existing object requires the notification of an *intent to use* by the team member (this is known as the *check-out*) and subsequent notification of the team member's *re-submission* of the object (this is known as the *check-in*). This constitutes a '*check-in-check-out*' pairing that is necessary for the system to be able to closely monitor and improve object concurrency. If a team member aborts an edit session and does not want to re-submit the edited version back to the repository, they must inform the system that this is the case in order to release any locks that exist for the given object. If the team member neglects or simply forgets to do this, it should be possible for the system to determine all the long transactions that a given team member is involved in and notify the team member of any outstanding transactions.

The following statements define the concurrency control policy.

I. When a team member attempts to check-out an object, he/she must declare whether they intend to simply view (read) or edit (write) the object.

II. If a team member wishes to edit (write) a specific object that is already in use for writing by another team member they have the following options:

   A. Create a new version of the object (see section 5.3.7—*Versioning*) without affecting the current team member.

   B. Enter into negotiation with the existing team member as to who should write the object. The outcome of this negotiation process will be either:

      1. the existing team member abandons or finishes their current editing session and allows the new team member to use his/her edited version, or

      2. the new team member goes ahead and creates a new version regardless.

III. If a team member wishes to make changes to (i.e. write) a specific object that is already in use for reading by another team member the system should be able to inform the reading team member that the object they are reading is about to be updated and hence the current version of the object may change i.e. they will be using information that could potentially be out-of-date (see §5.3.7—*Versioning).*

A typical scenario that shows this policy in use is shown in Figure 5-2. This diagram shows the chain of possible events happening along some given timeline. At point 1, a new object called *CAD model*, say, is placed in the repository. Some time later at point 2, Bob wants to view *CAD model* and so checks it out of the repository for reading. Some time later at point 3, Mary decides to edit *CAD model* to create a new version. While Mary is doing this, Frank also decides to edit *CAD model* to create a new version (i.e. both Mary and Frank are now editing *CAD model*). Since Mary is already editing *CAD model*, the system must notify Frank that the requested object is already checked-out for writing since it is possible that edits could be duplicated by both Mary and Frank. This duplication of effort hinders the Concurrent Engineering process and so should be avoided whenever possible. The options open to Mary and Frank are as follows:

a) Frank can abort and let Mary continue her editing session.
b) Mary can abort and let Frank edit instead.
c) Mary can finish and commit her edit so that Frank can then work on Mary's new version of *CAD model*.
d) Mary can carry on regardless and let Frank edit CAD model also resulting in two versions of the same object which may need to be merged at some later stage (see §5.3.7—*Versioning*).

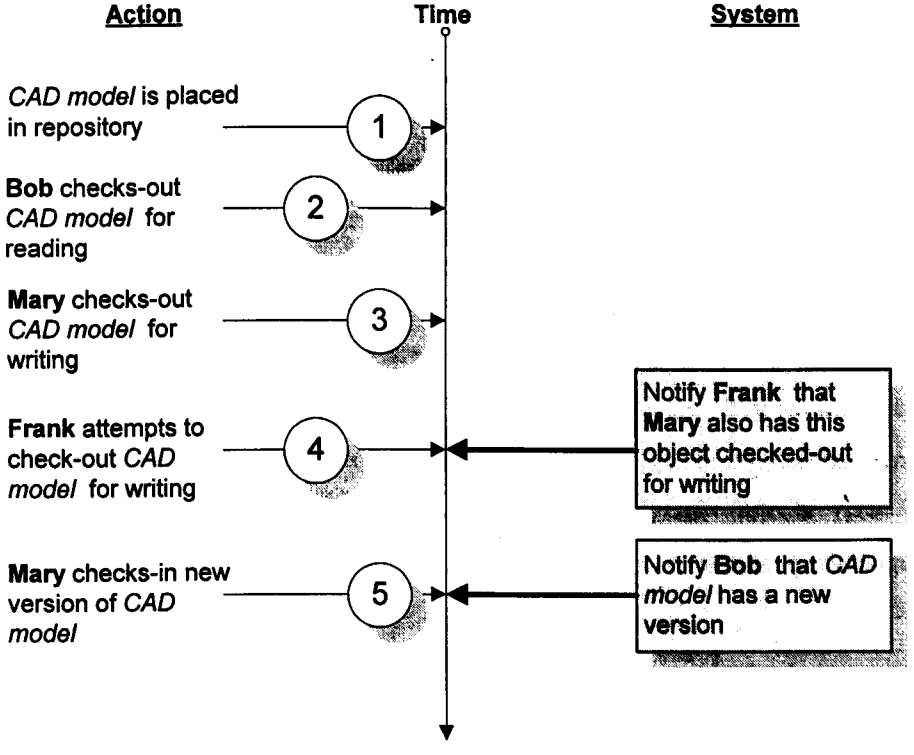


Figure 5-2. Concurrency control scenario.

At some later time, point 5 on the diagram, Mary finally checks-in her new version of *CAD model* at which point the system must notify Bob (who is still viewing *CAD model*) that there is a new version of the object he is viewing. This allows Bob to work with the most up-to-date information if so desired. If for example, Bob was performing a Finite Element analysis of *CAD model*, it might be preferable to abort his current effort and analyse the latest design instead. It can be seen then that the CONCERT architecture supports a semantic locking mechanism with lock notification to proactively support collaborative group working.

5.3.5 Project management and history

In CE projects it is extremely important to be able to learn from past projects. It is believed that by learning from, and assessing how to avoid certain types of problem, reductions in development time and cost can be achieved. This has been reinforced by the US DARPA Initiative in Concurrent Engineering (DICE) project, which advocates the capturing of corporate history (for a complete description of the DICE project the reader is referred to Cleetus and Usjio 1989). In addition to this, project management initiatives such as those required by quality standards such as the ISO 9000 certification process impose additional demands. ISO 9000 requires that project tasks undergo a two-stage sign-off procedure which includes the task being *checked* by an independent team member (i.e. not the person who performed the task) followed by the task being *released* again by an independent team member (i.e. not the person who performed the task or checked it).

With this in mind, the following statements define the project history policy of the CONCERT environment:

I. Once an object[13] has been submitted to the repository is should not be allowed to be deleted at any time during or after completion of the project.

II. Changes to existing objects should be non-destructive, i.e. they should not actually change the original information. This requires that a versioning mechanism be used to create new versions of objects based on original objects (see §5.3.7—Versioning). This versioning process is an ideal way of capturing a design history during a project.

[13] An object in this sense is any item that can be generated during the life cycle of a product and is capable of being stored in the CONCERT repository.

III. All tasks (and projects) should include a two-stage sign-off procedure in accordance with the ISO 9000 process defined above. This additional requirement makes each step of the project fully accountable (see non-repudiation in §5.3.6 below).

5.3.6 Security

Schneier (1997) declares that it is impossible to guarantee 100% security but it is possible to work towards a goal of 100% risk acceptance. In the field of competitive manufacturing, security is an extremely important issue and therefore any environment used to support this must aim to minimise risks to security infringements. This covers a number of aspects such as:

- *Data security*—Ensuring that a competitor cannot get access to product data either by 'eavesdropping[14]' or by premeditated attack. This also concerns legitimate employees erroneously changing information by accidental means or through some deliberate action.

- *System security*—Ensuring outside intruders cannot access the system. There may also be instances during product design when a manufacturing organisation might legitimately require collaboration with external consultants or service providers, for example the use of a service bureau to deliver Rapid Prototyping models of CAD data (Deitz 1995). Such a bureau would need to be given controlled, restricted access to data within a CONCERT environment to allow the retrieval of the necessary CAD data electronically, thereby supporting concurrent processing.

- *Non-repudiation*—Barrett and Tangney (1995) argue that a fundamental requirement for CSCW support is anonymous communication. Anonymity however, is a potential security risk that conflicts with the goal of non-repudiation, whereby each team member is made irrefutably responsible for his or her actions. This research proposes that all communication and transactions therefore be made publicly accountable.

This policy specifies the CONCERT architecture security model. This model has three tiers that provide security at distinct interface points to the system. These are:

1. *Network access*—This level concerns team members accessing the environment over a network. Security here is concerned with determining whether a particular user has

[14] In computer networks, digital eavesdropping can be accomplished relatively easily by physically connecting a suitable device to the network cabling and 'listening' to the data that is on the network.

permission to access the environment. This should be done by way of a username / password combination entry system.

2. *Data access*—This level exists to ensure that each user has the necessary permissions to perform an operation on a particular data object. This can be done using an access control list (ACL). Every object in the data repository has an associated ACL that specifies team members and privileges. These privileges determine whether a given team member can read or update an object.

3. *Data transmission*—This level ensures that data cannot be openly viewed whilst being transported over the network. This is achieved by applying a data encryption and decryption process.

The three-tier security model is shown in Figure 5-3.

The access control policy specifies which team members can access which objects. Certain special cases exist whereby objects within the repository or indeed the whole repository contents need to have their access blocked so that changes cannot be made. This may be necessary for example when a particular design object is undergoing a group review process and it is therefore undesirable for team members to be able to change this reference object. Another example may be the locking of the entire repository when it is being backed up for security purposes. The notions of 'freezing' an object (or the entire repository) to prevent its contents being changed and then the subsequent 'thawing' of the frozen component is therefore introduced.

The following statements define the access control policy.

I. Whenever a new object is submitted into the repository, it must have an associated Access Control List (ACL). The ACL specifies:
   A. Those team members who can read the object, and
   B. Those team members who can write new versions of the object[15].

II. When a team member wishes to access a given object, the system must first check that the team member has the necessary access permission to perform the requested operation (i.e. read or write).

III. Only team members designated as CONCERT System Administrators have the ability to freeze and thaw individual objects within the repository or the entire repository. System administrators are special classes of team member who have

[15] This also gives read access by default since a team member cannot change a given object unless they can read the actual object first.

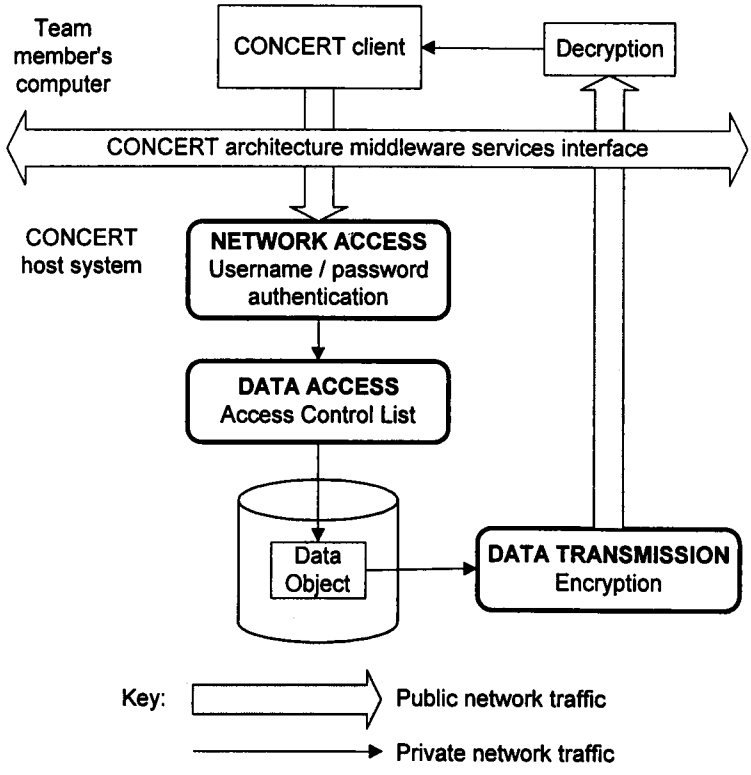


Figure 5-3. Three-tier security policy of the CONCERT architecture.

additional privileges that enable them to perform routine maintenance tasks within the CONCERT system.

5.3.7 Versioning

In product development, it is common for different versions of a design to coexist during a single life cycle. An example of this is a product line based on one single common part with several product variations, e.g. car, sports car, estate car, etc. Version control is a much used technique in the software development industry (Rochkind 1975; Tichy 1985) and similar techniques have successfully been applied in a concurrent engineering environment to capture changes to a product model with the use of delta-files (Hardwick et al. 1995).

The version control policy determines how and when that new versions of objects are created. The following statements define the version control policy.

I. To capture the design history of an object, previous versions are not deleted from the repository. Instead they are retained forming a version tree as shown in Figure 5-4. This shows how various revisions of a design have evolved over time.

II. A data object forms a parent-child relationship with new versions of itself. This means that when a new version of a data object is created it becomes a child of the

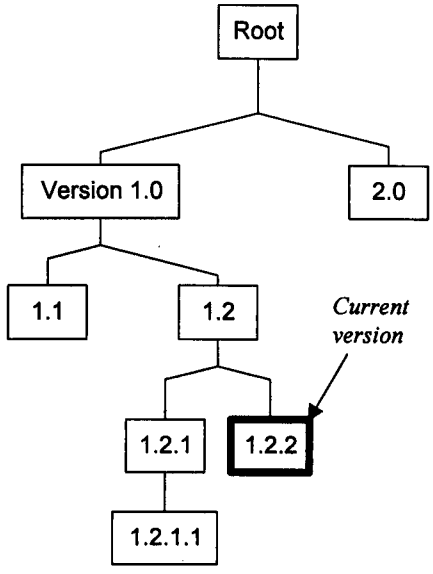


Figure 5-4. A data object version tree.

parent data object. This child version can itself be the parent of many children (making the original parent a grandparent and so on *ad infinitum*.) As an example of this, the data object denoted by the name "1.2" in Figure 5-4 has two children: "1.2.1" and "1.2.2"; and one grandchild: "1.2.1.1" (whose parent is "1.2.1"). In this case, version "1.2.1.1" is a more recent version of "1.2".

III. When a data object is checked-out, it can automatically be checked-in at the correct place in the version tree by locating the parent and adding the new data object as a new child.

IV. The *current version* of an object is defined as the object that is the primary focus for future development. All data object version trees have one current version. This is used to denote stable product definitions that have group consensus. Any object within a version hierarchy can be designated as the current version. If no such object is declared, this value should default to the highest-ranked child in the version tree, i.e. in Figure 5-4, if "1.2.2" was not specified as the current version, it would default to "1.2.2" anyway as this is the highest-ranked child.

When two or more team members create child versions of some object, it may be desirable to merge these separate versions into one common composite object which has all the changes from each version merged to create a new object (Mills et al. 1993). This merging process requires that conflicts in each version be reconciled to produce the composite object. For example in Figure 5-4, it may be desirable to merge version "1.2.1.1" and version "1.2.2" creating the new object at version "2.0". If there are conflicting changes in the two versions to be merged then this must be reconciled either manually by a process of

negotiation or automatically using intelligent knowledge-based rules, for example. The CONCERT architecture supports both manual and automatic merging mechanisms.

5.4 The CONCERT architecture

The CONCERT architecture has been design using object-oriented methods in an effort to improve the flexibility and extensibility of the architecture. CONCERT is a set of distributed middleware services that can be used to facilitate Concurrent Engineering (see Figure 5-5). A middleware service is a general-purpose service that sits between heterogeneous computer platforms and applications allowing multiple platforms to interact with a common application (Bernstein 1996). Each of these services is defined as a unique software object that has a publicly defined interface which describes the operations that the service can perform (see Appendix B for full details of these interfaces). The CONCERT architecture allows Internet-based software clients (operated by virtual CE team members) to access these object-oriented middleware services via a globally available Object Request Broker mechanism.

The CONCERT middleware support services comprises four core components: a *distribution support* component, a *collaboration support* component, a *project support* component and a *repository support* component. These four loosely coupled, globally distributed support services work together to provide the main functionality of the architecture. In addition to the four core support components, the data object repository is also located within this layer. The repository is a key component in a Concurrent Engineering support system. This is where all configuration items and data objects that are produced throughout the entire life cycle of the project are stored. It should be noted that the only link to the data object repository is via the repository support service interface. This precaution is used to provide a high level of protection to the data (a more detailed discussion of security considerations is given in §5.3.6).

The following sections briefly summarise the purpose of each of the four support services.

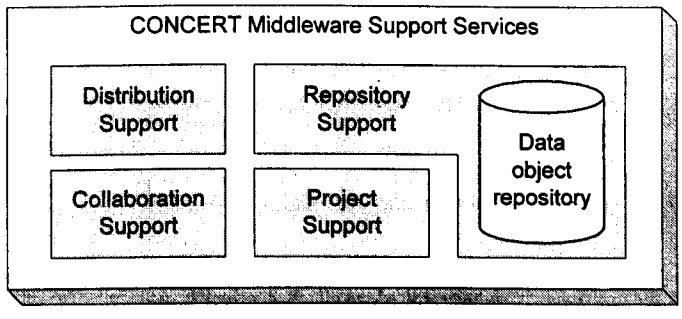


Figure 5-5. The CONCERT architecture.

A more detailed description of each of the services is given in Appendix B.

5.4.1 Distribution Support Service

The Distribution Support Service (DSS) is primarily concerned with providing facilities to support distributed access to the support services. Its responsibilities include:

- maintaining a list of current valid users and their contact details
- managing the authentication of users via a secure password list
- co-ordinating session management for users
- acting as an encryption key trustee for users

It is the DSS that handles and co-ordinates the login and logout of the support environment and it is also called upon by the other support services to authenticate users. A subset of the DSS system interface is shown in Figure 5-6 (for a summary of the Fusion notation used the reader is referred to Appendix A.)

5.4.2 Collaboration Support Service

The Collaboration Support Service (CSS) is primarily concerned with providing a means for

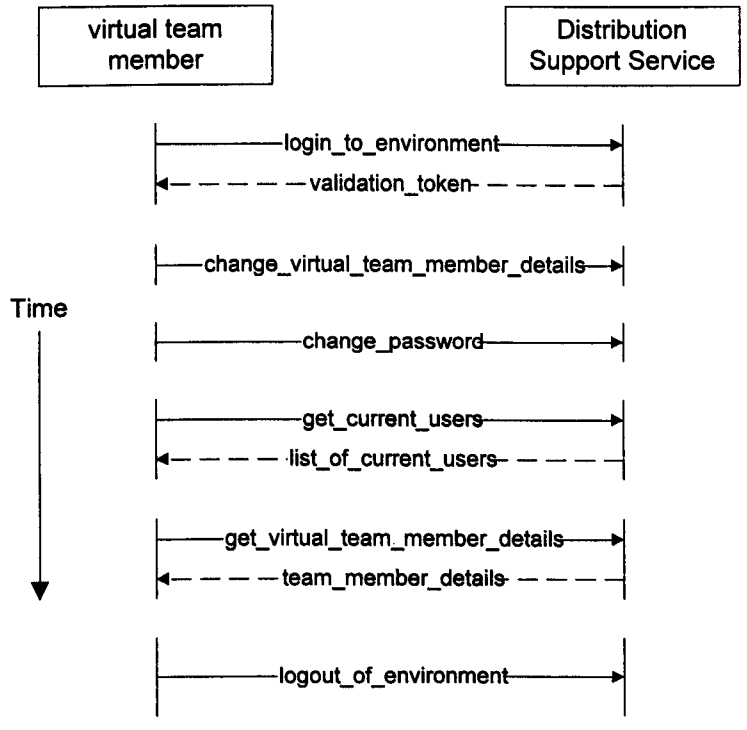


Figure 5-6. Sample interface for Distribution Support Service.

co-operative working and communication within the architecture. Its responsibilities include managing communication channels such as:

- text-based conferences (multi-user and one-to-one)
- electronic mail
- bulletin boards
- group consensus voting services to support group decision making
- message sending and broadcasting
- event sending and / or channelling

All communications between team members are handled by the CSS. This can call upon the DSS to either find a team member's workstation details (for sending synchronous messages direct to the members display) or e-mail address from their contact information stored by the DSS (for asynchronous message sending).

5.4.3 Project Support Service

The Project Support Service (PSS) is primarily concerned with providing a means for specifying, monitoring and controlling projects and tasks within the architecture. Its responsibilities include:

- the management and administration of project plans
- monitoring of the allocation and sign-off of tasks and sub-tasks
- project management reporting
- the management of past project histories

The PSS undertakes these roles in accordance with ISO 9000 quality requirements.

5.4.4 Repository Support Service

The Repository Support Service (RSS) is primarily concerned with providing data storage and retrieval within the architecture. Its responsibilities include:

- the storage and retrieval of data objects in the repository
- managing version control of data objects
- managing concurrency control of the data

- providing a means for searching the repository
- providing a means for translation of data formats
- the management and validation of access control lists (ACLs)

The RSS looks after persistent, long-term storage of data and therefore requires additional system administration facilities such as the back up and restoration of the data and locking of the repository contents while this is taking place.

5.4.5 Data object repository

The CONCERT architecture specifies that the repository must be capable of storing all of the various types of data that can be produced during the product life cycle. Object databases are the most likely candidates for this task since they can readily cope with complex-structured data such as video, audio and proprietary binary information. In addition, object databases are more extensible in coping with new user defined data formats which is a desirable property of the architecture.

5.5 The CONCERT environment

The CONCERT architecture provides a number of services to the outside world. The *CONCERT environment*[16] is the combination of this architecture with applications that utilise these services to provide functionality to users. To achieve this, an *application layer* is implemented to interface with the CONCERT architecture and which provides a platform for user applications. It is at the application layer that user software tools are implemented and legacy components (i.e. applications and databases) interact with the system (see Figure 5-7). This section discusses the role of the application layer in more detail.

5.5.1 The application layer

The application layer sits on top of an Object Request Broker that is used to provide a global access to the middleware support services. It is at the application layer that the high-level functionality of the environment is implemented (as opposed to the low-level functionality of the support services.) Figure 5-7 shows four different types of component in the application layer interacting with a CONCERT system. These include the *CONCERT workbench application*, *third-party applications*, *legacy applications* and *legacy databases*. These are described in more detail below.

[16] The term *CONCERT architecture* is used to denote the middleware services shown in Figure 5-5 and the term *CONCERT environment* to denote the combination of the CONCERT architecture, Object Request Broker and application layer as shown in Figure 5-7.

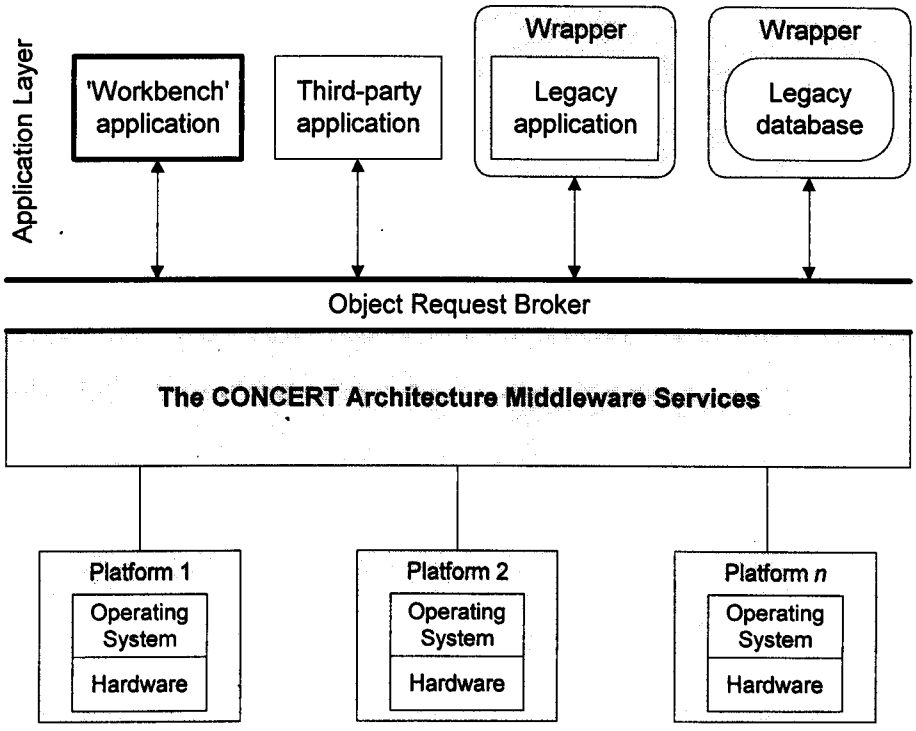


Figure 5-7. The CONCERT environment.

a) *The CONCERT workbench application*—The workbench application provides a graphical front-end to the user so that they can access the services of the environment. An example of a tool within the workbench is the project progress-reporting tool. This uses the functionality of the project support service to provide the information and deliver the information to the user in a graphical tree format. Another example is the group text conferencing tool that uses the low-level 'publish-and-subscribe' functionality of the collaboration support service to provide a user-friendly graphical interface to participants.

b) *Third-party applications*—These are applications by external software vendors that conform to the CONCERT architecture specifications. An example of this might be a CAD software company who have written a bespoke CAD package that can access the CONCERT repository directly by using the repository support service instead of using local disk storage.

c) *Legacy applications* —Existing legacy applications and databases can continue to be used by 'wrapping' them in a specially written software layer which intercepts and re-routes commands to the CONCERT support services and vice-versa. For example, the Pro/ENGINEER CAD system (Parametric Technology Corporation 1993), provides application developers with an Application Programming Interface called Pro/DEVELOP which allows external applications to execute operations on Pro/ENGINEER's data and monitor user functions such as 'saving' and 'loading' of

designs. This wrapping technique has been used successfully in similar engineering environments (for example Goldstein 1994; Norrie and Wunderli 1996) and is widely used in the software engineering community to incorporate legacy systems into new object-oriented systems (Garnett 1997).

d) *Legacy databases*—This is essentially the same as legacy applications except that it covers the wrapping of existing database applications. These wrappers could enable existing database applications to redirect requests to store and retrieve information from some local file system to the CONCERT repository instead.

5.6 Objectives revisited

A number of objectives for the CONCERT architecture were stated in §5.2. These were: *Wide availability, Scalability, Extensibility, Generality and Architecture neutrality.* These are now revisited showing how the CONCERT architecture and environment address these objectives.

Wide availability—The modular structure of the CONCERT architecture means that it can be easily integrated into existing engineering systems. In addition, the concept of legacy application and database wrappers means that existing technology can be readily integrated to the CONCERT environment.

Scalability—Scalability within the architecture comes from the loose coupling of the four support services and data object repository. This allows these components to be moved (or migrated) without affecting the system as a whole. As an example of this, it is possible for say, a constraint management service to be added without affecting the existing services. This new service could use the functionality of the existing services but the existing services will be unable to use the constraint service since they could not possibly know its interface and how it could be applied. However, by sub-classing existing services, they could be re-engineered to use the new service interface, thus extending the life span of the architecture.

Extensibility—Extensibility within the architecture is evident in two dimensions: horizontal and vertical extension. Horizontal extension is made possible via the creation of additional support services or sub-classing existing support services. This is a result of each of the four support services having a publicly defined interface. It is entirely feasible therefore, for developers to use an alternate implementation for a given service without affecting the other services in the architecture[17]. An example of this might be the replacement of one database

[17] The only proviso being that the new service provides at least the same functionality of the one it replaces, i.e. supports at least the same interface.

technology with another to enable faster processing or increased flexibility. Of course the new service can also extend the functionality of the existing service to include new facilities. For example, it is beyond the scope of this research to provide an implementation of videoconferencing within the *Collaboration Support Service* (CSS) but this could be developed by a third-party who could then subclass the CSS to include support for videoconferencing. This technique is also known as 'black-box reuse' in the software engineering community.

Vertical extension of the environment is also possible via the development of third-party applications within the application layer that make use of the support services' interfaces and classes in order to address any new high level viewpoints that may arise. For example, it would be possible to provide a shared whiteboard tool using the existing low-level publish and subscribe facilities within the Collaboration Support Service.

Generality—In order to address this objective, existing network protocols and application platforms must be employed. The CONCERT environment, in using the standard TCP/IP protocol and by employing the Java Virtual Machine ensures that the environment will have wide generality.

Architecture neutrality—Heterogeneous platforms require some means of architecture neutrality. In the CONCERT architecture, this has been addressed by using an Object Request Broker to allow multiple platforms to access the common core services. In addition to this, the notion of language homogeneity through the use of the Java language further supports this objective.

5.7 Summary

This chapter has introduced a computer system architecture (CONCERT) that can be used to help facilitate virtual Concurrent Engineering teams. The design of this architecture is based on objectives such as wide availability, scalability, extensibility, generality and architecture neutrality. A number of guidelines and policies for developing support environments have also been defined that can be used by other software developers wishing to build support environments.

The CONCERT architecture is based on four core support services: distribution support, collaboration support, project support and repository support (which is supported by a data object repository). These support services have been designed using the principles of object-orientation and are incorporated as Internet-based middleware objects to enable wide availability, scalability and architecture neutrality. These support services are highly

extensible through the techniques of sub-classing and aggregation, commonly termed as 're-use' in the software engineering community. An environment based on CONCERT is also described which serves to show how the architecture fits into a typical real-world application.

The following chapter describes the development of an information system object model that can be used to support the operation or the CONCERT architecture or any other such architecture designed to support CE.

Chapter 6

6. A information system object model to support virtual CE teams

6.1 Introduction

This chapter describes an object-oriented information system model devised as a result of this research to support virtual concurrent engineering teams. This information model is derived from the CE-RM presented in Chapter 4 and supports the computer systems architecture developed in the previous Chapter. The relationship between the object model and both the CE-RM and CONCERT architecture is once again shown in Figure 6-1.

The object model is presented in this chapter using plain English text for the purpose of providing an overview of its component parts. A complete graphical object model using the Fusion notation is given in Appendix C which shows the associations and relationships between the objects described in this chapter (see Appendix A for a summary of Fusion notation).

This chapter also describes how the object model provides support for both the middleware components of the CONCERT architecture and the viewpoints identified in the CE-RM. Finally, data storage implications for the object model are summarised and a solution based on object databases is discussed.

6.2 Information system object model

The information system model presented in this Chapter is an object-oriented model. Before describing the model, the motivations for choosing an object-oriented methodology are given. This section then goes on to discuss how the proposed information model relates to both the CE-RM in Chapter 4 and the CONCERT architecture in Chapter 5. A brief discussion on the implications for actual physical storage of the information model completes this section.

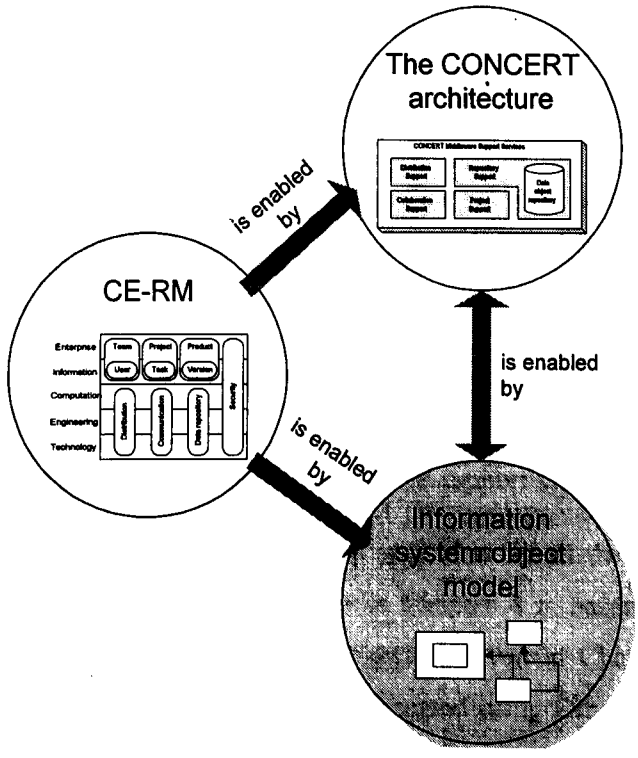


Figure 6-1. Object model overview.

6.2.1 Motivation for using object-orientation

The prime motivation for using object-orientation (OO) for developing the CONCERT information system model stems from the following factors:

- *Easier model to understand*—Sommerville (1996) highlights the fact that for some systems there is a clear correspondence between entities in the real world and their respective objects in the system which serves to improve the comprehension of the design.
- *Ease with which changes can be applied*—Objects allow changes to be localised, i.e. happen at the individual object level as opposed to the system level.
- *Greater flexibility*—Evolution of object models can be more easily achieved using the techniques of inheritance and encapsulation to produce new objects based on existing objects with only the differences between the components needing to be specified.
- *Object reuse*—The ability to reuse object specifications within a project (and even from past or similar projects) brings productivity improvements for the system designer resulting in models being built in a shorter time-scale.

- *A consistent model*—The same model can be used from analysis through to implementation. Gray (pg. 6, 1994) reiterates this by describing OO as being a "...more coherent development model."

- *Object model can simplify implementation*—For practitioners of object-orientation, implementation of object models can be much simplified by the use of object-oriented programming languages such as C++, Smalltalk and Java.

- *Objects may be physically distributed and may execute in parallel* (Sommerville 1996)—These characteristics were highlighted in the analysis given in Chapter 4 as being of great importance in the building of CE support systems.

The core underlying theme of object-orientation is evident in all the phases of the work proposed in this thesis: the viewpoint analysis in Chapter 4 is based on object-oriented analysis techniques; the computer systems architecture given in Chapter 5 is built from object-based services which are distributed and accessed using Object Request Brokering facilities; the information model proposed in this Chapter is also object-oriented and, as will be seen in the following chapter, the actual prototype implementation of the proposed system is done using an object-oriented programming language. This consistency of model throughout these stages has served to provide a clearer and swifter understanding at each stage transition, i.e. from analysis to design and from design to implementation.

6.2.2 Base object types

In the following description and for the remainder of this Chapter, the names of classes within the information system object model will be distinguished by the use of a different typeface, e.g. Virtual Team Member. A summary of the base object types in the CONCERT information system model is given in Table 6-1.

Table 6-1. Summary of CONCERT object classes.

Class name	Super-type	Description
Access Control List		Specifies which users can access a given repository component and in which mode, i.e. read or write. This is a group of access permissions.
Access Permission		Holds the details of an access permission for a given user. (See also Access Control List)
Annotation		An additional reference item that can be used to support a point of view or point to other sources of information. This is an abstract class. Annotations are catered for in messaging objects and repository components.
Bulletin Board		A place where informative messages can be posted for a given period of time. A bulletin board is an asynchronous communication tool. (See also Bulletin Board Message)
Bulletin Board Message	Message	A generalised type of message that can expire after a set date. (See also Bulletin Board)
CE Project Configuration		The main class used to hold all configuration items within a given CE project. This is used to store the Project Plan, Team, Bulletin Board, Repository and any Conferences or Problem Statement details.
Conference		A place where CONCERT users can communicate synchronously by sending messages. (See also Message)
Data Workspace	Repository Component	A storage folder where repository components can be held. These are a means for grouping like objects together.
E-mail Message	Message	Used for creating and sending electronic mail messages to CONCERT users or external users or systems.
Message		Describes any sort of message that can be sent between CONCERT users.
Opinion		An opinion on some problem statement by a particular user. Can be one of: in favour, against or abstain. (See also Problem Statement)

Table 6-1 continued

Class name	Super-type	Description
Outcome		The result of a group voting session on a given problem. (See also Opinion, Problem Statement)
Problem Statement		A problem expressed as a statement on which other team members can register their vote.
Project Plan		Describes the project plan, i.e. it is made up of a number of tasks and sub-tasks. (See also Task)
Repository		The holder for all Repository Components in a given CE Project Configuration.
Repository Component		An abstract class used to define components that can be stored in the CONCERT repository.
Repository Object	Repository Component	A storage class for any configuration item produced during a project, i.e. a physical file or data item.
System Administrator	Virtual Team Member	A special class of Virtual Team Member who can perform additional system maintenance operations.
Task		Describes a specific task during a given project. Can be composed of a number of sub-tasks. (See also Project Plan)
Team		The grouping of virtual team members into a known team. (See also Virtual Team Member)
Text Annotation	Annotation	A text-based annotation.
URL Annotation	Annotation	An annotation based on a Uniform Resource Locator (URL) on the World-Wide Web.
Validation Token		Issued to a user only when they have successfully logged into the CONCERT environment. This token is then used in all service requests to validate that user.
Virtual Team Member		The class that defines individual team members. Holds contact information that can be used by the CONCERT environment and other users.
Workstation		Holds attributes on the actual computer workstation used by a particular CONCERT user during a session.

A sample object diagram for the Repository object is given in Figure 6-2. The model has been produced using the Fusion object modelling notation (Coleman et al. 1994) which was developed by Hewlett-Packard. The Fusion method is the result of an amalgamation of best practice from a number of popular object modelling notations was chosen for its clarity in presenting object models and the ease by which practitioners of other object modelling notations can interpret it. The notation shows aggregation by way of enclosing classes within the aggregate object (for example the *Repository Component* class is an aggregate of both an *Access Control List* class and the *Annotation* class). Relationships between classes are depicted by a diamond (e.g. the *is contained in* relationship between the Repository Component and Data Workspace classes). Inheritance is shown in Fusion models by connecting sub-classes to the super-class using a triangle (e.g. both the Data Workspace and Repository Object classes are sub-classes of Repository Component). For the complete summary of the Fusion notation the reader is referred to Appendix A while the complete information system object model is given in Appendix C.

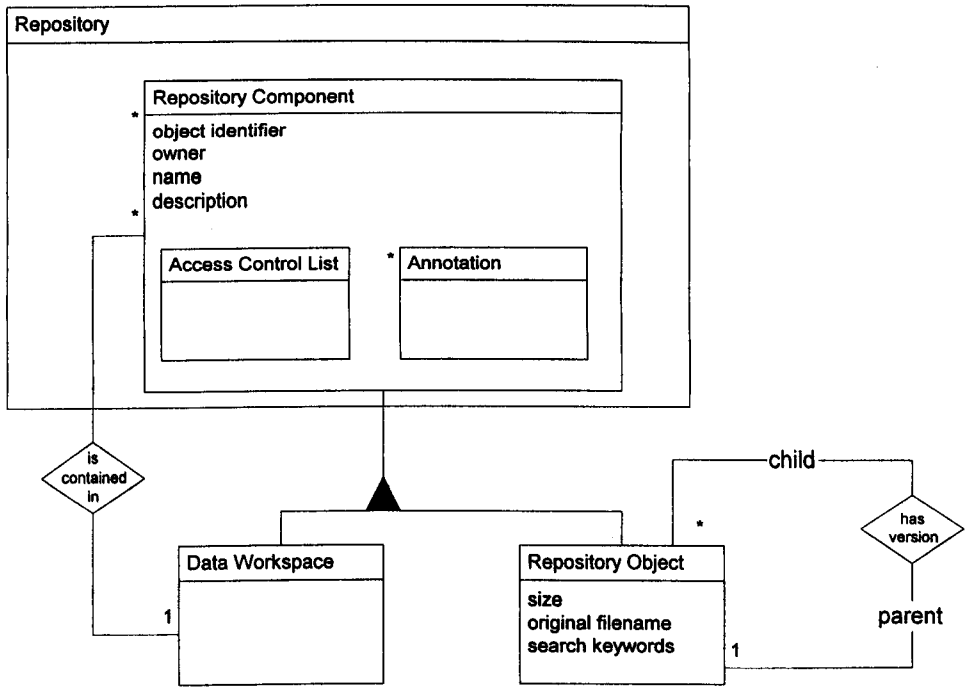


Figure 6-2. Repository object diagram.

6.2.3 Supporting the CE-RM through the information model

This section highlights the relation between the object types in the information model presented in this Chapter and the object types identified in the CE-RM presented in Chapter 4. The information system object model primarily supports the Enterprise and Information viewpoint classes of the CE-RM but, as will be seen in §6.2.4, also provides support for the Computation, Engineering and Technology viewpoint classes.

In the CE-RM, the Enterprise and Information viewpoint classes are further partitioned into the classes: *Security*, *Team* and *User*, *Product* and *Version* and *Project* and *Task*. The relation between each of these and the information system object model is described below:

- *Security*—The security viewpoint is addressed firstly by the Access Control List (and Access Permission) classes that serve to control access to data by authorised members only. Secondly, the Validation Token class ensures that only properly authenticated users can actually perform operations within the system.

- *Team and User*—The classes Team and Virtual Team Member map directly to the viewpoint classes *team* and *user*. A Team object is composed of one or more Virtual Team Members. These classes capture the information requirements of the *team* and *user* viewpoints, e.g. the Virtual Team Member class holds information such as name, address, e-mail address, telephone and fax number and a graphic image.

- *Product and Version*—The CONCERT system Repository, which holds all items produced during a CE project encapsulates the *product* and *version* viewpoint objects. In the information system object model, the Repository Component and its sub-classes Data Workspace and Repository Object capture the information requirements of the Repository. Repository Objects can have child versions that are, themselves, Repository Objects. Data Workspaces can contain any number of Repository Components, i.e. other Data Workspaces or Repository Objects. A product may undergo many revisions (i.e. have new versions created) before it is finalised and this data structure captures this ability. Data Workspaces are a means of grouping like objects together, e.g. in the design of a motor car, a data workspace might be created to hold all designs for a particular component such as the dashboard. Within this data workspace, there might be another, lower-level data workspace for holding designs associated with a particular gauge or dial within that dashboard.

- *Project and Task*—The Project Plan and Task classes map directly to the CE-RM project and task viewpoint objects. The Task class is a recursive data structure which is capable of containing other Tasks and so on *ad infinitum* while the Project Plan class holds one or more Tasks. These classes hold information such as key dates (i.e. checking and release dates) and who carried out the task. Through the incorporation of ISO 9001 quality requirements within these classes they can be used to track and plan projects and tasks and provide full audit capabilities.

It can be seen that these classes address the Enterprise and Information viewpoints of the CE-RM but as will be expanded in the next section, some of these classes also provide support for the lower level viewpoints of Computation, Engineering and Technology.

6.2.4 Supporting the CONCERT architecture through the information model

This section highlights the relation between the object types in the information model presented in this Chapter and the object-based support services identified in the CONCERT architecture presented in Chapter 5. The information system object model provides information support for all four CONCERT support services and provides the means for data persistence in the data repository. The information system object classes that are pertinent to each of the four CONCERT support services are described in more detail below:

- *Distribution support service*—To support the notion that users will access the CONCERT environment from distributed computer workstations, the Workstation class holds information regarding that user's current settings, e.g. the physical location of the workstation and its logical position within some network, i.e. its Internet Protocol (IP) address. Another important class is the Validation Token which is used for security and authentication purposes and whose instances are only ever issued by this service. Finally, the Virtual Team Member forms the basis of user authentication and serves as a reference point for user contact information.

- *Collaboration support service*—This service is concerned with providing means for synchronous and asynchronous communication between virtual team members. This communication takes the form of message sending. The Message class and its sub-class Bulletin Board Message are used in the Conference and Bulletin Board classes respectively to allow virtual team members to send and receive synchronous and asynchronous text messages. In addition to this, the E-mail Message class also allows team members to communicate asynchronously with other internal users or external people.

  Group working assistance is also provided by this service. Three classes provide information support for a group voting mechanism: the Problem Statement, which is used to provide a means for generating group discussion; the Opinion, which allows virtual team members to state their own opinion; and the Outcome, which provides an actual record of the group voting process.

- *Project support service*—The CE Project Configuration is the central focus for a given CE development project. This class is managed by this service and utilised by the other support services as a reference point to the objects contained within. This

service primarily provides the means for project management and tracking via the Project Plan and Task classes that provide the information support for this role.

- *Data repository support service*—The Repository is the software equivalent of the physical storage facility within the support environment. This service requires a suitable data structure for holding all the data that is produced during a CE project and ensuring that access is securely administered. The Repository Component class and its sub-classes Data Workspace and Repository Object model the information requirements of this data structure.

  Incorporating the Access Control List class within the Repository Component class intrinsically caters for security. Because the Access Control List is a component of the super-class, Repository Component, it means that it is applicable to both the Data Workspace class and the Repository Object class. This means we can restrict which team members can actually read or create new versions of specific Repository Objects and also specify which team members can retrieve items from or add items to a particular Data Workspace.

It can be seen that these classes provide information support to the CONCERT support services which, in turn, enables these to support the Computation, Engineering and Technology viewpoints of the CE-RM.

6.2.5 Implications of object model on information storage

The information model proposed above requires persistent storage—a Repository Object produced by a Virtual Team Member needs to be retained for later sessions so that other Virtual Team Members can access it, i.e. the lifetime of the individual objects is longer than the lifetime of the application used to create them. A number of candidate database technologies are available for this persistent storage task. These include:

- relational databases,
- extended or object-relational databases, and
- object oriented databases.

To store object-oriented models in relational databases, the objects must first be flattened down into relational tables (see for example Ch.5 in Loomis 1995) i.e. by mapping the object classes into tables. This mapping process causes a loss of semantic information in the classes and can require many more lines of programming code to implement which invariably leads to greater possibility for errors. A performance overhead is also incurred while this mapping and re-mapping takes place (Cattell 1994).

Extended or object-relational databases include features that are akin to object-orientation, such as using the term *class* instead of *table* and allowing tables to inherit features of other tables. However, they are still built on relational database technology and therefore suffer from many of the same problems as those highlighted in the previous paragraph. Object-oriented databases, on the other hand, can store objects and their relationships and associations directly without the need to flatten the individual components. Complex associations between objects such as that given between the Repository and Repository Component (and its sub-types Data Workspace and Repository Object) which involve recursion and inter-linking can be stored without change in object-oriented databases whereas the mapping of these classes into relational tables will incur additional complications such as those described above. Performance in retrieving this information is also improved in object-oriented databases which is an important issue in competitive manufacturing where information needs to be available in a timely manner.

6.3 Summary

The object model developed in this chapter is a self-contained and complete information system model for use by developers of computer-based CE support systems. This chapter has also shown how this information model supports and can be integrated into one such support system: the subject of this thesis. It does this by discussing the links between both the CE-RM (Concurrent Engineering Reference Model) presented in Chapter 4 and the CONCERT architecture presented in Chapter 5 with the information system object model.

In keeping with the underlying trend in this thesis, the information model described in this chapter has been developed using an object-oriented methodology. This is consistent with models presented in previous chapters in this thesis and will also serve as the means for the implementation discussed in the following chapter. For the reasons highlighted in section 6.2.5, it is considered expedient to use an object-oriented database management system (ODBMS) as the data storage system for the information model proposed in this chapter. Using an ODBMS is in keeping with the object-oriented nature of the entire design thereby ensuring consistency throughout the entire project.

The next chapter describes how the models proposed in this and the preceding two chapters have been implemented into a complete computer-based support environment for virtual concurrent engineering teams.

Chapter 7

7. System implementation

7.1 Introduction

This chapter gives an account of an implementation of the CONCERT architecture described in Chapter 5 and the information system object model described in Chapter 6. This implementation will be referred to hereinafter as the CONCERT environment. Recall the definition of environment given in Chapter 1:

> *"An environment is the realisation of an architecture and supporting systems. In the case of realising a computer systems architecture, the environment would be the combination of software and hardware required to implement the architecture."*

This chapter begins by describing the goals of the CONCERT environment in relation to the goals of distributed computing environments. The CONCERT environment is an example of a distributed system and therefore needs to address the same objectives. The individual components of the CONCERT environment are described in detail along with details of an actual prototype test-bed that was devised in order to evaluate the system.

7.2 Distributed system goals

The CONCERT environment is a distributed computer system. A consequence of this is that it must support certain transparency characteristics in order to provide a level of service suitable for use within a distributed environment. Chapter 2 introduced this concept by describing ten types of transparency, namely *access*, *concurrency*, *failure*, *federation*, *location*, *migration*, *performance*, *replication*, *resource* and *scaling* transparencies. Throughout this chapter, these concepts will be highlighted again to show how the individual components of the CONCERT environment meet these goals.

7.3 Environment components

The environment consists of the following software components[18]:

- A globally available Object Request Broker (ORB) (hereinafter referred to as the 'CONCERT ORB').
- The four distinct CONCERT support services (i.e. *Distribution*, *Collaboration*, *Project* and *Repository* support services).
- A client application (hereinafter referred to as the 'workbench') used to access the services and provide high-level tools to users of the environment.

These components, when taken as a whole, constitute the CONCERT environment. Each of these components is described in greater detail in the following sections.

7.3.1 CONCERT ORB

The CONCERT ORB performs the role of registering the location of the support services and publicising this to client applications, i.e. it performs a task similar to the *Naming Service* in the Object Management Group's CORBA Common Services specification (Object Management Group 1997). There are differences however, which relate to the security of the overall process, which are discussed in more detail in §7.3.1.1. The CONCERT ORB requests such as *bind* (which associates a service with a given name), *unbind* (which removes any association) from server processes and *lookup* (which is used to locate a given service using a simple name) from client applications. The interface for the CONCERT ORB is shown in Figure 7-1.

The main role of the CONCERT ORB is to act as a service database for clients and servers within the environment. In order to fulfil this role, it also has to perform certain 'housekeeping' chores such as periodically checking that all registered services are still responding (i.e. the processes are still alive) in an effort to improve *failure transparency* characteristics. Any 'dead' service will have its entry removed from the CONCERT ORB database so those clients requesting that service can be better informed of its current state. The CONCERT ORB also persistently stores its own internal state (i.e. its service database) to disk so that, in the event of a system failure, it can attempt to gracefully recover thereby improving overall *failure transparency* within the environment.

[18] These components were first introduced in Chapter 5 (see Figure 5-7 on page 81).

- **public BindKey bind(String serviceName, ObjectRef service) throws AlreadyBoundException**

  Binds a service with a given name into the CONCERT ORB service database.

  Parameters:
  serviceName - the name of the service
  service – a reference to the object which is the service
  Returns: a unique bind key
  Throws: AlreadyBoundException if the service name is already bound

- **public void unbind(String serviceName, BindKey bindKey) throws NotBoundException, AccessException**

  Unbinds a service with a given name from the CONCERT ORB service database.

  Parameters:
  serviceName - the name of the service
  bindKey - the unique key issued at bind time
  Throws: NotBoundException if the service name is not bound
  Throws: AccessException if the bind key is invalid

- **public ObjectRef lookup(String serviceName) throws NotBoundException**

  Looks up a service with a given name from the CONCERT ORB service database.

  Parameters:
  serviceName - the name of the service
  Returns: a remote object reference to the requested service
  Throws: NotBoundException if the service name is not bound

Figure 7-1. Interface definition for the CONCERT ORB.

If a given service needs to move (migrate) to a new machine, it can do so in a process that involves unbinding itself from the ORB, moving to the new machine, restarting the server and then rebinding. When client applications contact the CONCERT ORB they will be informed of the new location automatically. This enables the CONCERT environment to support both *migration* and *location transparency*.

The diagram in Figure 7-2 shows the four steps that are needed for the complete environment to function correctly. These steps are described as follows:

1) Each support service binds itself in the CONCERT ORB. The ORB allows services to bind from any location even across company boundaries. In this way the ORB supports the notion of *federation transparency*.
2) Client applications then look-up the location of a support service in the CONCERT ORB at run-time. This provides support for *location transparency* to applications and users.
3) The CONCERT ORB returns a remote object reference to the actual support service.

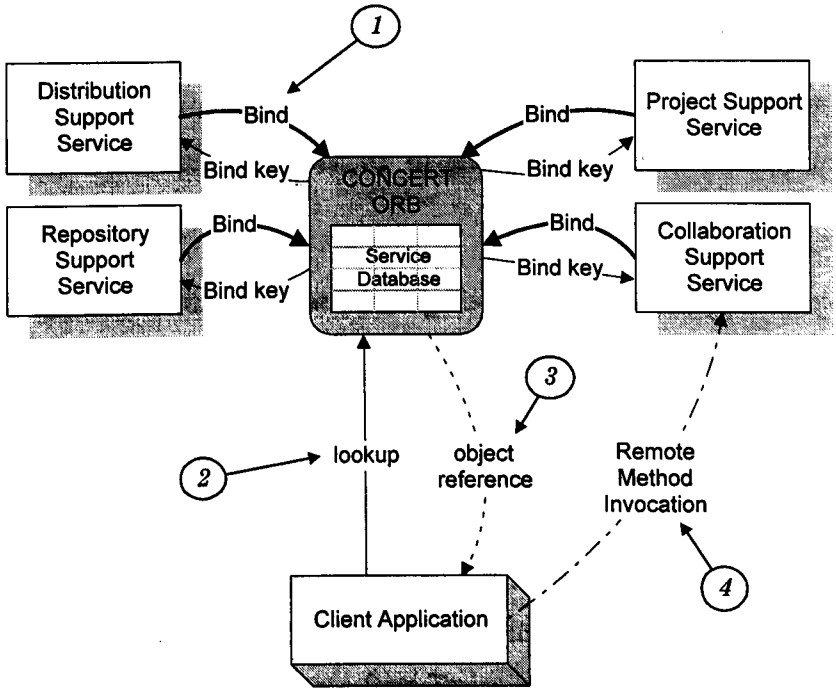


Figure 7-2. Achieving location and migration transparency within the CONCERT environment.

4) The client application calls methods in the remote service directly using the remote object reference. This is an example of *access transparency*—the user is unaware of how the client application accesses the actual service.

One aspect of transparency that is not specified in Chapter 2 and one which I believe is becoming increasingly important in distributed system building is *security transparency*. For the purposes of this thesis, the consequence of security transparency is defined as: "participants are unaware that communications and transactions take place securely." The following section describes some consequences of security transparency in more detail.

7.3.1.1 Security considerations

A common way of implementing computer viruses is via a 'Trojan Horse' application[19]. This is where a rogue software application fools the user into thinking it is some other legitimate application perhaps by using the same name. To ensure that services are not unbound and replaced by rogue processes masquerading as legitimate services within the CONCERT environment, each service that is bound in the CONCERT ORB is issued with a unique 'bind key'. This bind key is a secret code that is sufficiently difficult to forge in

[19] Trojan Horse: A program that either pretends to have, or is described as having a (beneficial) set of features but which, either instead or in addition, contains a damaging payload. (Source: McAfee Associates: http://www.mcafee.com/)

order to deter intruders. A service can only be unbound with the correct bind key, i.e. by the service itself or some trusted third party that has been entrusted with the bind key. This feature, in addition to the Validation Token that is issued to users who successfully login to the CONCERT environment (*see* Chapter 6) provide support for *security transparency* within the environment.

One interesting point worth raising at this stage is that CORBA-compliant Object Request Brokers allow services to *rebind* themselves in a single step, e.g. by calling a method such as:

```
ORB.rebind(ServiceName, Service);
```

This allows a new service to take over the role of the existing one or it can be used to simplify the migration process when a running service is moved to a new machine. This approach, however, lends itself to abuse from Trojan services. In addition to this, rebinding in this way contravenes the security constraints imposed by the CONCERT ORB since a service can only be unbound if its BindKey is known (which a new service could not possibly, or at least should not, know). A full description of the CONCERT ORB secure rebinding process is given in §7.5.1.

7.3.2 Support services

Each support service is defined as a remotely accessible software object with a publicly defined interface. Client applications can access the functionality of each service by requesting operations via this interface. These interfaces were discussed in Chapter 5 and are prescribed in Appendix B (Support service system interface models). The Repository Support Service also maintains its own object database that has been custom-built in Java.

In order to address *failure transparency*, the support services are each capable of recovering gracefully from system shutdowns or crashes. They do this by persistently storing the internal state relevant to that service on permanent storage, i.e. disk. If a service should fail, it can restart and restore its previous state from disk and resume where it left off before it failed. If one support service should fail, the other services within the environment will not be affected[20].

Each support service (and the CONCERT ORB) is also capable of responding to multiple requests simultaneously through the use of synchronisation techniques such as semaphores

[20] This is unless, of course, another service requests an operation of the failed service.

and monitors (Hoare 1978). In this way, it can be seen that the environment supports *concurrency transparency*.

7.3.3 Workbench application

The workbench application is a client application that is used by virtual team members to access the functionality of the support services and environment. The workbench employs a graphical user interface (GUI) to deliver high-level functionality to users of the environment (i.e. both CONCERT system administrators and general users). Examples of this high-level functionality include:

- A text conferencing application that makes use of the low-level publish-and-subscribe functionality of the Collaboration Support Service.

- A file upload application that can be used to deposit items into the CONCERT repository and which makes use of the Repository Support Service low-level interface.

In both of these examples the low-level interfaces of the CONCERT support services (described in Appendix B) are hidden from the user who instead sees a familiar GUI-based work environment. In order to utilise any of the CONCERT support services, the workbench first contacts the CONCERT ORB to determine the location of the actual services thereby maintaining *location transparency* between the user and the application.

7.4 Software components

Each of the software components described above has been implemented in the Java language (Arnold and Gosling 1996) using the mechanisms of Remote Method Invocation (RMI) within the Java language to provide support for distribution. RMI provides a method for request brokering between Java components by marshalling data and requests into a form suitable for transport across a TCP/IP network. Java was also chosen for its cross-platform capabilities in order to provide a consistent user interface on multiple platforms.

Figure 7-3 shows a section of a Java class developed to display the contents of a text conference in a window. This code illustrates how the CONCERT ORB is used to look up the current location of the Collaboration Support Service, which is later used to finish the conference.

```
public class ConferenceWindow extends AppWindow implements ActionListener {
  public ConferenceWindow(CONCERT_ORB ORB, ValidationCouple couple, long confID) {
    super("Conference");
    try {
      css = (CollaborationSS)ORB.lookup("CollaborationSupportService");
      ConferenceMessage msgs[] = css.openConference(couple, confID);
    } catch (Exception ex) {
      InfoDialog m = new InfoDialog(this, "An error has occurred",
        "Exit", true, "Please report the following error:\n"
        + "'" + ex.getMessage() + "' to your CONCERT administrator.");
      m.setVisible(true);
      dispose();
//
// user interface code not shown for clarity
//
  }

  public void actionPerformed(ActionEvent e) {
    if (e.getSource() == finish) {
    try {
      css.finishConference(couple, confID);
    } catch (NotAuthorisedException ex) {
      InfoDialog d = new InfoDialog(this,
        "Not Authorised", "OK", true,
        "You are not authorised to finish this conference."
        + "\nOnly the conference owner can finish a conference.");
      d.setVisible(true);
  }
}
```

Figure 7-3. Sample Java code for accessing support service.

Since the CONCERT support services share common characteristics, it is a logical step to make these services sub-classes of some common base server object type. This makes development of new support services an easier process since there is no wasted effort re-coding standard methods. This has been done in this implementation of the CONCERT environment. The class hierarchy can be seen using the Fusion notation in Figure 7-4.

The entire CONCERT environment software comprises around 150 Java classes, two fifths of which is responsible for the portable GUI components. This totals around 30 thousand lines of Java code (30 KLOC) but due to the compact nature of Java executable bytecode, the complete run-time executable code for the environment is small enough to fit on a single

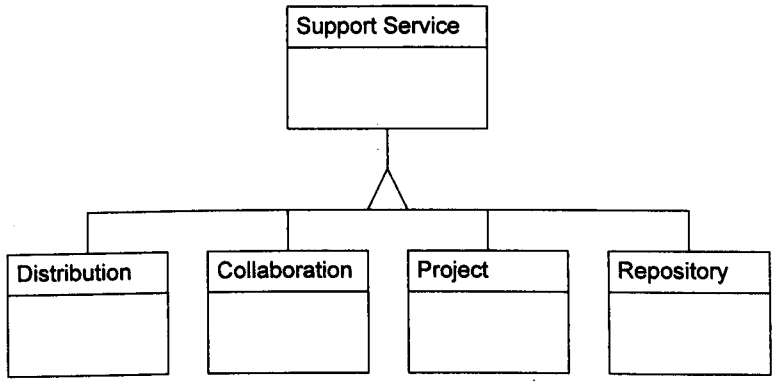


Figure 7-4. Support service class hierarchy.

3½-inch 1.44Mb floppy disk. This makes it extremely portable and able to operate on a wide variety of devices such as workstations, PCs, laptops and even some palmtop devices or PDAs.

7.4.1 Screen shots

The following screen shots of the CONCERT environment in use were taken using an Intel Pentium-class PC running Windows95 with the Java Development Kit version 1.1.5.

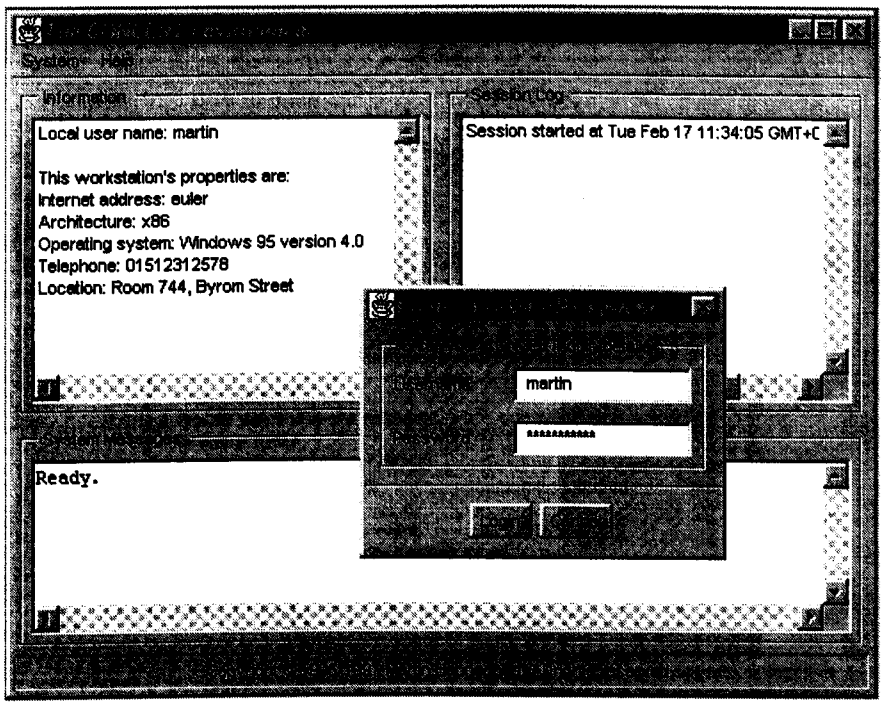


Figure 7-5. Workbench interface and login dialog.

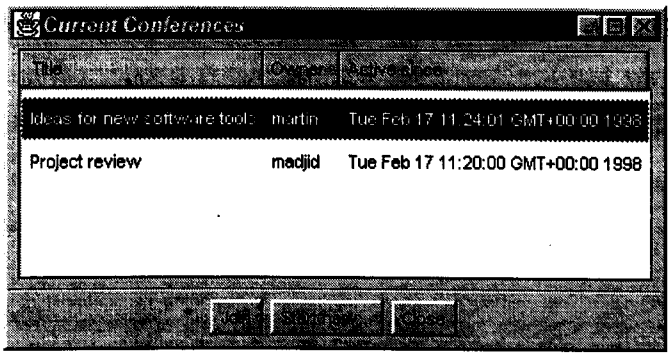


Figure 7-6. Viewing conferences that are currently taking place.

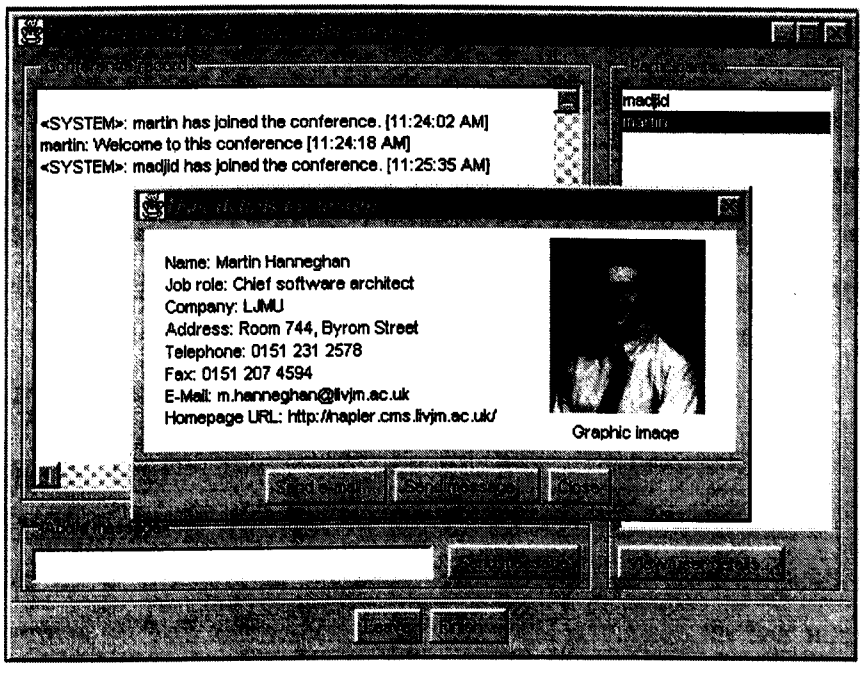


Figure 7-7. In a text conference and viewing user details.

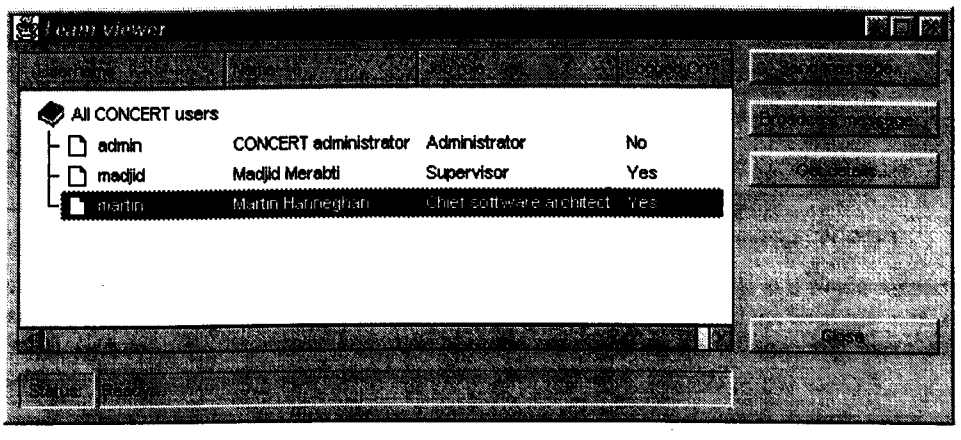


Figure 7-8. Viewing team details.

7.5 Configuration of prototype test-bed

In order to evaluate the support environment, a prototype test-bed was set-up within the School of Computing and Mathematical Sciences at Liverpool John Moores University. This test-bed was configured as a distributed system on top of an existing Intranet system[21]. The configuration consisted of the following off-the-shelf components:

- Four Sun Microsystems Inc. Sparc 4 workstations running the Solaris 2.5 operating system (one to host each of the four Support Services[22]).
- One Intel Pentium-class PC running the Microsoft Windows 95 operating system (to host the global CONCERT ORB).
- A mixture of Intel Pentium-class PCs, Sun Sparc workstations and Apple Macintosh PCs acting as clients to the environment.

All these components were connected via the Internet using the standard TCP/IP protocol (see Figure 7-9). Version 1.1.5 of the Java Development Kit was used on all machines in this network. It should be noted that the particular hardware used was not chosen for any particular reason. Any platform that supports a Java Virtual Machine[23] is capable of performing any of the roles described above.

7.5.1 System operation

In order to facilitate global operation, the CONCERT ORB is placed at a known position on the Internet, i.e. at a known IP address[24]. This position can then be publicised to all interested parties, i.e. both servers and clients. Once this is in place, the four support service servers are executed and each support service makes itself known to the CONCERT ORB by following the binding process described earlier in this chapter. Using the mechanisms of Java RMI, the CONCERT ORB can be requested to provide a remote object reference for each support service bound in its database.

---

[21] Actually a Local Area Network (LAN) that uses standard Internet-based connections (i.e. TCP/IP).

[22] It should be noted however that it is not necessary to dedicate a whole machine to a single support service. Indeed, all the components described in section 7.3 could in fact be executed on the same machine, resources permitting. However, for the purposes of evaluating the robustness of the software and improving failure transparency, the test-bed uses individual machines for each service.

[23] At the time of writing this list includes various Intel / Microsoft Windows platforms, various UNIX platforms, IBM OS/2, Apple Macintosh as well as a growing number of Network Computer (NC) devices. This accounts for upwards of approximately 75% of all computer platforms currently in use.

[24] This IP address need not be fixed and can change at any time, the prerequisite being that affected parties within the environment can be notified that such a change has occurred and modify themselves accordingly.

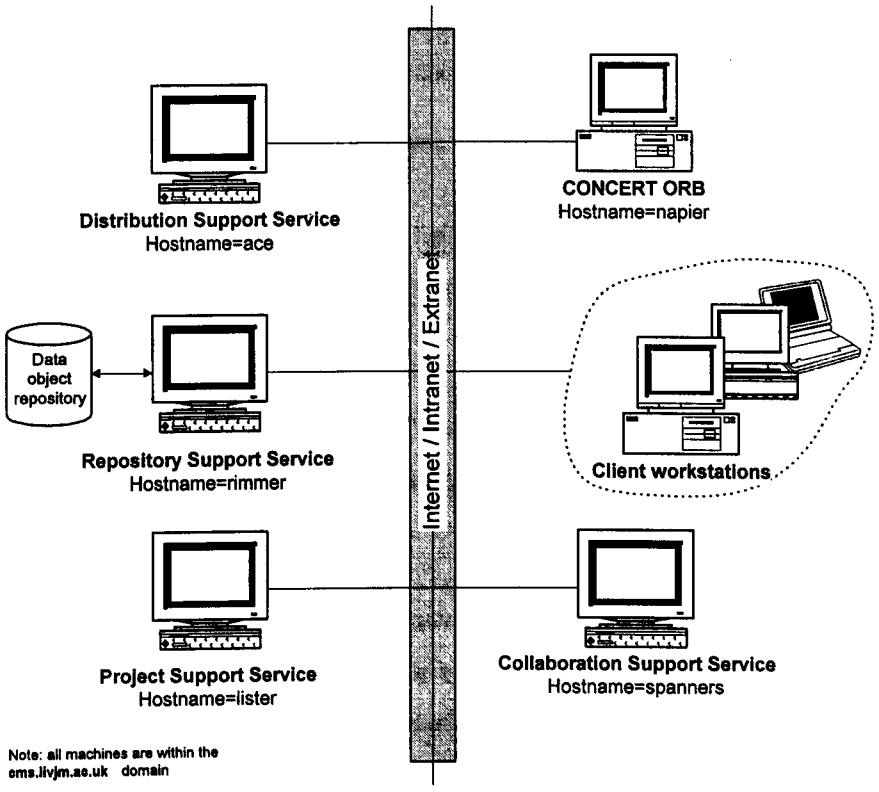


Figure 7-9. Sample configuration of test-bed environment.

Client applications then contact the CONCERT ORB at run-time to determine where the required support services are located (see Figure 7-2 on page 98). This method of location transparency means that both clients and servers within the environment need only ever know one point of contact—the CONCERT ORB. The support services can be dynamically located anywhere on the Internet since they always inform the CONCERT ORB of their current location. This flexible configuration provides a means for the support services to migrate to other machines relatively easily. The migration process can be achieved using the following 3-stage procedure:

1. *unbinding*—first the existing service is unbound from the CONCERT ORB. This is done by calling the *unbind* method in the **CONCERT_ORB** object, e.g.

   ```
   CONCERT_ORB.unbind(ServiceName, BindKey)
   ```

   Here the `BindKey` parameter is the unique key issued when the service was initially bound in the CONCERT ORB.

2. *moving*—the service is physically moved and re-started on a new machine. In practice this means killing the existing process on one machine and restarting it on another.

3. *re-binding*—the service is then re-bound in the CONCERT ORB. This is done by once again calling the *bind* method in the **CONCERT_ORB** object, e.g.

```
CONCERT_ORB.bind(ServiceName, RemoteService)
```

By using the same value for `ServiceName` as the one that was initially bound in the CONCERT ORB, clients can be sure to find the service again once it has been successfully bound.

This process performs the equivalent of the rebind operation of standard ORBs but incorporates the CONCERT architecture's unique security controls.

7.6 Rejected implementation candidates

The viewpoint analysis given in Chapter 4 introduced the notion of a *Technology Viewpoint*. This viewpoint is concerned with the choice of actual technology that will be applied to a design solution. This section describes some other possible technology candidates that could be used to implement the CONCERT architecture. It also discusses the reasons why the actual choices were made and gives a description of the shortcomings of the rejected candidates.

7.6.1 CONCERT ORB versus CORBA-compliant ORB

In order for software clients to access the CONCERT middleware support services there needs to be some intermediary that can be contacted to route requests to the appropriate location. This is the role of the Object Request Broker. A number of options for request brokering are discussed in Chapter 2, but for object-oriented systems the OMG CORBA and Java RMI specifications are more appropriate.

The Remote Method Invocation protocol within the Java language is a lightweight protocol which performs a similar task to the CORBA IIOP (Internet Inter-Orb Protocol). However, it has a greatly reduced 'footprint' in software terms and is an ideal tool for prototyping ORB-based software systems. David Curtis of the OMG (Curtis 1997) reiterates this stating that RMI is "...a viable alternative to CORBA for some applications," citing the overlap between RMI and CORBA. The adoption of Java RMI does not preclude the use of CORBA and IIOP since both technologies can coexist within the same system.

7.6.2 Java versus HTML and CGI

An early prototype of the CONCERT architecture was produced which used features of HTML and CGI in an attempt to provide a heterogeneous environment for concurrent engineering teams (Hanneghan et al. 1996b). This WWW-based implementation proved to

be too restrictive for many of the same reasons that were highlighted in Chapter 3 for similar approaches. These restrictions were evident in the areas of user interface, overall client and server flexibility, session management and server processing load and security.

It should be noted however that the current implementation is flexible enough to allow it to be used over the WWW if so desired with little modification. The workbench application and its associated high-level tools can be utilised using the Java applet mechanism that allows Java applications to be downloaded and executed on client machines in a standard WWW browser. Figure 7-10 shows a screenshot of the WWW-based prototype.

7.7 Summary

So far, this thesis has described three building blocks for the construction of computer-aided support environments for CE. These are: the Concurrent Engineering Reference Model (CE-RM) proposed in Chapter 4; the CONCERT architecture proposed in Chapter 5; and the information system object model proposed in Chapter 6. Each of these three constituent parts can be used in isolation or combined into one complete environment to support Concurrent Engineering. This chapter has described one such combined application of these components in building the CONCERT environment. The CONCERT environment is a

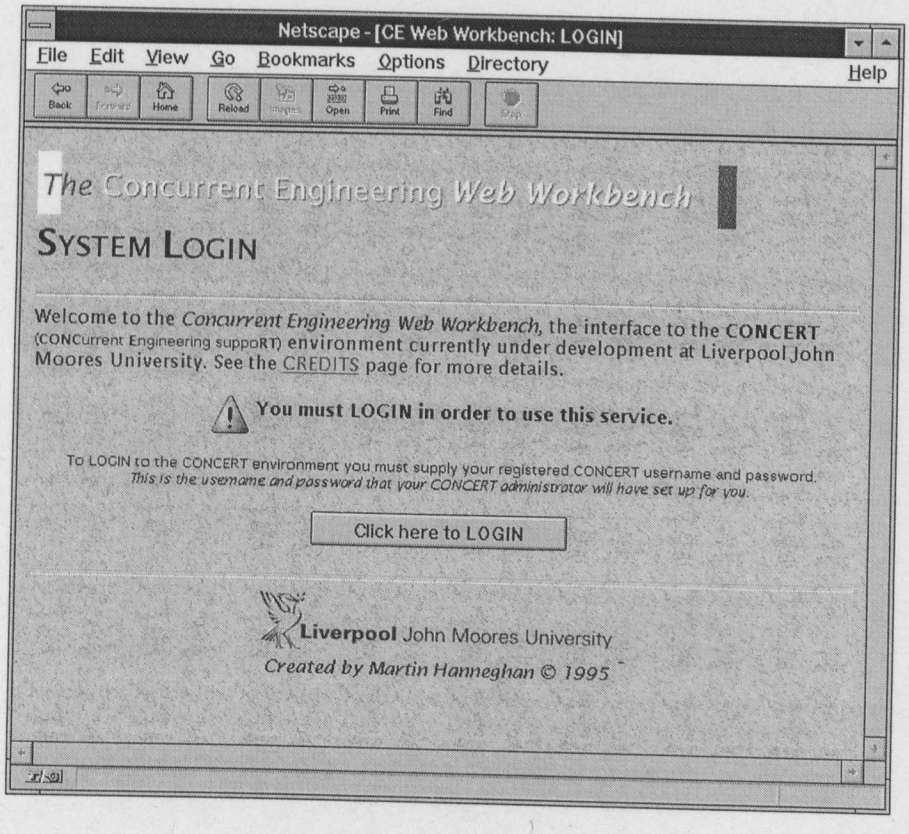


Figure 7-10. Screen shot of early prototype environment.

distributed system built using the Java language that allows global co-operation, communication and information sharing mechanisms. It is a realisation of the CE-RM, CONCERT middleware architecture and the object information system that can be used as a platform for delivering CE projects.

Chapter 8

8. Validation and case studies

8.1 Introduction

In this chapter, an evaluation of the CONCERT environment prototype is given which provides details of two case studies of the proposed environment comparing and contrasting this new environment with existing work. The case study scenarios cover two diverse subject areas: (i) the field of engineering design and manufacturing, and (ii) the field of remote learning. Some of the limitations of this work are also described and finally in this chapter some additional system application areas are discussed to further highlight the generic nature of the system design.

8.2 Case studies

This section describes two case studies to which the CONCERT environment has been applied. These case studies are:

- *Study A*—The development of a guided missile. This project was initially devised as part of the Madefast programme (for further information about Madefast the reader is referred to Cutkosky et al. 1996). This is described in §8.2.1.

- *Study B*—The development and delivery of an undergraduate degree module. This was an application within the School of Computing and Mathematical Sciences at Liverpool John Moores University. This is described in §8.2.2.

These particular case studies were selected to: (i) show the validity of the environment in a traditional product development scenario that employs a Concurrent Engineering strategy; (ii) demonstrate the environment's functionality in global scale concurrent projects; and (iii) highlight the generic applicability of the environment to other complementary problem domains. Each case study is presented under five general topics:

1. The study scenario (to help put the study into perspective.)

2. Characteristics of the study, i.e. issues that are important to the success of the project.

3. The steps that need to be taken to apply the CONCERT environment to this project.

4. Anomalies between the project before and after applying the CONCERT environment.

5. Positive aspects of the study that may prove to be a useful vehicle for future research.

In a majority of concurrent development processes, the chain of events follows a pattern[25] similar to that shown in Figure 4-1 on page 55. The relevant steps shown in this diagram will be revisited during the discussion of the following case studies.

8.2.1 Case Study A: The development of a guided missile

This case study is an application of the CONCERT environment to a modern hard engineering problem, i.e. a relatively complex product concurrently designed and manufactured using CAD / CAM techniques. Due to the impracticalities associated with repeating a major industrial study, the procedures, methods and operations undertaken as part of the Madefast project have been re-applied to the CONCERT environment. This study should therefore be viewed as a "virtual" case study in which the mechanisms of the proposed environment can be validated against a known reference point.

8.2.1.1 Scenario

Madefast is an ARPA-sponsored project to demonstrate technology developed under the MADE (Manufacturing Automation and Design Engineering) program. The project was developed to design and build a "seeker" prototype. The term seeker is used to describe a variety of air-to-air missile that is guided by sensors that are designed to lock onto and track a target. The seeker in the Madefast case study was a much simpler device whose purpose was simply to track a beam of light on a wall, but similar principles are involved. The seeker was to be completed within the six-month time schedule shown in Figure 8-1. This timeline clearly shows the concurrent nature of this project during both the *organisation* and the *design and build* phases.

[25] It should be noted that this is a generalisation and is not the case for all projects. For example, step 3 might not necessarily result in any new data items being produced and therefore step 4 will not occur and step 5 will follow step 3. However, to illustrate the concepts discussed here, it is assumed that a least one item is produced when a task is performed.

The project involved a number of participants situated throughout the United States. The participants included: CMU Engineering Design Research Centre, The Cornell Computer Science SimLab Project, Enterprise Integration Technologies (EIT), Lockheed Artificial Intelligence Centre, MIT Artificial Intelligence Laboratory, MSU Intelligent Systems Laboratory (ISL), Rockwell (Palo Alto Lab), Stanford's Centre for Design Research (CDR), Stanford Knowledge Systems Laboratory, Texas A&M Computer Aided Manufacturing Laboratory and the University of Utah. (A diagram showing each participant and their geographical location is shown in Figure 8-2.)

8.2.1.2 Characteristics of this study

The main characteristics of this study are:

a) The project covers the product life cycle from conception through to actual production starting from scratch with no initial design information.

b) There are high levels of collaboration during the initial planning and design / production phases although there is no initial physical meeting of everyone involved in the project. Electronic collaboration methods are the preferred means.

c) There is wide geographic dispersion of the team members (covering the length and breadth of the United States) and there is no team management structure imposed. Instead, the team is based on a peer-to-peer structure that is commonly found in Concurrent Engineering projects. Additionally, participants are invited to join and leave

the project as and when they desire. These features denote a classic virtual team structure.

d) No tasks are assigned to individual team members. Instead, team members take them on as and when they are capable of providing the resources to do so.

e) There is a six-month time constraint on the project.

f) Documents should be authored in HTML (Cutkosky et al. 1996) using the WWWeasel software package (Glicksman et al. 1994). WWWeasel enables authors to write multimedia hypertext documents that contain links to distributed information. This sets the prescribed common information model for the project.

These characteristics are typical of modern CE projects that are heavily reliant on technology and involve widely dispersed teams working to tight deadlines.

8.2.1.3 Applying the CONCERT environment to this project

There are three steps that need to be taken in order to apply the CONCERT environment to this project. These are each described in detail in this section.

Step 1: Establishing the CONCERT support services location—In the CONCERT environment, the four support services can either be located at some central site or distributed among any combination of the participants' sites. In the Madefast project, the

primary personnel were Stanford, EIT and Lockheed[26] while each of the other participants maintained their own WWW server where information was stored (Cutkosky et al. 1996). These primary sites of the Madefast project therefore seem logical places to host the CONCERT support services. Similarly, the location of CONCERT ORB can be determined using similar criteria.

Step 2: Establishing a project configuration—Instantiating a new CE Project Configuration object will create a new Repository and Bulletin Board. It will also initialise the list of Conferences and Problem Statements for new use. The Team and Project Plan objects are then also created (initially empty) and populated with relevant information. Once the configuration has been created, the virtual team can be formed. This involves creating entries for each team member (e.g. name, address, email address, telephone number, etc.) and assigning these team members to a team i.e. creating instances of the Virtual Team Member and adding these to the Team object. The Madefast team is a dynamic structure that can grow or shrink as required (as specified in §8.2.1.2) and this is modelled by the CONCERT Team object that contains one or more Virtual Team Members.

The Project Plan details all the tasks that are required to fully specify and complete a given project e.g. such as those specified in Figure 8-1 or in more detail as required. This entails populating the Project Plan object with one or more Task instances. Since a CE team should ideally have no management hierarchy, any team member can create new Tasks within a Project Plan. One of the advantages of this formal project plan is that Quality initiatives can be more rigorously enforced to help improve product quality (a primary goal of Concurrent Engineering—*see Definitions in Chapter 1*). In accordance with DIN ISO 9000 quality standards, the CONCERT environment forbids a team member to mark a task as checked if that team member was responsible for carrying out the task. Similarly it will forbid a team member to mark a task as released if that team member was responsible for either carrying out the task or marking it as checked. In the CONCERT information system model, a Project Plan is made up of one or more Tasks and these Tasks can be composed of zero or more sub-Tasks. This fits in well with the Madefast project where groups of participants where allocated main tasks such as Component Design Analysis and Qualitative Modelling[27] in which they could specify sub-tasks, as they deemed necessary.

[26]Source: http://www-ksl.stanford.edu/email-archives/madefast.messages/39.html

[27]Source: http://www-ksl.stanford.edu/email-archives/madefast.messages/39.html

Step 3: Carry out concurrent work—Once the previous two steps have taken place, work can begin on actually carrying out the tasks and producing the product. The activities involved here include:

- *Submission of new information to the repository*—It should be noted that it is only possible to store electronic information in the repository (for obvious reasons!) but this can include digitised versions of media such as photographs, sounds and videos. If physical references need to be made then the item stored in the repository might be a document that can be used to locate the physical reference; e.g. "the stereolithography model is kept in Bill's office."

- *Checking out items from the repository (and subsequent check-in)*—This is where team members perform work on actual items and possibly produce revisions of existing items. Items can be checked out in read-only mode when no editing needs to be performed or in read-write mode when a new version needs to be created.

- *Checking and releasing completed tasks*—Once a given task has been performed it must be checked and released as part of Project Management.

- *Communication between team members*—As part of the communication process (see Figure 8-3) team members may discuss issues surrounding items that have been stored in the repository or any other issues concerned with the project. This may entail holding group conferences or asynchronously sending (e-mail) messages[28].

Step 3 is an iterative process that repeats until the project is complete (as shown in Figure 4-1 on page 55.)

[28]The Madefast project was officially started via an e-mail message and e-mail was the predominant method of communication. The reader is referred to the Madefast archives at the Internet address http://www-ksl.stanford.edu/email-archives/madefast.messages/39.html for the contents of this original e-mail message.

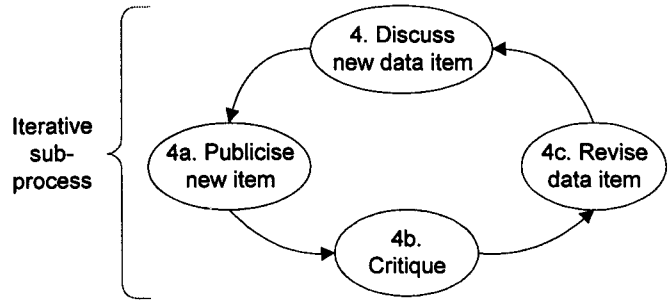


Figure 8-3. The process of group collaboration.

8.2.1.4 Anomalies in this study

The main anomaly of this study which differs in perspective from that of the CONCERT architecture is in the area of data ownership. In the Madefast project, data is owned and maintained by each individual participant site, i.e. if a participant creates some new data item, that participant's site is then responsible for maintaining that data item. A consequence of this is that the collective repository of data is composed of data at multiple sites. This differs from the CONCERT architecture view that data should be maintained and located centrally to avoid islands of automation and reduce wasted effort through data duplication. However, it is possible for the CONCERT architecture to still support this way of working while reaping the benefits of both perspectives: the Madefast view of 'site responsibility' can be achieved by designating repository Data Workspace objects on a per-site basis. Recall that the repository is made up of a number of individual Data Workspaces and these Data Workspaces can contain zero or more Repository Objects. In use, if for example a participant at Cornell University created a new Repository Object called 'Design spec.', they would then place this item in the Data Workspace called 'Cornell University'. The benefit of this arrangement under the CONCERT architecture is that each Data Workspace has a default Access Control List which can be set to allow write access to only team members from the given site (i.e. Cornell University in this case) and read access to all others. This configuration of the CONCERT architecture gives each site control over individual data management and data accountancy while maintaining a centrally held repository of all data items. This example can be seen in the tree diagram given in Figure 8-5.

8.2.1.5 Positive aspects of this study

In applying the CONCERT environment in this case study many of the shortcomings of the Madefast project are addressed. These shortcomings are evident in areas such as:

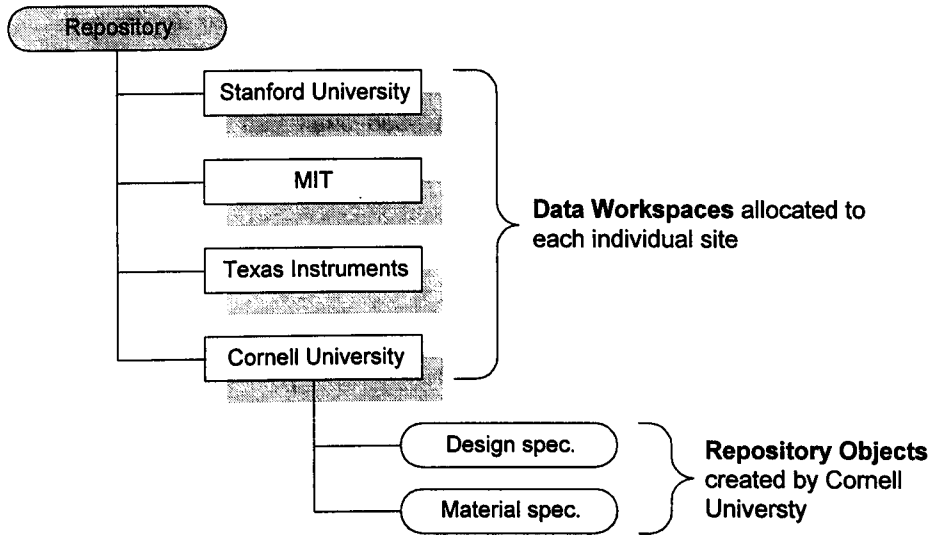


Figure 8-5. Repository tree showing Madefast data management perspective.

- *Security*—In manufacturing projects such as Madefast, which concerns the transmission of sensitive information (i.e. data regarding missile components), the means of this transmission must be secure. Cutkosky et al. (1996) documents that "companies will not send confidential information [over the Internet] if they think it can be intercepted". Since the Madefast environment uses WWW-based tools, the risks to security in this project are well documented. The CONCERT environment however, does not suffer from the same problems since it has been designed with a rigorous security model from the outset (see Chapter 4 for a more detailed explanation of the CONCERT security model).

- *Session creation and management*—The World-Wide Web is a stateless and session-less medium (Cutkosky et al. 1996; Hanneghan et al. 1996b). As an anonymous medium, the WWW has no way of determining user characteristics except for client Internet Protocol (IP) addresses that are used for communication. Since the Madefast prototype uses the WWW, it too suffers from this problem. The CONCERT environment is able to record state and session information from users since it does not use the WWW as its interface. It manages sessions by requiring users to log in and out of the environment before they can utilise the functionality of the support services. CONCERT achieves the same global ease of access however since it incorporates Internet-based connectivity i.e. using Internet Protocol (IP) addresses for location.

- *Group collaboration tools*—Collaboration tools that can be used over the WWW are in short supply mainly due to the complexity of building such tools on such a weak

software platform infrastructure (see session management above). This is evident in the Madefast prototype by the lack of tools such as group voting and bulletin boards. These are two of the tools that have been successfully implemented within the CONCERT environment and which the author believes could have helped improve communications and productivity within the Madefast project.

- *Decentralised document management*—In the Madefast project, each participant site maintains ownership and editorial control over their contribution to the project (Cutkosky et al. 1996). In a worst case scenario, this could allow a negligent participant to substitute a design document with a revised version in order to avoid litigation, if for example the original design developed some costly flaw. The CONCERT environment combats this problem with a non-repudiation policy. Firstly, it maintains a single centrally controlled repository of all data items created during the project life cycle[29] and, secondly, it forbids the removal of any item once it has been submitted to the repository in an effort to maintain an accurate project history.

- *Project management*—Formal project management techniques are not present in the Madefast prototype which instead relies on participants notifying each other when a particular task is complete. The CONCERT environment provides a means for all team members to determine the status of any task from one common point of contact: the Project Support Service. This helps to improve the concurrency and collaboration between team members working on related tasks.

- *Hardware restrictions*—The Madefast demonstration relies on point-to-point conferencing and UNIX workstations running X-Window software (Cutkosky et al. 1996). The CONCERT environment has been designed to operate within heterogeneous environments and will operate on any platform that can support a Java Virtual Machine.

The Madefast project created what Cutkosky et al. (1996) call a "living project web" i.e. a record of all that took place during the Madefast project (e.g. email, data, reports, etc.). This is also true of the CONCERT environment: every email message and conference created within the CONCERT environment is automatically stored in the repository where it can be viewed (i.e. checked-out in read-only mode) like any other data item. This provides a common method for accessing any archive material. Cutkosky *(ibid.)* states that "access to shared, comprehensive online documentation accelerates the process of reaching a consensus" and this is a primary objective of the CONCERT environment. Cutkosky *(ibid.)*

[29] Note that this includes all conferences, group decisions and e-mail messages sent within the environment and that each item is uniquely traceable to a specific team member at a specific time.

also reports that a "project web" is useful for bringing new team members up to speed with a project and again the CONCERT repository facilitates this method of working.

8.2.2 Case Study B: the production and delivery of an undergraduate degree module

This case study is an application of the CONCERT environment to the domain of remote learning which aims to show the generic nature of the environment and how it can be applied to diverse domains.

8.2.2.1 Scenario

The School of Computing and Mathematical Sciences at Liverpool John Moores University runs a number of undergraduate and postgraduate degree programmes. Each programme is composed of a number of prescribed modules which students must study. The development of a module requires that a number of lecturers work together designing and writing the learning materials that will fulfil the module syllabus. To complete the remote learning process, it also requires that the students can then access the module material on-line and electronically submit completed work for assessment. To explain the concepts of this case study, a single module has been used although the entire programme of modules could also be developed in the same way.

8.2.2.2 Characteristics of this study

The main aspects of this study which need to be addressed are:

a) There is an initial high level of collaboration and consultation period between the lecturers while the course material is developed. This may involve many lecturers iteratively developing the same item or concurrently developing multiple items. The course material includes such items as lecture notes, practical session notes, slides, diagrams, tutorial questions and answers. In addition to this, formal examination questions must be specified and placed in a private location where students cannot access them until the appropriate time.
b) There will be high levels of concurrency from students during course delivery as they access the learning materials. Students may access the material from various locations around the campus, from home, or indeed from anywhere in the world.
c) There is a need for secure submission to prevent plagiarism. When a student submits a piece of work for assessment it must be a one-time operation. Non-repudiation is also needed to unambiguously determine who submitted the work and at what time.
d) There needs to be some means or recording the entire process for external moderation and assessing quality standards.

e) There is a need for restricted access to data items. For example a moderator must be able to view exam questions before the exam but students should not. Also when a student submits a piece of work, only the lecturer should be able to read it in an effort to prevent plagiarism.

f) Documents are required to be formatted in plain text (ASCII) or any of the Microsoft Office™ software suite formats (i.e. Word™, Excel™, PowerPoint™, Accces™, etc.). This sets the prescribed information model to be used within the project.

It can be seen that this scenario exhibits similar characteristics to Study A, including levels of concurrency, widely dispersed team members (students) and security considerations.

8.2.2.3 Applying the CONCERT environment to this project

The same three steps that were carried out in Study A also apply to this study. To reiterate, these were:

Step 1: Establishing the CONCERT support services location—Since the CONCERT environment is being used to facilitate learning within a University, it is considered reasonable to host the CONCERT support services centrally at this University. This may or may not involve running each service on a separate machine (as discussed in Chapter 7). Likewise the CONCERT ORB can also be located at the University.

Step 2: Establishing a project configuration—Once again, a CE Project Configuration is created. This time the virtual team is composed of all the stakeholders in this particular project. These are shown in Figure 8-6 and include:

- *Lecturers*—These are the person(s) responsible for developing the course content and delivering the information.
- *Course Leader*—This is the person with overall responsibility for the entire course.
- *Administration staff*—These people are responsible for enrolling students and general administration such as informing students and staff about timetables, examination dates, regulations, etc.
- *External examiners / moderators*—These people are required to ensure that the level of course content and examination is of the required national standard.
- *Students*—These are the people who study on a particular course or programme.

The Repository object will be used in this case to store all the items produced during the lifetime of the module. This includes the items produced by the lecturers and any work that will be submitted by the students.

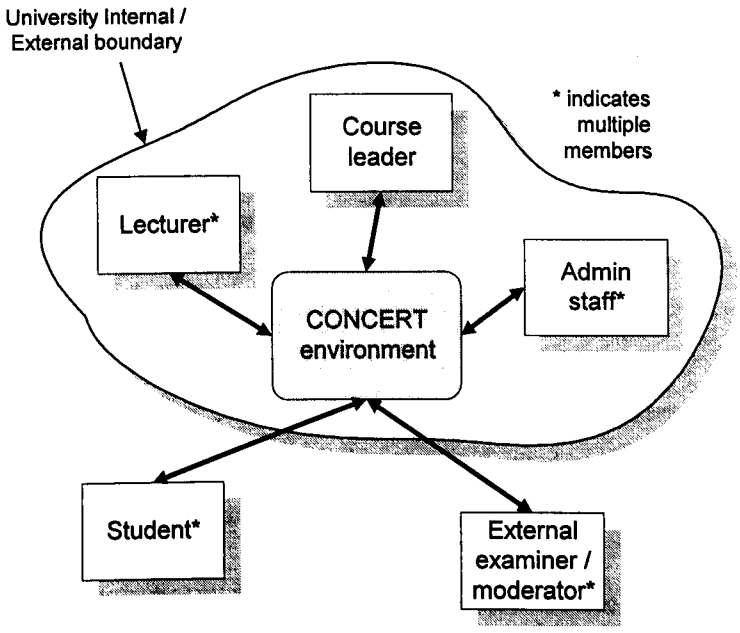


Figure 8-6. Virtual team (case study B).

It is not absolutely necessary to utilise the Project Plan for all types of CE project. Projects which require plans are typically those which need to have tasks signed-off when they are completed in order to prove that a particular milestone has been achieved, e.g. for contractual reasons. For projects of this nature, the CONCERT environment allows a record to be maintained that can provide details of the status and sign-off dates of any task. For this case study, the project plan contains tasks such as write course material, set examination question(s) and set coursework assignment(s). This enables the external moderator to clearly see what tasks are complete and which remain. If for example the course does not need to be externally moderated, it may be convenient to ignore the Project Plan.

Step 3: Carry out concurrent work—The activities involved here include:

- *Development of course material*—Initially the lecturers develop the course learning materials using a concurrent, iterative process as described in Study A.

- *Accessing learning resources*—During the actual delivery of the module, the students can checkout the learning materials from the repository in read-only mode (note that the permissions set by course tutor). During scheduled classes, the tutor can broadcast messages to each student's workstation to draw attention to specific events or items just as he would in a traditional lecture.

- *Submitting information electronically*—Student must submit work to be assessed by the tutor. This involves checking in completed work with read-write access permissions set for the tutor. A Data Workspace can be created for each specific class, group or student as deemed appropriate by the lecturer. At examination time the course leader can modify the access permission for the exam script to give read access to the students.

- *Communicating feedback*—The tutor annotates the students' work with suitable feedback comments and saves a new version in the repository that can only be read by the appropriate student. The tutor then submits examination and assessment marks into the repository as a spreadsheet model for the external moderator. He can also e-mail individual students with their grade if required.

Step 3 above cycles until such time that the module is deemed complete. By concurrently running more than one module, a whole programme of courses could be delivered using the CONCERT environment. This could be achieved by either using a single CE Project Configuration (and hence one Repository) for all modules within a programme, or by using individual configurations for each module.

8.2.2.4 Anomalies in this study

Remote learning is an obvious example of a problem that would benefit from videoconferencing (see Chapter 2 for further information on videoconferencing). The traditional teaching method of a tutor standing in front of a class of students is difficult to simulate electronically due to the lack of visual clues and observations of group dynamics within a class. Videoconferencing could provide a means to begin to address these issues and is a tool that would benefit other forms of CE (see discussion on *Further work* in Chapter 9).

By using the CONCERT environment in the way described above, plagiarism (i.e. the submission of another student's data files as one's own) cannot be avoided. It should be noted however that this is also a problem with traditional teaching methods.

8.2.2.5 Positive aspects of this study

The capability of the CONCERT environment as a remote learning aid is further supported by a number of positive points that have emerged as a result of this study. These include:

- Team members can participate from home which may be significant for handicapped or less mobile people.

- Discussion groups can make use of the bulletin board, conferencing facilities and group decision-making tools to provide peer help with assignments and tutorials.

- Students can study when it is most convenient for them to do so. Since all learning materials are stored in the repository (including conferences and group decisions), students can access them after the event if they need to catch up on a particular topic.

- External moderators and assessors can easily evaluate the quality of a course by tracing its project history. This may be a requirement for accreditation by professional bodies.

- The flexibility of the CONCERT architecture allows the Workbench application to be individually customised to specific domains. In this case any references to project management could be removed from the student's Workbench application menus since they are not relevant to the student in this case.

- Since student assessment is in electronic format, it may be possible to automatically check work for signs of plagiarism via some special software. If this is desired it may prove useful to implement this function as an operation available via either the CONCERT Collaboration Support Service or Repository Support Service for example.

This case study has shown some interesting similarities between CE viewed from a traditional manufacturing standpoint and the seemingly non-related field of education and remote learning. It is my opinion that valuable lessons can be learned from both domains when trying to support a single perspective.

8.3 Evaluation

Chapter 3 of this thesis described a number of barriers to providing tools to support concurrent engineering. It also set out a number of requirements (prescribed by the author, leading practitioners and academics) for future environments that were deemed essential factors for success. Each of the main points raised in Chapter 3 have been re-classified in this section and grouped under one the following headings: communication, project management, data management and geographic dispersion. This section will show how the work presented in this thesis addresses each of the issues raised.

8.3.1 Communication

A number of authors cite communication factors as requirements for CE support. Examples from Chapter 3 include:

- Trapp et al. (1992) describes a need for "communication and co-ordination tools which support interaction, negotiation and conflict resolution."

- Londono et al. (1992) describes a need for tools to help achieve "common visibility" among team members.

- The NIIIP Consortium (1995) specifies a requirement for "common communication protocols."

- Smith (1988) calls for "communication of upstream and downstream concerns" and "co-ordination of group problem solving activities."

In addition to these, the author (in Chapter 3) also specifies that CE support requires "...a communication sub-system for effective participation (both asynchronous and synchronous)".

The CONCERT environment addresses these issues by providing a dedicated Collaboration Support Service which allows various high-level mechanisms such as e-mail, group problem solving tools, synchronous text conferencing and bulletin boards. In addition if provides a low-level protocol for allowing applications to utilise a publish-and-subscribe facility for sharing information between multiple concurrent users.

8.3.2 Project management

Project management factors also rate highly in support system requirements. Examples from Chapter 3 include:

- Londono et al. (1992) calls for "tools for assessment and tracking of progress."

- Trapp et al. (1992) describes a need for "mechanisms for sharing process data."

- Molina et al. (1995a) requests tools that can "monitor the history of the design process."

The author also adds "facilities for controlling and tracking the project tasks and resources" from Chapter 3 to the above list.

The CONCERT environment addresses these issues by providing a data model for maintaining project management information that includes Quality aspects that are influenced by ISO 9000. The Project Support Service additionally provides a single contact point for obtaining such information.

8.3.3 Data management

Concurrent engineering produces large amounts of data that must be managed effectively. Many authors cite data management as a major requirement, for example (from Chapter 3):

- Sobolewski and Erkes (1995) call for "management of security, persistence and maintenance of the information."

- Molina et al. (1995a) describe a need for "information and knowledge sources...that can be readily modified and that are easily accessible."

The author adds the provision of "centralised, accountable data management facilities" and "data format translation" (from Chapter 3) to this list.

The CONCERT environment addresses these issues by providing a unique data model that can be used to store information produced during a CE project and by employing a 3-tier security model that protects the information stored in the CONCERT repository. The Repository Support Service provides a single means of access to the repository contents.

8.3.4 Geographic dispersion

Wide geographic dispersion is becoming increasingly prevalent in CE projects. Many authors therefore cite this as a basic requirement for CE support. For example (from Chapter 3):

- Londono et al. (1992) describe the need to be able to "scale up easily from small projects with a few people to large, complex projects with a great number of people."

- Trapp et al. (1992) reiterates this adding the need for "transparent mechanisms [to work on] heterogeneous hardware platforms in multiple geographic locations."

- Sobolewski and Erkes (1995) describe a need for "a locator for service taxonomy" and "access to distributed information from distributed computer workstations."

To these, the author adds "the ability to treat geographically dispersed resources as though they were local", "generic and portable" and "facilities for inter-networking" (see Chapter 3) as requirements which support geographic dispersion within CE teams.

The CONCERT environment addresses these issues by intrinsically supporting the notion of transparency throughout all of its individual components. A dedicated Distribution Support Service further emphasises the importance of this aspect in the CONCERT architectural

model. The CONCERT ORB provides additional support for distribution within the environment via its dynamic service database that is used to inform clients of the current location of services.

8.3.5 Evaluation summary

It is by no coincidence that the grouping of the points raised in the above four sections corresponds to the functionality contained in the four CONCERT support services. The logical grouping of the issues in this way provides a means for identifying required operations of new CE support environments. An interesting property of this grouping shows that as few as four support services need be devised to support the listed requirements[30]. The CONCERT support services provide a vehicle for performing these operations in an open, easily accessible way. It can be seen that the CONCERT architecture and environment provides an adequate and "viable software architecture" (Trapp et al. 1992) from which to support CE.

8.4 Scope of proposed design and implementation

As with any major body of work, there are limitations with the proposed system design and the current implementation. This section highlights these and suggests ways to overcome these problems. A summary of the relative merits and limitations of the proposed system design and implementation can be found in Table 8-1.

Having a centralised database (repository) means that there is a high dependency on the availability of the database server, i.e. the Repository Support Service. If for some reason the database server stops functioning, data access is forbidden. This high dependency also creates a high load on the database server machine especially when servicing requests from multiple users concurrently. A high load on the database server will effectively increase response times to users resulting in user dissatisfaction with the system. In the CONCERT environment, it is believed that this is outweighed by the fact that a centralised database ensures that data is more secure, consistent and simpler to administrate.

[30] By comparison, the OMG CORBAServices (Object Management Group 1997) total fourteen distinct services.

Table 8-1. Advantages and limitations of proposed system.

Feature	*Limitations*	*Advantages*
Centralised database	• High dependency on database server availability. • High load imposed on database server can increase user response times.	• Data is more consistent than can be achieved by integrating "islands of automation." • Simpler to administrate.
Internet-based	• The system is reliant on the quality of the network and connections. • Security fears associated with Internet traffic.	• The Internet is readily accessible to smaller companies as well as organisations in developing nations. • The system can also be applied to Intranet and Extranet environments without modification.
Implemented in the Java language	• Java is a relatively new language and is constantly evolving. • Java has yet to be used in large-scale commercial applications.	• Freely available. • Java is now being incorporated into leading Operating Systems, which will broaden its use.
Object-oriented model	• Software wrappers are required to integrate legacy systems.	• It is easier to apply changes and new features to the model.
No prescribed information model	• The STEP protocol is a very effective model for CE.	• The repository can store any electronic data formats.

In an effort to enable easily accessible, global collaboration, the CONCERT environment uses the Internet as its communication medium. Quality of Service (QoS) and security are the two main problems with this approach. The QoS aspect is impacted by network bandwidth and peak access times while Internet security fears are well documented. QoS can begin to be addressed when considered in an Intranet environment (i.e. an internal, controlled network) where access levels can be predicted and bandwidth can be controlled. Security considerations require additional measures to be taken when transporting sensitive

information over the Internet. The CONCERT environment does address the highlighted security problems at a number of levels (see Chapters 4, 5 and 6 for more details).

The CONCERT environment does not prescribe that any particular product information model be used. Instead, it allows any data item in electronic format (i.e. word-processed document, CAD drawing, spreadsheet analysis, etc.) to be placed within the repository. However, the repository structure described in Chapter 6 does provide sufficient structure for product information models to be created in a hierarchical manner such as those used in a Bill of Materials, or alternatively in a manner which implies functional decomposition.

8.5 System application areas

A consequence of the generic design of the CONCERT architecture and environment is that it can be applied to a number of additional problem areas, i.e. it is not restricted to the field of Concurrent Engineering. This section describes some of these additional application areas in more detail.

8.5.1 Collaborative Engineering / Manufacturing

This application area has been shown in case study A (see §8.2.1) as applied to a real concurrent engineering problem. However, collaborative engineering projects do not necessarily have to be CE projects. In some development processes it can prove extremely difficult to run tasks in parallel, thus forcing sequential development of the product during certain stages. This development may still require collaborative development between non-collocated team members, for example a manufacturing plant that awaits designs from a foreign collaborator before production can begin. The CONCERT environment, in its facilities for information sharing and group collaboration and communication, caters for this scenario.

8.5.2 Remote learning

Remote learning is the process whereby a number of students access learning information (i.e. lectures, class notes, etc.) remotely using either electronic or conventional means (such as the postal service). The entire life cycle for this process however, encompasses development of course material, access of learning resources, undertaking of examinations and subsequent assessment. It can be seen that this has many parallels with the tasks undertaken during Concurrent Engineering such as group working, concurrent remote access from non-collocated users and information sharing. This application area has been shown in greater detail in case study B (see §8.2.2.)

8.5.3 Teleworking

BT Laboratories (1995) defines Teleworking as: "...a flexible way of working which covers a wide range of work activities, all of which entail working remotely from an employer, or from a traditional place of work, for a significant proportion of work time. The work often involves electronic processing of information, and always involves using telecommunications to keep the remote employer and employee in contact with each other." From this definition, it is apparent that Teleworking involves concurrent working from widely dispersed team members. It also stresses the need for communications and electronic processing. This makes Teleworking a real candidate to benefit from some form of CE support environment such as the CONCERT environment. One difference between Teleworking and CE can be found in the organisational structure that is imposed upon the team. Teleworking typically maintains traditional hierarchical management structures while CE eschews this in favour of work teams with a flattened hierarchy in which there is little or no management presence. Both of these organisational methods are accommodated in the CONCERT environment through the Project Support Service that provides a number of facilities for project management.

8.5.4 Sensitive sources of collaboration

The manufacture of traditional mass-produced goods has relatively low security requirements since the onus is on producing the goods as cheaply as possible. However, for more sensitive items such as aerospace and weapons manufacture (case study A is such an example), security is an important issue. Here, it is of paramount importance that designs, plans and specifications are kept confidential from hostile countries or competitors. These high security requirements were a primary reason for including such a stringent security model in the CONCERT architecture since CE, by its nature, is a means of gaining competitive advantage. The CONCERT environment is therefore suitable for application areas such as those described here since it provides a robust security model from three aspects: user authentication, access control and data encryption.

Additional examples of sensitive areas of collaboration include newspaper or magazine production. This is a very competitive area where it is important to be first to print with a leading story. Additionally there may be many collaborators on a particular story, e.g. researcher, writer, editor, photographer and typesetter who each need to share information and communicate with each other (generally in electronic format).

National or international Police collaboration, where an offence covers many jurisdictions and involves many Police forces is another such example of sensitive collaboration. Here there is a need to share confidential crime files and communicate with fellow officers, i.e. it is a highly collaborative activity. Applying a Concurrent Engineering strategy to this problem domain is desirable in an effort to reduce the cost and time it takes to catch criminals. This is analogous to reducing the development cost and lead-time in a traditional manufacturing sense. Existing projects in this area such as LinguaNet (Johnson 1997) and the Schengen Information System (ibid.) exhibit many parallels to the work described here.

8.6 Summary

This chapter has shown how the CONCERT architecture and environment implementation has been applied in two case studies. These case studies cover two very different problem domains and yet are adequately catered for by the proposed design. A number of additional application areas were also discussed which serve to reinforce the generic capabilities of the proposed architecture and environment. The genericity of the CONCERT architecture and environment is not at the expense of supporting a Concurrent Engineering strategy; instead, it shows how the goals of CE have many parallels with other problem domains and therefore provides a healthy basis for cross-discipline research into the many inter-related fields.

Chapter 9

9. Conclusions and further work

9.1 Introduction

This thesis has described the results of research into the problems of designing and building Concurrent Engineering support environments. In particular, it has presented a prototype support environment that can enable virtual team workers to collaborate on CE projects.

This chapter summarises the thesis and highlights its contributions to this particular research field. Finally, some ideas for future work and concluding remarks are presented.

9.2 Thesis summary

Chapter one introduced the main theme of this thesis and described the background for this work. It highlighted the main problems associated with CE projects and put forward a newly proposed definition of concurrent engineering in the light of the findings of this research.

Chapter two described a number of technological factors that have had an influence on the ideas proposed in this thesis. These included distributed computing environments and Computer-Supported Co-operative Working (CSCW) in addition to recent advances in product information modelling and virtual working. It also looked at a number of complementary initiatives such as the Internet and World-Wide Web and the prevalence of distributed object-based systems.

Chapter three presented a survey of current state-of-the-art in CE support systems and gave a summary of the views of CE practitioners and leading academics. It concluded by presenting a detailed and up-to-date list of user requirements for CE support environments while noting key areas of research that need to be addressed.

Chapters four, five and six presented the design of a new architecture to facilitate virtual Concurrent Engineering teams. Chapter four introduced this design through a novel viewpoint-based reference model that takes into consideration the many various facets of CE projects and synthesises these into a single coherent model. From this reference model, a

high-level computer system architecture which describes the various components needed to address the issues raised was discussed in Chapter five and an object-oriented information system model which captures the unique information requirements of CE projects followed in Chapter six.

Chapter seven demonstrated a physical implementation of the environment described in Chapters four, five and six and discusses the decisions made in implementing the system in the way proposed. An evaluation of the implemented system is given in Chapter eight which looks at its application in two quite different case studies and this Chapter concludes with comments on how the proposed system can also be applied to other problem domains in addition to Concurrent Engineering.

9.3 Aims and objectives revisited

Chapter 1 introduced a number of aims and objectives for this thesis. To summarise, the main aim of this research was "...to present a global computing and information system architecture that will facilitate the day-to-day running of Concurrent Engineering projects." In doing this, it was hoped that the following questions could be answered:

1. What aspects of current computer technology are useful to Concurrent Engineering?
2. What new technologies are needed to support CE?
3. What frameworks exist for supporting CE?
4. What components are useful building blocks for next-generation frameworks?

In order to achieve these aims, the following objectives were also highlighted in Chapter 1:

a) Acquire an understanding of current CE product development processes.
b) Review current computer-based systems used in manufacturing.
c) Develop a computer systems architecture that can be applied to manufacturing organisations globally.
d) Develop an information systems model that can be applied to CE projects.
e) Effectively demonstrate the complete system in a live application.

Questions 1 and 2 (and hence Objectives a and b) posed above dictated the contents of the literature survey presented in Chapter 2. An answer to question 3 (in relation to Objective b) is given in Chapter 3. Chapter 4, which presented an object-oriented (component-based) reference model for CE, proposes an answer to question 4. Objectives c and d are the subject of chapters 5 and 6 respectively while Objective e is the topic of chapters 7 and 8.

The following section discusses these points in relation to the contribution to knowledge offered by this work to show how it addresses the main aim of the research.

9.4 Contribution to knowledge

In the course of this research, some key ideas have emerged that contribute to the body of knowledge in the field of Concurrent Engineering. Firstly, this work draws on and integrates a number of disparate problem domains as depicted by the shaded region in Figure 9-1. These include the fields of Computer Supported Co-operative Working (CSCW) and distributed computing which have led to the relatively new topic of virtual team working; object technology and interoperable systems which more recently has expanded to include emergent Internet technologies; data storage and management which includes product information models; and finally manufacturing systems.

The proposed system provides a uniform environment for users to store, access and process information from the entire product life cycle as opposed to succumbing to the weight and complexity of multiple 'islands of automation' (Koonce 1995). It does this while supporting both collocated and non-collocated collaboration transparently by employing the Internet as a global information medium and providing a means for both synchronous and asynchronous communication.

The main contributions of this work can be summarised by the following points. It provides:

- A new working definition of Concurrent Engineering.

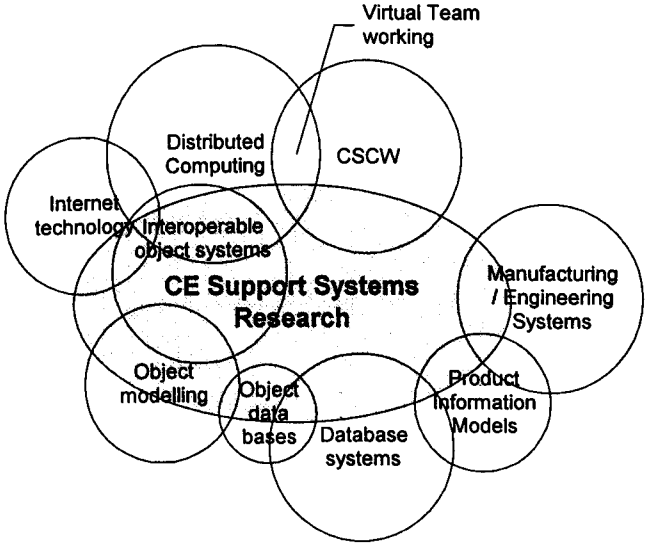


Figure 9-1.
Research gap addressed by this research.

- A state-of-the-art set of user and system requirements for CE support environments.
- A new reference model for designing CE support systems.
- A distributed computer system architecture.
- An information system model.
- A working reference implementation.

Each of these aspects is described in more detail in the following sub-sections.

9.4.1 Definition of Concurrent Engineering

There have been many definitions of CE over the past decade. The following definition attempts to reconcile previous terminology in light of recent work and by incorporating the findings of this thesis.

> *"Concurrent Engineering is a systematic approach to parallel development of all product life cycle activities, from initial conception through design, planning, production and disposal. It is an enriched communication infrastructure which is unconstrained by geographical location that encourages right-first-time methods through cross-functional team working and consensus."*

This definition serves to provide meaning to modern CE practitioners and the next generation of support environment designers.

9.4.2 Requirements for CE support environments

Chapter 3 highlighted nine fundamental system and user requirements for CE support. These requirements are based on experienced CE practitioners' views and leading academic contributions. In addition, consideration has been given to recent technological advances that make CE more accessible to organisations.

9.4.3 Reference model

The Concurrent Engineering Reference Model (CE-RM) proposed by the author and described in Chapter 4, provides a framework for building CE support systems. The CE-RM is a viewpoint-based reference model which draws on existing work on distributed

computing systems and expands this into the area of Concurrent Engineering support systems. Designed using object-oriented methods, this reference model can be easily adapted or extended to accommodate future viewpoints.

9.4.4 Distributed computer system architecture

The CONCERT architecture developed as a result of this research, is an object-oriented middleware architecture that enables distributed clients and servers to inter-operate. Once again designed using object-oriented methods, this middleware architecture employs an Object Request Broker to enable transparent participation regardless of physical location.

9.4.5 Information system model

The CONCERT information system is a stand-alone object-oriented model that captures the information requirements of CE projects. When used in conjunction with the CONCERT architecture, it provides a complete support solution for CE development.

9.4.6 Reference implementation

The CONCERT environment is a realisation of the CONCERT architecture components. Built using the Java language, it provides a reference platform for future work and enables CE practitioners to evaluate CE support tools, for example new and existing collaboration schemes.

9.5 Further work

In this section, a number of paths are described from which the further exploitation of the work in this thesis can proceed. Undoubtedly there is scope for extension of the existing support services and this task is eased by the fact that the CONCERT architecture has been designed in an open manner. Firstly however, the application of the CONCERT environment to real world industrial applications is described.

9.5.1 Industrial trials

The Madefast case study described in Chapter 8 is typical of the type of project that can benefit from an environment such as CONCERT. Since this particular case study was performed virtually, additional industrial applications are required to further develop this work. It is therefore important for the further success of this research to encourage industry to use the ideas presented in this thesis with the goal of further developing the CONCERT environment software perhaps by exploiting the ideas for further work presented in this section.

9.5.2 *Agent technology*

Case and Lu (1996) argue that software agents should play a key functional role in collaborative design environments and Petrie (1996) documents a number of projects that employ agent technology for engineering. The CONCERT architecture does already support the concept of agents when required for certain tasks. For example, the Repository Support Service will act as an agent to resolve version conflicts. It does this by informing both conflicting parties of the situation and mediating the resolution. Having said this, it might be also be feasible to implement an additional *Agent Support Service* or 'agent shell' (Williams and Taleb-Bendiab 1997) that would handle all agent traffic and correspondence autonomously on behalf of the other support services. The support services could then delegate responsibility of agent-based tasks to the Agent support service. This would certainly reduce the workload of the other services if this were a determining factor.

9.5.3 *Additional collaboration services*

There is scope within the environment for further enhancement to the Collaboration Support Service in particular. The work done in the field of CSCW (see Chapter 2 for more details) is very promising for assisting a CE process. While the current tools provided by the Collaboration Support Service give an adequate level of collaboration support, additional tools such as shared whiteboards, brainstorming tools and videoconferencing would also prove useful. To provide these tools within the CONCERT environment is beyond the scope of this thesis, however, to open design of the architecture allows additional tools such as these to easily incorporated in a process that is transparent to the user.

9.5.4 *Legacy software wrappers*

Much research, especially by the object-oriented community, in the area of software wrappers has provided a means for legacy system integration. Organisations that have reengineered processes typically need to retain legacy applications. Concurrent Engineering is one such reengineering example. Within the CONCERT environment, the use of wrappers has been promoted as a means for integrating both legacy applications and legacy database systems. What is beyond the scope of this research is the specification of a formal means for wrapping these legacy systems. One area of future research could be to provide computer-aided support tools for wrapping legacy software. These tools could be used to automatically generate a software wrapper that could be used to integrate legacy components into the CONCERT environment.

9.5.5 *Telephony*

Growing interest in the area of Computer / Telephony Integration (CTI) within business organisations (Clegg 1997) could also prove of use to organisations employing CE. The

transition from traditional to computer-based communications technology can sometimes be a difficult and painful process for some team members. The telephone however is a much-used tool in business. CTI integrates the telephone with computer technology making the telephone a more effective tool. Examples of how this technology could be used within the CONCERT environment include:

- Displaying team member contact information on screen when an incoming call is received.
- Automatic call redirection based on a team member's current location (as determined by the environment).
- Extending the Collaboration Support Service to enable multi-way conference calls to be initiated based on contact information maintained by the environment.

9.5.6 Offline access and batch processing

Another area that could be investigated is that of providing a procedural language that could be parsed by the CONCERT support services in order to carry out certain tasks in an automated fashion. In its current guise, the CONCERT environment utilises interactive communication with users, via the Workbench application, in order to fulfil a request. A language could be devised so that various tasks could be automated, for example uploading a constantly changing price list on a weekly basis. This language would be able to specify what operation needs to be done, what data to include and what alternative action to take in case of conflict. In this sense, the language would need to incorporate features of database query languages such as SQL or OQL and fourth-generation languages (4GL).

A hypothetical example to store a regularly updated document in the repository is shown in Figure 9-2. This procedural code could be either written by hand (for knowledgeable users) or generated by a software application that could guide the user through the steps they wish to automate.

```
LOGIN AS ("John", "password");
  USING CONCERT.RepositorySupportService
    UPDATE RepositoryObject WHERE OID = 8192233
      WITH DataFile = "C:\DESIGNS\CURRENT.DOC",
        Description = "Current design",
        ReadAccess = (ALL),
        WriteAccess = ("John, Dave, Mary")
    ON ERROR NOTIFY VIA E-MAIL AND NEXTLOGIN;
  ENDUSING
ENDLOGIN
```

Figure 9-2. An example of a procedural language for task automation.

9.5.7 Performance enhancements

The current implementation of the CONCERT environment is sufficiently powerful to support Concurrent Engineering, yet simple enough to allow proof of concept without running over budget or time. There are however, a number of improvements that could be made to enhance the performance of the current system implementation. These include:

- *Service replication*—to allow more than one instance of a support service to run simultaneously thereby providing better response times to users.
- *Data repository replication*—to allow team members to access data from a local site rather than some remote site.

By incorporating these features, users could be automatically directed by the CONCERT ORB to the closest physical service to improve network response. Both these measures would improve system response times for team members using the system, but would increase the complexity of the current software due to the increased level of synchronisation that is needed for replication. This complexity however, is a development issue (i.e. it remains transparent to the user) and therefore could be implemented without affecting users of the system.

9.5.8 Data repository

The current repository is a simple object database that was developed purely to accomplish the tasks required within the architecture. It is simple, yet effective but provides none of the advanced facilities of a true Object Database Management System (ODBMS). Further work would ideally look to using an ODBMS in place of the current repository. The benefits of this are twofold: firstly the *Repository support service* would be able to delegate some of its

processing to the ODBMS engine thus easing its workload. Secondly the ODBMS would be able to provide object replication as standard[31] (see §9.5.7 for further details).

9.5.9 Further integration with external Information Systems

The information produced during the life cycle of a concurrently engineered product may account for a small proportion of the total information produced by an enterprise. Enterprise information provision seeks to produce quality information from across the board of all the enterprise's information systems. Ideally, this information needs to be gathered transparently and with no human involvement. It would therefore be useful if the CONCERT environment could be extended to gather enterprise information from complimentary manufacturing Information Systems such as financial systems, MRPII systems and Executive Information Systems.

9.5.10 Product information retrieval

A consequence of storing data concerned with the entire product life cycle is that the repository quite quickly fills up with a sea of information. 'Fishing' for an item from this 'sea' relies on the searcher's knowledge of the structure of the repository, i.e. which item they are looking for, in what project and in what workspace. This doesn't really facilitate the searching of past project histories in which the searcher has no knowledge of this structure.

Further research could concentrate on developing new techniques to search, navigate or visualise the data in the repository (Keim and Kriegel 1996). Examples of areas that could be explored include:

- The use of Virtual Reality Modelling Language (VRML) (Bell et al. 1995) to visualise, and navigate through, complex data hierarchies.
- The application of Artificial Intelligence (AI) techniques to locate information from past project histories based on natural language processing for example.

In addition to this there is much scope for research into the *data mining* of Concurrent Engineering data. Data mining has been described as knowledge discovery in databases and a "process of nontrivial extraction of implicit, previously unknown and potentially useful information," (Chen et al. 1996). CE projects generate substantial amounts of information whose structure and inter-relationships can be extremely difficult for humans to process and understand. Data mining techniques could therefore be a useful tool in analysing past CE projects.

[31] Object replication is a feature commonly found in commercial object database systems, (see for example the

9.6 Concluding remarks

Technology has been an enabling factor in the success of many CE projects during the past decade. For global organisations to effectively collaborate and share information, they need to utilise common tools and adhere to a prescribed framework. This framework must be open and flexible to allow it to evolve at the same pace as the organisation. This thesis addresses the unique needs of CE support environments in a consistent way using object-oriented methods by providing four key building blocks; (1) the CE-RM: a reference model for building CE support systems; (2) CONCERT: a middleware-based support architecture; (3) an information system object model for CE; and (4) a reference platform implementation written in the Java language. These components serve to advance the body of knowledge in the field of CE research. It is the author's belief that environments such as CONCERT proposed in this thesis are valuable tools that can aid the CE process and serve as useful infrastructures for building the next generation of support systems.

The philosophy of Concurrent Engineering has been extensively applied to the fields of engineering design and manufacturing; many of the sources of literature cited in this thesis stem from these domains. However, the work proposed in this thesis has also highlighted a number of additional domains that have many parallels with the ethos of CE including education, office integration, teleworking, and global group collaboration. This thesis has drawn on many aspects of these multidisciplinary areas to provide a platform for wide-ranging integration and support for Concurrent Engineering.

Itasca ODBMS, IBEX Corporation 1995).

REFERENCES

Adams, D. A., Ferguson, C.-J., and Irgens, C. (1995). "Information Provision to the Concurrent Engineering Environment," in *Proceedings of 11th International Conference on Computer-Aided Production Engineering*, The Institution of Mechanical Engineers, London, The Institution of Mechanical Engineers, pp. 29-34.

Aiken, M., Krosp, J., Shirani, A., and Martin, J. (1994). "Electronic Brainstorming in Small and Large Groups," *Information & Management*, 27(3), pp. 141-149.

Amundsen, M., and Hutchison, K. K. (1990). "Concurrent Engineering Environment for Electronic Circuit Design," in *Proceedings of Second National Symposium on Concurrent Engineering*, Morgantown, WV, pp. 60-75.

Andrews, T. A., and Krieger, D. (1993). "Concurrency Control for Workgroups," *Object Magazine*, 2, pp. 38-45.

Anklesaria, F., McCahill, M., Lindner, P., Johnson, D., Torrey, D., and Alberti, B. (1993). "The Internet Gopher Protocol: Internet RFC 1436," available on the World-Wide Web at the URL <ftp://ds.internic.net/rfc/rfc1436.txt>.

ANSA. (1989). *The ANSA Reference Manual Release 1.00*, A.P.M. Cambridge Ltd, Poseiden House, Castle Park, Cambridge, CB3 0RD, UK.

Anupam, V., and Bajaj, C. (1994). "Shastra - an Architecture For Development Of Collaborative Applications," *International Journal Of Intelligent & Cooperative Information Systems*, 3(2), pp. 155-172.

Apple, IBM, Netscape, Oracle, and Sun. (1996). "NC Reference Profile," available on the World-Wide Web at the URL <http://www.nc.ihost.com/nc_ref_profile.html>.

Arnold, K., and Gosling, J. (1996). *The Java(tm) Programming Language*, Addison-Wesley, Reading, MA.

Autodesk Inc. (1997). "AutoCAD homepage," available on the World-Wide Web at the URL <http://www.autodesk.com/>.

Bahler, B., Dupont, C., and Bowen, J. (1994a). "An Axiomatic Approach That Supports Negotiated Resolution of Design Conflicts in Concurrent Engineering," in *Artificial Intelligence in Design '94*, J. S. Gero and F. Sudweeks, eds., Kluwer Academic Publishers, New York, pp. 363-379.

Bahler, D., Dupont, C., and Bowen, J. (1994b). "Mediating Conflict In Concurrent Engineering With a Protocol Based On Utility," *Concurrent Engineering: Research and Applications*, 2(3), pp. 197-207.

Baran, N. (1995). "Special Report: Internet & Beyond," BYTE, July, pp. 68-86.

Barrett, S., and Tangney, B. (1995). "Aspects - Composing CSCW Applications," in *Proceedings of OOIS'95 - The 1995 International Conference on Object-Oriented Information Systems*, Dublin, Ireland, J. Murphy and B. Stone, eds., Springer-Verlag, pp. 51-56.

BCS Security Committee. (1995). "Internet Security," *The BCS Computer Bulletin*, 7(6), pp. 16-17.

Bell, G., Parisi, A., and Pesce, M. (1995). "The Virtual Reality Modeling Language: Version 1.0 Specification," available on the World-Wide Web at the URL <http://vrml.wired.com/vrml.tech/vrml10-3.html>.

Bentley, R., Busbach, U., and Sikkel, K. (1996). "The Architecture of the BSCW Shared Workspace System," in *Proceedings of 5th ERCIM/W4G Workshop of CSCW and the Web*, Sankt Augustin, GMD, pp. 31-42.

Bentley, R., Horstmann, T., Sikkel, K., and Trevor, J. (1995). "Supporting Collaborative Information Sharing with the World Wide Web: The BSCW Shared Workspace System," in *Proceedings of the 4th International World-Wide Web Conference*, Boston, MA, O'Reilly & Associates, pp. 63-74.

Berners Lee, T., Cailliau, R., Luotonen, A., Nielsen, H. F., and Secret, A. (1994). "The World-Wide Web," *Communications of the ACM*, 37(8), pp. 76-82.

Bernstein, P. A. (1996). "Middleware: A Model for Distributed System Services," *Communications of the ACM*, 39(2), pp. 86-98.

Birrell, A. D., and Nelson, B. J. (1984). "Implementing Remote Procedure Calls," *ACM Transactions of Computer Systems*, 2, pp. 39-59.

Booch, G. (1991). *Object-Oriented Design with Applications*, Benjamin Cummings, Menlo Park, CA.

Bounab, M., Derniame, J., Godart, C., and Morel, G. (1993). "DMMS: A PCTE Based Manufacturing Environment," in *Proceedings of PCTE'93 Conference*, Paris, France, I. Campbell, ed. Syntagma Systems Literature, pp. 431-449.

Bretl, R., Maier, D., Otis, A., Penney, D. J., Schuchardt, B., Stein, J., Williams, E. H., and Williams, M. (1989). "The GemStone Data Management System," in *Object-Oriented Concepts, Databases, and Applications*, W. Kim and F. H. Lochovsky, eds., ACM Press and Addison-Wesley, pp. 283-308.

BT Laboratories. (1995). "An Overview of Teleworking," available on the World-Wide Web at the URL <http://www.labs.bt.com/innovate/telework/reports/contents/anover.htm>.

Callahan, J. R., Ramakrishnan, S., and Wei, S. (1995). "Web Integrated Software Environment Home Page (WISE)," available on the World-Wide Web at the URL <http://research.ivv.nasa.gov/projects/WISE/index.html>.

Case, M. P., and Lu, S. C. Y. (1996). "Discourse Model For Collaborative Design," *Computer-Aided Design*, 28(5), pp. 333-345.

Cattell, R. G. G. (1994). *Object Data Management: Object-Oriented and Extended Relational Database Systems (Revised Edition)*, Addison-Wesley, Reading, MA.

Cattell, R. G. G., Barry, D., Bartels, D., Berler, M., Eastman, J., Gamerman, S., Jordan, D., Springer, A., Strickland, H., and Wade, D., eds. (1997). *The Object Database Standard: ODMG 2.0*, Morgan Kaufmann Publishers, San Francisco, CA.

Chen, M.-S., Han, J., and Yu, P. S. (1996). "Data Mining: An Overview From a Database Perspective," *IEEE Transactions On Knowledge and Data Engineering*, 8(6), pp. 866-922.

Chen, P. P.-S. (1976). "The Entitiy Relationship Model - Toward a Unified View of Data," *ACM Transactions on Database Systems*, 1(1), pp. 9-36.

Cleetus, J., and Usjio, W. (1989). *The Red Book of Functional Specifications For The DICE Architecture*, Concurrent Engineering Research Center, Morgantown, WV, USA.

Cleetus, K. J. (1992). "Definition of Concurrent Engineering," Report Number CERC-TR-RN-92-003, Concurrent Engineering Research Center, West Virginia University, Morgantown, WV.

Clegg, B. (1997). "Ringing in the Changes," PCWEEK, 3 June 1997, pp. 28-29.

Coad, P., and Yourdon, E. (1991a). *Object-Oriented Analysis*, Prentice Hall, Englewood Cliffs, NJ.

Coad, P., and Yourdon, E. (1991b). *Object-Oriented Design*, Yourdon Press.

Codd, E. F. (1970). "A Relational Model of Data for Large Shared Data Banks," *Communications of the ACM*, 13(6), pp. 377-387.

Coleman, D., Arnold, P., Bodoff, S., Dollin, C., Gilchrist, H., Hayes, F., and Jeremaes, P. (1994). *Object-Oriented Development: The Fusion Method*, Prentice Hall, New Jersey.

Colquhoun, G. J., Baines, R. W., and Crossley, R. (1996). "A Composite Behavioural Approach to Manufacturing Systems Modelling," *International Journal of Computer Integrated Manufacturing*, 9(4), pp. 327-338.

Coulouris, G., Dollimore, J., and Kindberg, T. (1994). *Distributed Systems: Concepts and Design*, (2nd ed), Addison-Wesley Publishing Company, Wokingham, England.

Coulson, G. (1993). "Multimedia Application Support in Open Distributed Systems," PhD Thesis, Computing Department, Lancaster University, UK.

Coupland, J. W. (1992). *Concurrent Engineering: An Information Pack*, IEE Technical Information Unit, London.

Curtis, D. (1997). "Java, RMI and CORBA: A White Paper," available on the World-Wide Web at the URL <http://www.omg.org/news/wpjava.htm>.

Cutkosky, M. R., Englemore, R. S., Fikes, R. E., Genesereth, M. R., Gruber, T. R., Mark, W. S., Tenenbaum, J. M., and Weber, J. C. (1993). "PACT: An Experiment in Integrating Concurrent Engineering Systems," *IEEE Computer*, 26(1), pp. 28-37.

Cutkosky, M. R., Tenenbaum, J. M., and Glicksman, J. (1996). "Madefast: Collaborative Engineering over the Internet," *Communications of the ACM*, 39(9), pp. 78-87.

Davis, W. J., and Jones, A. T. (1989). "A Functional Approach to Designing Architectures for CIM," *IEEE Transactions on Systems, Man and Cybernetics*, 19(2), pp. 164-173.

Deitz, D. (1995). "Service Bureaus For Concurrent Engineering," *Mechanical Engineering*, 117(10), pp. 20.

Doumeingts, G., Chen, D., and Marcott, F. (1992). "Concepts, models and methods for the design of production management systems," *Computers in Industry*, 19, pp. 89-111.

Dowlatshahi, S. (1994). "A Comparison Of Approaches to Concurrent Engineering," *International Journal Of Advanced Manufacturing Technology*, 9(2), pp. 106-113.

Drucker, P. F. (1991). "The New Productivity Challenge," *Harvard Business Review*, November-December, pp. 69-79.

Easterbrook, S., Finkelstein, A., Kramer, J., and Nuseibeh, B. (1994). "Coordinating Distributed Viewpoints: The Anatomy of a Consistency Check," *Concurrent Engineering: Research and Applications*, 2(3), pp. 209-222.

Ellsworth, J. H., and Ellsworth, M. V. (1994). *The Internet Business Book*, John Wiley & Sons, New York.

Eòlas Technologies. (1995). "Eòlas Technologies Home Page," available on the World-Wide Web at the URL <http://www.eolas.com/>.

Erkes, J. W., Kenny, K. B., Lewis, J. W., Sarachan, B. D., Sobolewski, M. W., and Sum, R. N., Jr. (1996). "Implementing Shared Manufacturing Services on the World-Wide Web," *Communications of the ACM*, 39(2), pp. 34-45.

ESPRIT Consortium AMICE. (1993). *CIMOSA: Open System Architecture for CIM, Project 688/5288*, vol. 1, (2nd revised and extended ed), Springer-Verlag, Berlin.

Evans, B. (1988). "Simultaneous Engineering," *Mechanical Engineering*, 110(2), pp. 38-39.

Farooqui, K., Logrippo, L., and de Meer, J. (1995). "The ISO Reference Model for Open Distributed Processing: an Introduction," *Computer Networks and ISDN Systems*, 27(8), pp. 1215-1229.

Fielding, R., Gettys, J., Mogul, J., Frystyk, H., and Berners-Lee, T. (1997). "Hypertext Transfer Protocol - HTTP/1.1," available on the World-Wide Web at the URL <http://www.w3.org/Protocols/rfc2068/rfc2068>.

Finger, S., Stivoric, J., Amon, C., Gursoz, L., Prinz, F., Siewiorek, D., Smailagic, A., and Weiss, L. (1996). "Reflections On a Concurrent Design Methodology - A Case-Study In Wearable Computer Design," *Computer-Aided Design*, 28(5), pp. 393-404.

Finkelstein, A., Kramer, J., Nuseibeh, B., Finkelstein, L., and Goedicke, M. (1992). "Viewpoints: A Framework for Integrating Multiple Perspectives in System Development," *International Journal of Software Engineering and Knowledge Engineering*, 2(1), pp. 31-58.

Firesmith, D. G. (1993). *Object-Oriented Requirements Analysis and Logical Design: A Software Engineering Approach*, Wiley, New York.

Garnett, L. (1997). "Wrapping Objects," *Journal of Object-Oriented Programming*, 9(8), pp. 38-43.

Gatenby, D. A., and Foo, G. (1990). "Design for X (DFX): Key to Competitive, Profitable Products," *AT&T Technical Journal*, 69(3), pp. 2-13.

Gatenby, D. A., Lee, P. M., Howard, R. E., Hushyar, K., Layendecker, R., and Wesner, J. (1994). "Concurrent Engineering - An Enabler For Fast, High-Quality Product Realization," *AT&T Technical Journal*, 73(1), pp. 34-47.

Gessler, S., and Kotulla, A. (1995). "PDAs as Mobile WWW Browsers," *Computer Networks and ISDN Systems*, 28(1&2), pp. 53-59.

Glicksman, J., Kramer, G. A., and Mayer, N. P. (1994). "Internet Publishing via the World-Wide Web," in *Proceedings of Groupware '94*, San Jose, California, pp. 431-442.

Goldberg, A. (1984). *Smalltalk-80: The Interactive Programming Environment*, Addison-Wesley, Reading, MA.

Goldschmidt, A. (1996). "Technical Opinion: Report on NIIIP," *Communications of the ACM*, 39(3), pp. 100-103.

Goldstein, D. (1994). "An Agent-Based Architecture For Concurrent Engineering," *Concurrent Engineering: Research and Applications*, 2(2), pp. 117-123.

Golfin, N. G., and Jackson, M. (1994). "Groupware Trial In BT," *BT Technology Journal*, 12(3), pp. 51-55.

Goscinski, A. (1991). *Distributed Operating Systems: The Logical Design*, Addison-Wesley, Sydney.

Gosling, J., and McGilton, H. (1995). "The Java Language Environment," White Paper, Sun Microsystems Inc., Mountain View, CA.

Graefe, U., and Thomson, V. (1989). "A Reference Model for Production Control," *International Journal of Computer Integrated Manufacturing*, 2(2), pp. 86-93.

Gray, N. A. B. (1994). *Programing with Class: A Practical Introduction to Object-Oriented Programming with C++*, John Wiley & Sons, Chichester.

Greenberg, S. (1991a). "An Annotated Bibliography of Computer Supported Cooperative Work," SIGCHI Bulletin, 23(3), pp. 29-62.

Greenberg, S. (1991b). "Personalizable Groupware: Accomodating Individual Roles and Group Differences," Research Report, Report Number 90/404/28, Dept. of Computer Science, University of Calgary, Alberta, Canada.

Gruia-Catalin, R. (1985). "A Taxonomy of Current Issues in Requirements Specification," *Computer*, 18(4), pp. 14-21.

Gupta, S. M., and Brennan, L. (1995). "Implementation Of Just-In-Time Methodology In a Small Company," *Production Planning & Control*, 6(4), pp. 358-364.

Hammer, M., and Champy, J. (1993). *Reengineering the Corporation: A Manifesto for Business Revolution*, Harper Collins, Inc.

Hanneghan, M., Merabti, M., and Colquhoun, G. (1995). "The Design Of An Object-Oriented Repository To Support Concurrent Engineering," in *Proceedings of OOIS'95 - The 1995 International Conference on Object-Oriented Information Systems*, Dublin, Ireland, J. Murphy and B. Stone, eds., Springer-Verlag, pp. 200-215.

Hanneghan, M., Heß, P., Hubel, H., Merabti, M., and Colquhoun, G. (1996a). "Concurrent Engineering Effizient Nutzen - Unterstützende Ansätze aus der Datenverwaltung *(Using Concurrent Engineering Effectively - Supporting Approaches from Data Management)*," *VDI-Z Integrierte Produktion*, 138(3), pp. 51-54.

Hanneghan, M., Merabti, M., and Colquhoun, G. (1996b). "The World-Wide Web as a Platform for Supporting Interactive Concurrent Engineering," in *Proceedings of Advanced Information Systems Engineering - 8th International Conference, CAiSE'96*, Heraklion, Crete, Greece, P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, eds., Springer-Verlag, Lecture Notes in Computer Science, 1080, pp. 301-318.

Hanneghan, M., Colquhoun, G., and Merabti, M. (1997). "A Scalable Intranet-Hosted Support Environment For Concurrent Engineering," in *Proceedings of IEE Colloquium on Internet Technology and the Integrated Enterprise*, Savoy Place, London, J. Boardman, ed. Institution of Electrical Engineers, Digest No: 97/149, pp. 9/1-9/4.

Hanneghan, M., Merabti, M., and Colquhoun, G. (1998). "A Viewpoint Analysis Reference Model for Concurrent Engineering," *to appear in Computers in Industry: An International Journal.*

Hanneghan, M., Merabti, M., Colquhoun, G., and Mills, B. (1996c). "Computer-Aided Concurrent Engineering Using The Internet," poster presented at Graduate School Promotion Launch, Liverpool John Moores University, Liverpool.

Harding, J. A., and Popplewell, K. (1996). "Driving Concurrency In a Distributed Concurrent Engineering Project Team - A Specification For an Engineering Moderator," *International Journal Of Production Research*, 34(3), pp. 841-861.

Hardwick, M., Downie, B. R., Kutcher, M., and Spooner, D. L. (1995). "Concurrent Engineering with Delta Files," *IEEE Computer Graphics and Applications*, 15(1), pp. 62-68.

Harrington, J., Jr. (1973). *Computer Integrated Manufacturing*, (Reprint 1979 ed), Robert E. Krieger Publishing Co., New York.

Hayes, F. (1992). "The Groupware Dilema," UnixWorld, 2, pp. 46-50.

Hoare, C. A. R. (1978). "Communicating Sequential Processes," *Communications of the ACM*, 21(8), pp. 666-677.

Hobby, J. (1997). "Driving Ambition," Computing, 5th June 1997, pp. 50-51.

Hori, M., Shinoda, Y., and Ochimizu, K. (1996). "Shared Data Management Mechanism for Distributed Software Development Based on a Reflective Object-Oriented Model," in *Proceedings of Advanced Information Systems Engineering - 8th International Conference, CAiSE'96*, Heraklion, Crete, Greece, P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, eds., Springer-Verlag, Lecture Notes in Computer Science, 1080, pp. 362-382.

IBEX Corporation. (1995). "ITASCA™ Distributed Object Database Management System: Technical Summary Release 2.3.5," IBEX Corporation, SA, Archamps, France.

IBM. (1987). *Computer Integrated Manufacturing: the IBM Experience*, Findlay Publications (for IBM UK Scientific and Industrial centre), Horton, Kent.

ISO. (1986). "The Ottawa Report on Reference Models for Manufacturing Standards, Version 1.1," Report Number ISO TC184/SC5/WG1, International Standards Organisation.

ISO. (1992a). "Draft Recommendation X.903: Basic Reference Model of Open Distributed Processing," Report Number ISO/IEC JTC1/SC21 N7055, International Standards Organisation.

ISO. (1992b). "Industrial Automation Systems - Product Data Representation and Exchange - Part 1: Overview and Fundamental Principles," Report Number ISO CD 10303-1, International Standards Organisation.

ISO. (1993). "Industrial Automation Systems - Product Data Representation and Exchange - Part 11: Descriptive Methods: The Express Language Reference Model," Report Number ISO CD 10303-11, International Standards Organisation.

Jager, M., Osterfeld, U., Ackerman, H. J., and Hornung, C. (1993). "Building a Multimedia ISDN PC," *IEEE Computer Graphics and Applications*, 13(5), pp. 24-33.

Jasnoch, U., Kress, H., Schroeder, K., and Ungerer, M. (1994). "The CoConut Environment," Technical Report, Report Number FIGD-94i001, Fraunhofer-Institut für Graphische Datenverarbeitung, Darmstadt, Germany.

Jo, H. H., Parsaei, H. R., and Sullivan, W. G. (1993). "Principles of Concurrent Engineering," in *Concurrent Engineering: Contemporary issues and modern design tools*, H. R. Parsaei and W. G. Sullivan, eds., Chapman & Hall, pp. 3-23.

Jo, H. H., Parsaei, H. R., and Wong, J. P. (1991). "Concurrent Engineering: The Manufacturing Philosophy for the 90's," *Computers & Industrial Engineering*, 21(1-4), pp. 35-39.

Johnson, E. (1997). "LinguaNet Project Summary," available on the World-Wide Web at the URL <http://www2.echo.lu/langeng/en/le1/linguanet/summary.html>.

Johnson, R., and Kleiner, B. H. (1995). "Technical Report - New Developments In Total Quality Management," *International Journal Of Vehicle Design*, 16(2-3), pp. 282-291.

Jorysz, H. R., and Vernadat, F. B. (1990). "CIM-OSA Part 1: Total Enterprise Modelling and Function View," *International Journal of Computer Integrated Manufacturing*, 3(3-4), pp. 144-156.

Kannan, R., Reddy, Y. V., and Cleetus, K. J. (1992). "Support Environment for Network Computing," Technical Report, Report Number CERC-TR-RN-91-007, Concurrent Engineering Research Center, West Virginia University, Morgantown, WV.

Kantor, B., and Lapsley, P. (1986). "Network News Transfer Protocol: Internet RFC 977," available on the World-Wide Web at the URL <ftp://ds.internic.net/rfc/rfc977.txt>.

Keim, D. A., and Kriegel, H.-P. (1996). "Visualization Techniques for Mining Large Databases: A Comparison," *IEEE Transactions On Knowledge and Data Engineering*, 8(6), pp. 923-938.

Kernighan, B., and Ritchie, D. (1988). *The C Programming Language*, (2nd ed), Prentice Hall, Englewood Cliffs, NJ.

Kim, W., Ballou, N., Chou, H.-T., Garza, J. F., and Woelk:, D. (1989). "Features of the ORION Object-Oriented Database System," in *Object-Oriented Concepts, Databases, and Applications*, W. Kim and F. H. Lochovsky, eds., ACM Press and Addison-Wesley, pp. 251-282.

Kim, W., and Lochovsky, F. H., eds. (1989). *Object-Oriented Concepts, Databases, and Applications*, ACM Press and Addison-Wesley.

Kirsche, T., Lenz, R., Luhrsen, H., Meyerwegener, K., Bever, M., Schaffer, U., and Schottmuller, C. (1993). "Communication Support for Cooperative Work," *Computer Communications*, 16(9), pp. 594-602.

Klein, M. (1996). "Core Services for Coordination in Concurrent Engineering," *Computers in Industry*, 29(1-2), pp. 105-115.

Klittich, M. (1990). "CIM-OSA Part 3: CIM-OSA Integrating Infrastructure - the Operational Basis for Integrated Manufacturing Systems," *International Journal of Computer Integrated Manufacturing*, 3(3-4), pp. 168-180.

Koonce, D. A. (1995). "Information Model Level Integration for CIM Systems - A Unified Database Approach to Concurrent Engineering," *Computers & Industrial Engineering*, 29, pp. 647-651.

Kotonya, G., and Sommerville, I. (1992). "Viewpoints For Requirements Definition," *Software Engineering Journal*, 7(6), pp. 375-387.

Kotonya, G., and Sommerville, I. (1996). "Requirements Engineering with Viewpoints," *Software Engineering Journal*, 11(1), pp. 5-18.

Kress, H. (1996). "Integration Factors within a Virtual Prototyping Environment," *International Journal of Flexible Automation and Integrated Manufacturing*, 4(1), pp. 29-41.

Lettice, F., Smart, P., and Evans, S. (1995). "A Workbook-Based Methodology For Implementing Concurrent Engineering," *International Journal Of Industrial Ergonomics*, 16(4-6), pp. 339-351.

Lewis, J. W., Bernstein, B. M., Kenny, K. B., Mohr, G., Ostrowski, M. C., Sarachan, B. D., and Sum, R. N. (1994). "The Concurrent Engineering Toolkit: a Network Agent for Manufacturing Cycle Time Reduction," in *Proceedings of CE'94: Concurrent Engineering Research and Applications*, Pittsburgh, PA, A. J. Paul and M. Sobolewski, eds., Concurrent Technologies Corporation, pp. 521-528.

Londono, F., Cleetus, K. J., Nichols, D. M., Iyer, S., Karandikar, H. M., Reddy, S. M., Potnis, S. M., Massey, B., Reddy, A., and Ganti, V. (1992). "Coordinating a Virtual Team," Technical Report, Report Number CERC-TR-RN-92-005, Concurrent Engineering Research Center, West Virginia University, Morgantown, WV.

Loomis, M. E. S. (1995). *Object Databases: The Essentials*, Addison-Wesley, Reading, MA.

Los, R., Beziau, D., Brodda, J., and Marzi, J. (1992). "Development and use of an Integrated Reference Model for one-of-a-kind Construction Site Manufacturing," in *'One-of-a-kind' Production: New Approaches*, B. E. Hischand and K. D. Thoben, eds., Elsevier, North Holland, pp. 329-340.

Lotus Corporation. (1993). Lotus Notes. Lotus Corporation, Lotus Park, The Causeway, Staines, Middlesex, TW18 3AG, UK.

Lucent Technologies. (1996). "The Inferno Network Operating System and Programming Environment," available on the World-Wide Web at the URL <http://inferno.lucent.com/inferno/index.html>.

Mandviwalla, M., and Olfman, L. (1994). "What Do Groups Need? A Proposed Set of Generic Groupware Requirements," *ACM Transactions on Computer-Human Interaction*, 1(3), pp. 245-268.

Massey, J. (1996). "Video Nation," *Computing*, 25th January, pp. 20-21.

McCool, R. (1994). "CGI Specification," available on the World-Wide Web at the URL <http://hoohoo.ncsa.uiuc.edu/cgi/>.

McKnight, S., and Jackson, J. K. (1989). "Simultaneous Engineering Saves Manufacturing Lead Time, Costs and Frustration," *Industrial Engineering*, 21(1), pp. 25-27.

Merabti, M., and Carew, M. (1994). "The COMBINE: A Distributed Support Environment for Concurrent Engineering," in *Proceedings of 1994 Engineering Systems Design and Analysis Conference (ESDA)*, London, M. M. Tanik, A. Ertas, and I. I. Esat, eds., 64-5, pp. 89-96.

Microsoft Corporation. (1997). "Microsoft VBScript," available on the World-Wide Web at the URL <http://www.microsoft.com/vbscript/>.

Mills, J. J., Graham, J. K., Elmasri, R. A., and Weems, B. P. (1993). "The Virtual Manufacturing Workbench - Representation and Interface Issues," *IFIP Transactions B-Applications In Technology*, B-10, pp. 231-244.

Molina, A., Alashaab, A. H., Ellis, T. I. A., Young, R. I. M., and Bell, R. (1995a). "A Review Of Computer-Aided Simultaneous Engineering Systems," *Research In Engineering Design - Theory Applications and Concurrent Engineering*, 7(1), pp. 38-63.

Molina, A., Ellis, T. I. A., Young, R. I. M., and Bell, R. (1995b). "Modelling Manufacturing Capability to Support Concurrent Engineering," *Concurrent Engineering: Research and Applications*, 3(1), pp. 29-42.

Mullery, G. P. (1979). "CORE - A Method for Controlled Requirements Specification," in *Proceedings of 4th International Conference on Software Engineering*, Munich, Germany, IEEE Computer Society, pp. 126-135.

Netscape Communications Corporation. (1996). "JavaScript Authoring Guide," available on the World-Wide Web at the URL <http://www.netscape.com/eng/mozilla/3.0/handbook/javascript/>.

Network Wizards. (1996). "Internet Domain Survey," available on the World-Wide Web at the URL <http://www.nw.com/>.

Nickerson, D. P. (1990). "Introduction to Concurrent Engineering in Design," in *Proceedings of Electro/90 Conference*, Boston MA, pp. 174-179.

NIIIP Consortium. (1995). "NIIIP Reference Architecture: Concepts and Guidelines," Report Number NTR95-01, National Industrial Information Infrastructure Protocols Consortium, (*available on the WWW at* http://www.niiip.org/).

Norrie, M. C., and Wunderli, M. (1996). "Agent-Based Tool Integration for Distributed Information Systems," in *Proceedings of Advanced Information Systems Engineering - 8th International Conference, CAiSE'96*, Heraklion, Crete, Greece, P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, eds., Springer-Verlag, Lecture Notes in Computer Science, 1080, pp. 383-401.

Nunamaker, J. F., Dennis, A. R., Valacich, J. S., Vogel, D. R., and George, J. F. (1991). "Electronic Meeting Systems to Support Group Work," *Communications of the ACM*, 34(7), pp. 40-61.

Object Management Group. (1995a). "The Common Object Request Broker: Architecture and Specification (Revision 2.0)," Technical Document, Report Number PTC/96-03-04, The Object Management Group, Framingham, MA.

Object Management Group. (1995b). "*CORBAfacilities:* Common Facilities Architecture (revision 4.0)," Report Number 97-06-15, The Object Management Group, Framingham, MA.

Object Management Group. (1997). "CORBAservices: Common Object Services Specification," Report Number 97-06-01, The Object Management Group, Framingham, MA.

Open Software Foundation. (1993). *OSF DCE Application Development Guide (Rev. 1.0)*, Prentice Hall, Englewood Cliffs, NJ.

Ousterhout, J. K. (1994). *Tcl and the Tk Toolkit*, Addison Wesley Publishing Company.

Ousterhout, J. K. (1997). "Scripting: Higher Level Programming for the 21st Century," White Paper, Sun Microsystems Laboratories, Mountain View, CA.

Parametric Technology Corporation. (1993). *Pro/ENGINEER User Manuals*, Parametric Technology Corporation, 128 Technology Drive, Waltham, MA 02154.

Petrie, C. J. (1996). "Agent-Based Engineering, the Web, and Intelligence," *IEEE Expert - Intelligent Systems & Their Applications*, 11(6), pp. 24-29.

Postel, J., and Reynolds, J. (1985). "File Transfer Protocol (FTP): Internet RFC 959," available on the World-Wide Web at the URL <ftp://ds.internic.net/rfc/rfc959.txt>.

Powell, D. (1996). "Group Communication," *Communications of the ACM*, 39(4), pp. 50-53.

Prasad, B. (1996). *Concurrent Engineering Fundamentals: Integrated Product and Process Organization*, vol. 1, Prentice Hall, New Jersey.

Pressman, R. S. (1997). *Software Engineering: A Practitioner's Approach (European adaptation)*, (4th ed), McGraw-Hill, Inc., New York.

Raggett, D. (1997). "HTML 3.2 Reference Specification: W3C Recommendation 14-Jan-1997," available on the World-Wide Web at the URL <http://www.w3.org/TR/REC-html32.html>.

Rochkind, M. J. (1975). "The Source Code Control System," *IEEE Transactions on Software Engineering*, SE-1(4), pp. 364-370.

Ross, D. T. (1977). "Structured Analysis (SA): A Language for Communicating Ideas," *IEEE Transactions on Software Engineering*, SE-3(1), pp. 16-24.

Royce, W. W. (1970). "Managing the Development of Large Software Systems: Concepts and Techniques," in *Proceedings of IEEE WESCON*, Los Angeles, CA, IEEE Press, pp. 1-9.

Santos, J. L. T., and Cardoso, J. B. (1993). "Computer Aided Mechanical Engineering Design Environment for Concurrent Design Process," in *Proceedings of the 1993 ITEC Workshop: Simulation in Concurrent Engineering*, Wembley, UK, R. Reddy and S. Medhat, eds., SCS, San Diego, pp. 71-83.

Scheer, A. W. (1992). "Architecture of Integrated Information Systems, ARIS," IDS Prof. Scheer GmbH, Saarbrücken.

Scheer, A. W. (1996). "Scheer Magazine Special," IDS Prof. Scheer GmbH, Saarbrücken.

Scheifler, R., and Gettys, J. (1992). *X Window System*, (3rd ed), Digital Press.

Schmitz, J., and Desa, S. (1993). "The Development of Virtual Concurrent Engineering and its Application to Design for Producibility," *Concurrent Engineering: Research and Applications*, 1(3), pp. 159-169.

Schneier, B. (1997). "Cryptography, Security and the Future," *Communications of the ACM*, 40(1), pp. 138.

Scrivener, S. A. R., Harris, D., Clark, S. M., Rockoff, T., and Smyth, M. (1995). "Designing at a Distance Via Real-Time Designer-to-Designer Interaction," in *Groupware for Real-Time Drawing: A Designer's Guide*, S. Greenberg, S. Hayne, and R. Rada, eds., McGraw-Hill, London, pp. 5-23.

Siemieniuch, C. E., and Sinclair, M. A. (1994). "Concurrent Engineering and CSCW: The Human Factor," in *Computer Support for Co-operative Work*, K. Spurr, P. Layzell, L. Jennison, and N. Richards, eds., John Wiley and Sons Ltd, Chichester, pp. 111-125.

Smith, B. (1994). "Blazing the Path: DEC's LinkWorks is an open design for multiplatform work flow," BYTE, August, pp. 147-150.

Smith, B., and Wellington, J. (1986). "Initial Graphics Exchange Specification (IGES) Version 3.0," Report Number NSBIR 86-3359, National Institute of Standards.

Smith, P. G. (1988). "Winning the New Products Rate Race," Machine Design, May 12.

Smith, R. P. (1997). "The Historical Roots of Concurrent Engineering Fundamentals," *IEEE Transactions On Engineering Management*, 44(1), pp. 67-78.

Sobolewski, M., and Erkes, J. (1995). "CAMnet - Architecture and Applications," available on the World-Wide Web at the URL <http://camnet.ge.com/camnet/papers/camnet-ce95.ps>.

Sohlenius, G. (1992). "Keynote Paper: Concurrent Engineering," *Annals of the CIRP*, 41(2), pp. 645-655.

Soley, R. M., ed. (1993). *Object Management Architecture Guide*, (2nd ed), The Object Management Group, Framingham, MA.

Sommerville, I. (1996). *Software Engineering*, (5th ed), Addison-Wesley Publishing Company, Wokingham, England.

Spurr, K., Layzell, P., Jennison, L., and Richards, N., eds. (1994). *Computer Support for Co-operative Work*, John Wiley & Sons, Chichester.

Srinivas, K., Reddy, R., Babadi, A., Kamana, S., Kumar, V., and Dai, Z. (1992). "MONET: A Multimedia System for Conferencing and Application Sharing in Distributed Systems," Report Number CERC-TR-RN-91-009, Concurrent Engineering Research Center, West Virginia University, Morgantown, WV.

Stonebraker, M., and Kemnitz, G. (1991). "The Postgres Next-Generation Database Management System," *Communications of the ACM*, 34(10), pp. 78-92.

Stonebraker, M., Rowe, L. A., and Hirohama, M. (1990). "The Implementation of POSTGRES," *IEEE Transactions On Knowledge and Data Engineering*, 2(1), pp. 125-142.

Stroustrup, B. (1991). *The C++ Programming Language*, (2nd ed), Addison Wesley, Reading, MA.

Tichy, W. F. (1985). "RCS - A System for Version Control," *Software - Practice and Experience*, 15(7), pp. 637-654.

Trapp, G., Lawson, M., and Burkett, B. (1992). "Workshop on Concurrent Engineering," in *Proceedings of 13th Annual Conference & Exposition: CAD (Computer-Aided Design) and Engineering 92 / Business Graphics 92*, Anaheim, CA, pp. 766-792.

US Air Force. (1981). "Integrated Computer-Aided Manufacturing (ICAM) Architecture Part II, Volume IV - Functional Modelling Manual (IDEF0)," Report Number AFWAL-TR-81-4023, Air Force Materials Laboratory, Wright-Patterson AFB, Ohio 45433.

Valenzano, A., Demartini, C., and Ciminiera, L. (1992). *MAP and TOP Communications: Standards and Applications*, Addison-Wesley, Wokingham, England.

Vinoski, S. (1997). "CORBA: Integrating Diverse Applications Within Distributed Heterogeneous Environments," *IEEE Communications Magazine*, 35(2), pp. 46-55.

Wade, A. E. (1995). "The ODBMS role in Distributed Client Server Computing," White Paper, Objectivity, Inc., Mountain View, CA.

Weston, R. H. (1995). "An Academic Perspective on Structured Methods and Tools for the Life-cycle Engineering of Information Systems," MSI Research Institute, Loughborough University of Technology, Loughborough.

Williams, M. J., and Taleb-Bendiab, A. (1997). "A Toolset for Architecture Independant, Reconfigurable Multi-Agent Systems," in *Proceedings of First International Workshop on Mobile Agents (MA '97)*, Berlin, K. Rothermel and Popescu-Zeletin, eds., Springer-Verlag, pp. 210-222.

Wingrove, S. J., Boardman, J. T., and Sagoo, J. S. (1997). "Development of a Concurrent Engineering Handbook Using Intranet Technology," in *Proceedings of IEE*

Colloquium on Internet Technology and the Integrated Enterprise, Savoy Place, London, J. Boardman, ed. Institution of Electrical Engineers, Digest No: 97/149, pp. 8/1-8/8.

Winner, K. E., Pennell, J. P., Bertrand, H. E., and Slusarzuk, M. M. G. (1988). "The Role of Concurrent Engineering in Weapons System Acquisition," Report Number R-338, Institute for Defense Analysis, Alexandria, Virginia.

Wu, P. L. (1995). "A Case-Study In FMS Production Planning and Dispatching," *International Journal Of Flexible Manufacturing Systems*, 7(4), pp. 361-372.

Yavuz, I. H., and Satir, A. (1995). "Kanban-Based Operational Planning and Control - Simulation Modeling," *Production Planning & Control*, 6(4), pp. 331-344.

Yenradee, P. (1994). "Application Of Optimized Production Technology In a Capacity Constrained Flow-Shop - a Case-Study In a Battery Factory," *Computers & Industrial Engineering*, 27(1-4), pp. 217-220.

Yeomans, R. W. (1987). "Design Rules and Development Guidelines for CIM Projects," in *Proceedings of 4th European Conference on Automated Manufacturing*, IFS Conferences Ltd, pp. 395-412.

Zachman, J. A. (1987). "A Framework for Information Systems Architecture," *IBM Systems Journal*, 26(3), pp. 276-292.

Zhang, H. C., and Zhang, D. (1995). "Concurrent Engineering: An Overview From Manufacturing Engineering Perspectives," *Concurrent Engineering: Research and Applications*, 3(3), pp. 221-236.

Zhang, H.-C., and Alting, L. (1992). "An Exploration of Simultaneous Engineering for Manufacturing Enterprises," *International Journal of Advanced Manufacturing Technology*, 7, pp. 101-108.

APPENDIX A: SUMMARY OF FUSION OBJECT MODEL NOTATION

The notation used in the models described in this thesis is the Fusion object modelling notation (Coleman et al. 1994) which was developed by Hewlett-Packard. The Fusion method is the result of an amalgamation of best practice from a number of popular object modelling notations was chosen for its clarity in presenting object models and the ease by which practitioners of other object modelling notations can interpret it.

Figure A1 provides a summary of the diagramming notation used to describe interfaces between classes of objects (as used in Appendix B). Figure A2 provides a summary of the Fusion notation used to develop the object models given in Appendix C.

Fusion system interface diagram notation

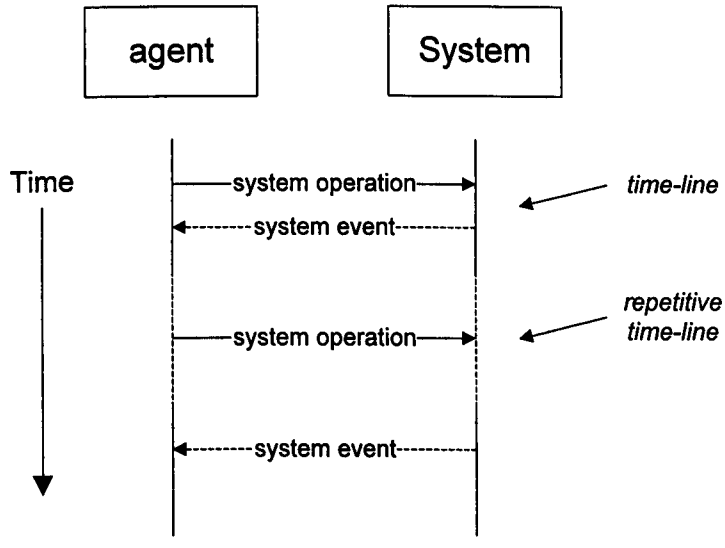


Figure A 1. Fusion system interface diagram notation.

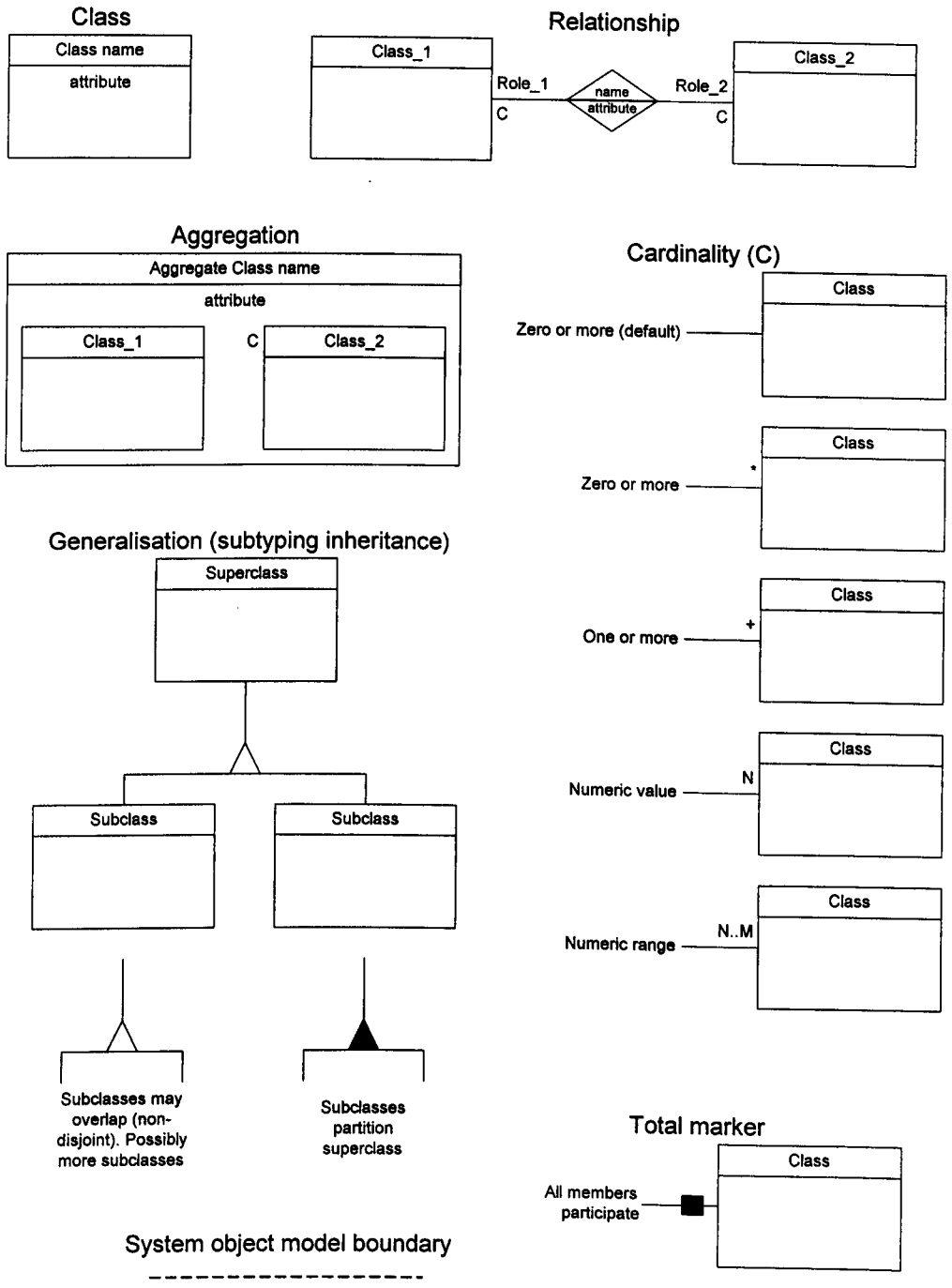
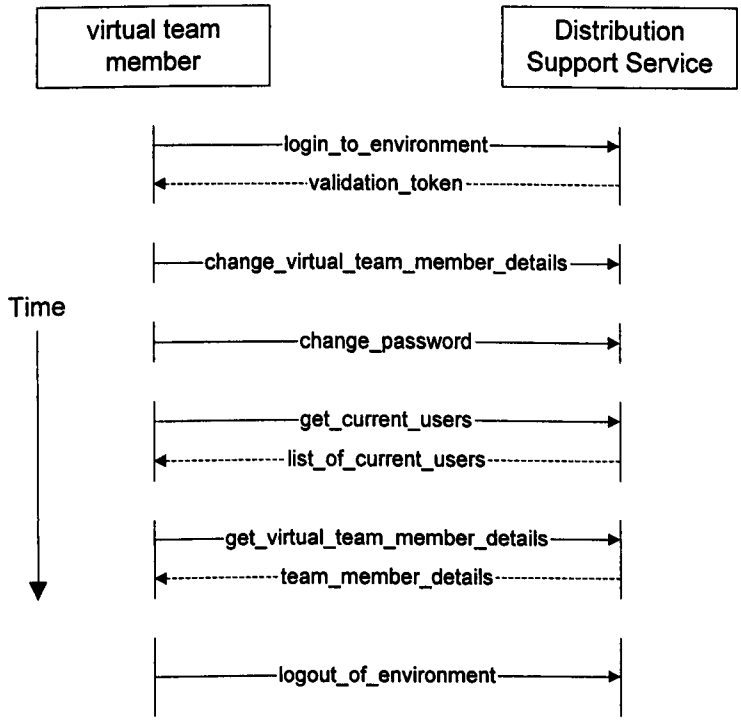


Figure A 2. Fusion object model notation.

B1. Introduction

This appendix provides a complete specification of the system interfaces of each of the four support services within the CONCERT architecture. This specification consists of the operations that can be requested of each service along with the corresponding return results. A summary of the Fusion object model notation used is given in Appendix A.

B2. Distribution support service



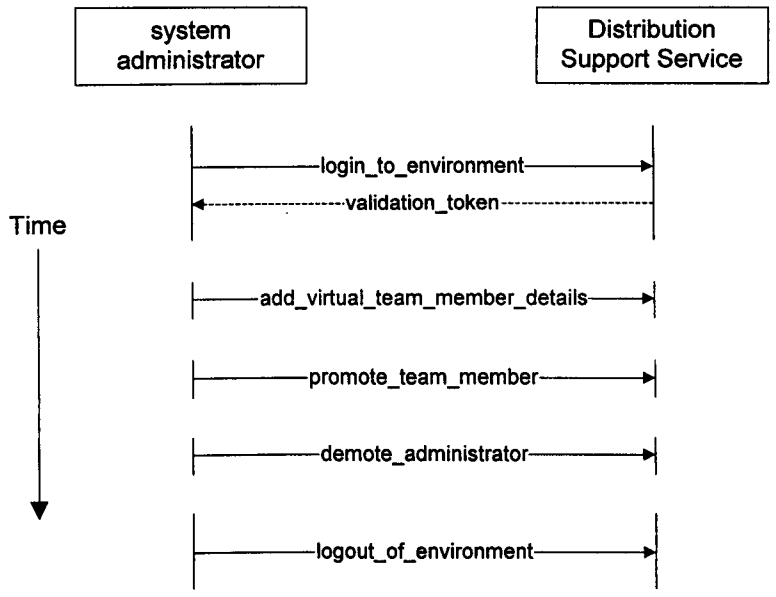
Description:

A team member will login to the environment via the Distribution Support Service (DSS). On doing this they will be issued with a validation token which must be supplied in every opreation request to any CONCERT support service for security reasons. The DSS will be called upon to validate this token by the other support services.

Once logged on successfully, the team member can issue requests to change his / her member details (contact information) or password. They can also find out who else is currently using the environment and retrieve their contact information.

Users must issue a logout request to end their session using the CONCERT environment.

Figure B 1. Distribution support service system interface — virtual team members.

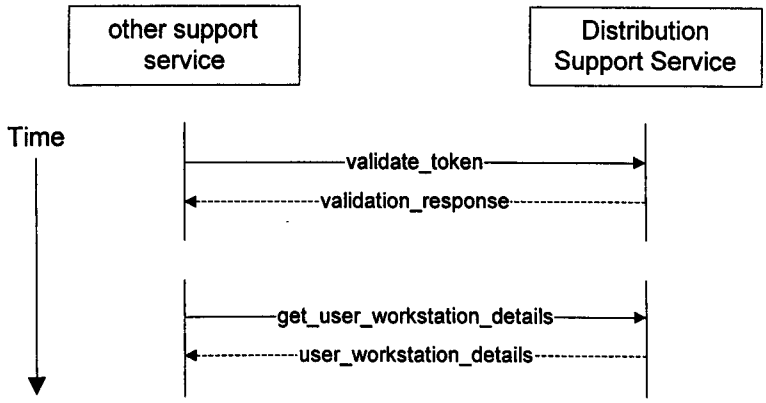


Description:

The CONCERT system administrator is a special class of virtual team member who can perform the same set of operations but who also has permission to perform system-level operations within the CONCERT environment.

Once logged on successfully, the administrator can issue requests to change not only their own member details (contact information) or password but other team members' details and passwords as well. Administrators can also add new virtual team members to the DSS user database and promote / demote existing team members to / from Administrator status.

Figure B 2. Distribution support service system interface — system administrators.



Description:

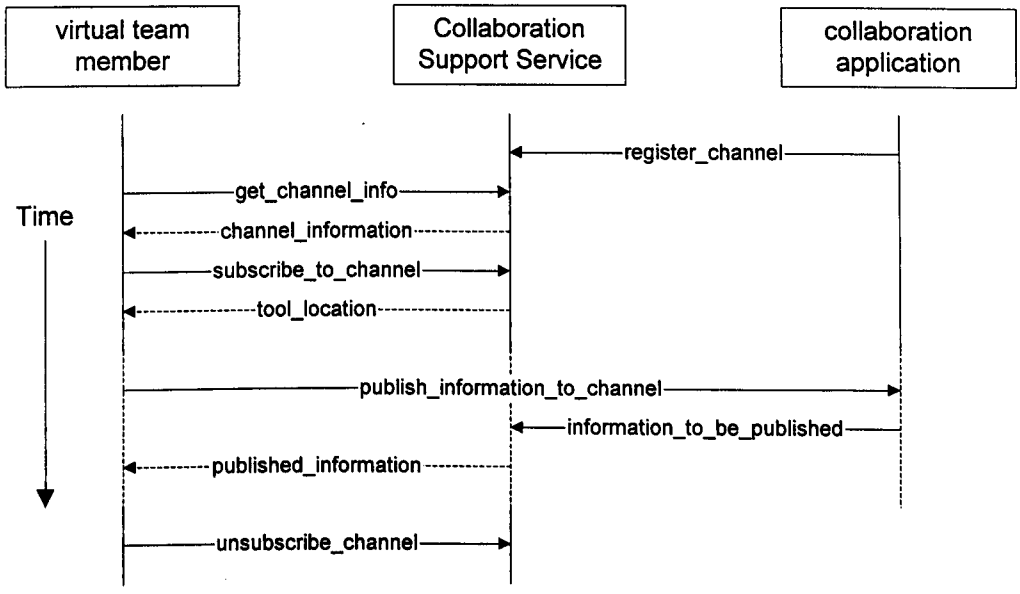
Other CONCERT support services will call upon the Distribution Support Service (DSS) to perform token validation (since every call to a method in any support service must involve validation of the caller). In response to a validation_token request, the DSS will issue a proceed or stop value.

Support services may occasionally need to know more specific details about a user currently logged into the environment such as their workstation type or network address. This can be obtained by a call to the DSS which maintains this information.

Figure B 3. Distribution support service system interface — requests from collaborating support services.

It should be noted that for security reasons, all support service requests must first be validated by the DSS (i.e. with a *validate_token* request). This is not shown in the following diagrams to aid clarity.

B3. Collaboration support service

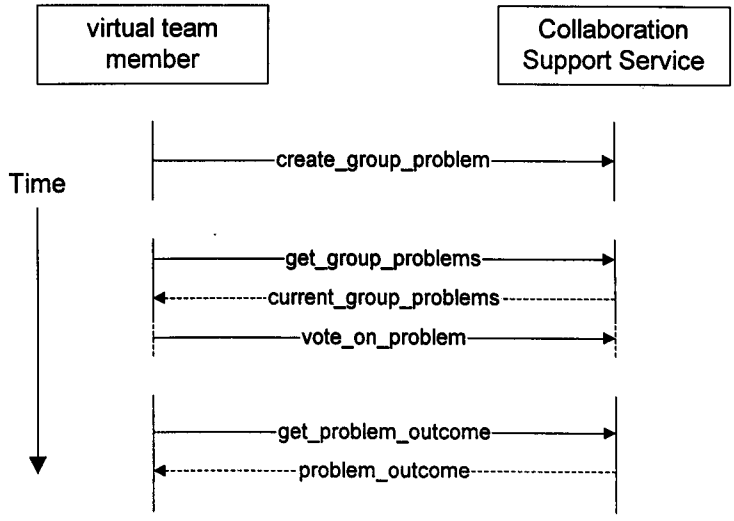


Description:

Collaboration applications that wish to use the publish and subscribe features of the CONCERT environment register themselves as a channel via the Collaboration Support Service (CSS). This allows virtual team members to poll the CSS to see which channels are currently available. When a virtual team member subscribes to a particular channel, the CSS will give the user the actual location of the collaboration tool so that they can interact directly with the application. The collaboration tool channels all events and messages via the CSS which in turn re-distributes the information to all subscribed virtual team members.

To end a channel session, the virtual team member informs the CSS that they wish to unsubscribe. This will stop the transmission of information to that user from that channel.

Figure B 4. Collaboration support service system interface — event and message channelling.

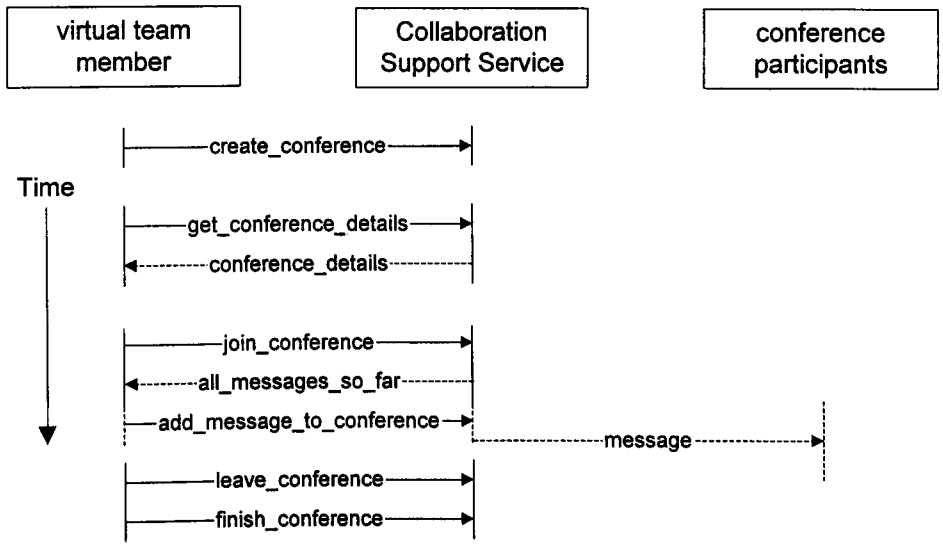


Description:

The Collaboration Support Service (CSS) allows virtual team members to perform gruop voting on problems that they encounter during a CE development process. First a team member creates (or proposes) and new group problem in the form of a statement. Other team members can then retrieve group problems via the CSS and register their vote (either in favour, against or abstain). Users can change their vote will the voting process is still active.

The outcome of a group problem can be obtained only after the voting process has concluded (i.e. when the expiry date of the problem has elapsed).

Figure B 5. Collaboration support service system interface — group voting.

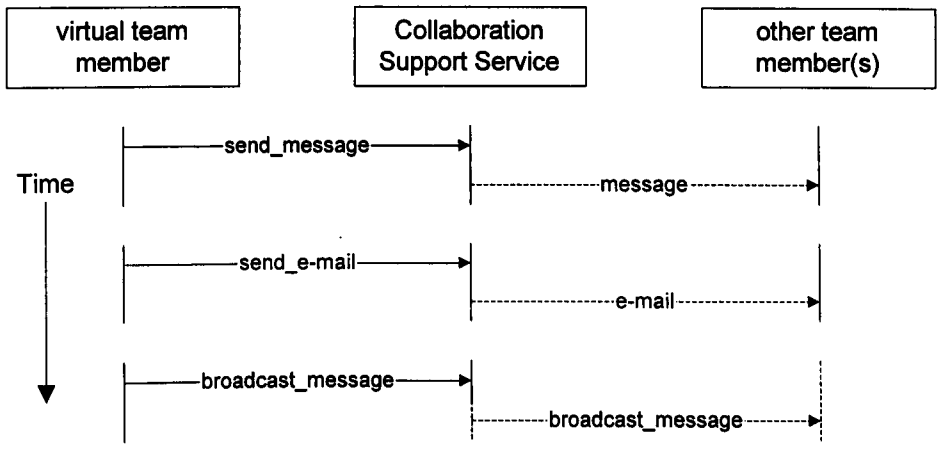


Description:

The Collaboration Support Service (CSS) is used to instantiate multi-user conferences. First a virtual team member creates a new conference after which other team members can retrieve conference details from the CSS. Team members must explicitly join a conference upon which they will receive all messages that have been added to that conference by other members before prior to their joining. On joining a conference, a team member can add new messages to that conference. When a new message is added, the CSS broadcasts the message to all participants currently subscribed to that conference (i.e. all members who have joined).

Team members must specify that they wish to leave a conference (they may return at any time as long as the conference is still running.) The conference must be explicitly finished to formally end the conference and prevent team members adding new messages.

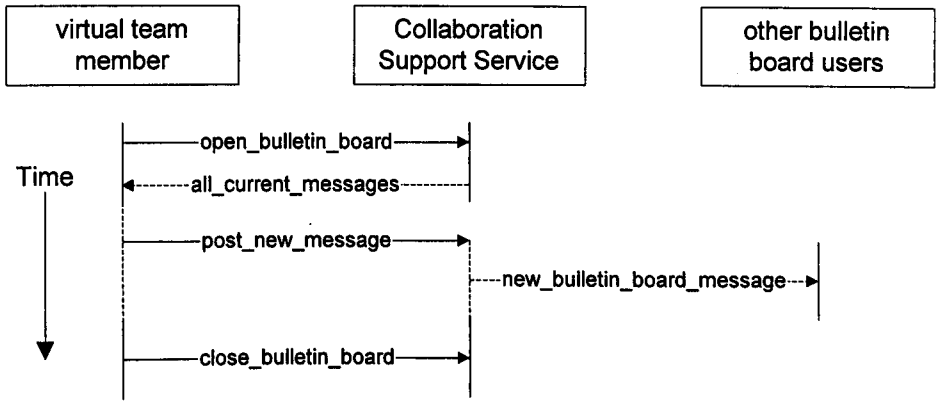
Figure B 6. Collaboration support service system interface — conferences.



Description:

The Collaboration Support Service (CSS) is used for messaging services. These include sending or broadcasting messages to other team members' CONCERT workstations and e-mail. A message is synchronously 'Sent' to a single user or 'Broadcast' to all current users (provided that those users are actually logged into the CONCERT environment at the time). E-mail is sent to a specific e-mail address asynchronously.

Figure B 7. Collaboration support service system interface — messaging services.



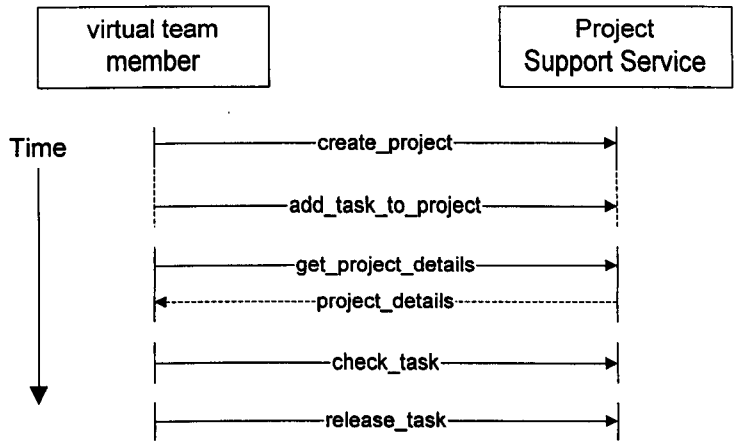
Description:

Bulletin boards are another form of messaging service in the Collaboration Support Service (CSS). Virtual team members 'open' the project bulletin board via the CSS which send the team member all the currently active messages stored on the bulletin board. The team member can then add new messages to the bulletin board which will automatically be re-distributed to all other users who currently have the same bulletin board open.

To end a bulletin board session, users must inform the CSS so that broadcasts are no longer sent to that user.

Figure B 8. Collaboration support service system interface — bulletin boards.

B4. Project support service

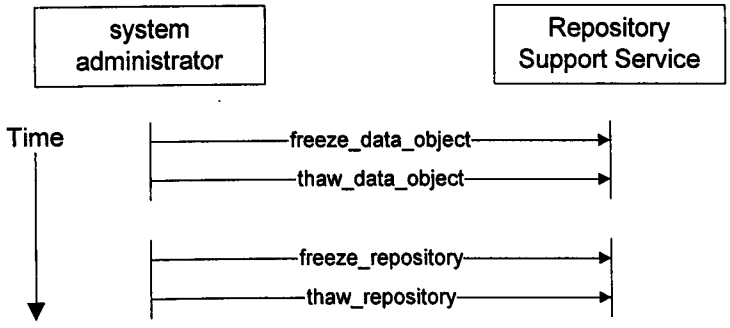


Description:

The Project Support Service (PSS) is called upon to manage projects and related tasks. The first step is to actually create a new project object and then populate this with one or more task objects. Users of the CONCERT environment can then retrieve these project details and actually perform the set tasks. This leads to some virtual team member checking the task (i.e. marking it finished) and then some other virtual team member releasing the task (i.e. marking it as complete).

Figure B 9. Project support service system interface.

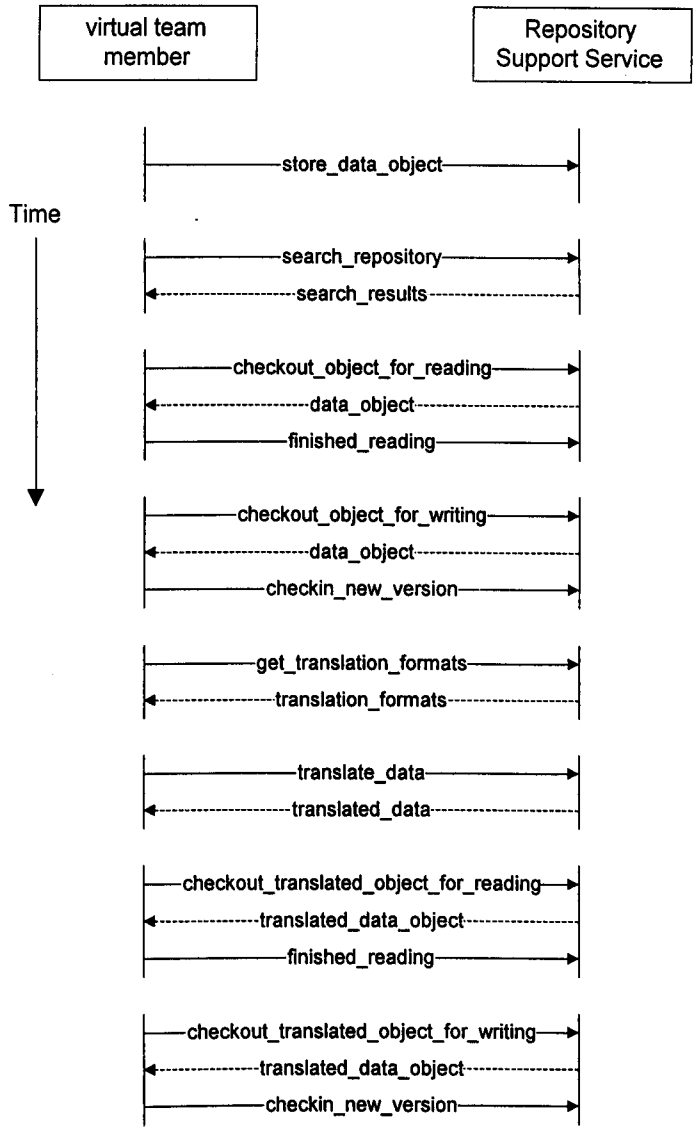
B5. Repository support service



Description:

System administrators can perform additional operations in the Repository Support Service (RSS). These include the freezing of individual data objects upon which the object is not allowed to be checked out of the repository for writing, and the subsequent thawing of the same data object. In addition, system administrators can freeze and thaw the entire data object repository in a similar way.

Figure B 10. Repository support service — system administrators.



Description:

The Repository Support Service (RSS) is called upon by virtual team members to perform data repository functions. Team members can store new objects into the repository after which there are three things that can happen to that object:

1) Users can search the repository for the object,
2) Users can checkout the object for reading, or
3) Users can checkout the object for writing.

Searching the repository will return a list of matching data objects (which may be empty). (For clarity, the checkout of data objects assumes no access control violations.)

When a user has finished reading or writing a checked-out data object they must inform the RSS so that (in the case of reading) notification locks can be released and (in the case of writing) a new version can be generated.

Translation of data is also performed by the RSS. The team member must first determine which formats can be translated by polling the RSS. Raw data can be translated by sending it to the RSS. Alternatively, team members can check out existing data objects in translated format for reading and writing. When a translated object is checked out for writing, the RSS must ensure that it is translated back into the correct format on return to the repository.

Figure B 11. Repository support service — virtual team members.

APPENDIX C: CONCERT INFORMATION SYSTEM OBJECT MODEL

The diagrams on the following pages are depicted in the Fusion notation. To fully appreciate the diagramming conventions of the Fusion method, the following excerpt, from the authors of Fusion, may be of use to the reader:

> *"Diagrammatic notations can be difficult to understand if the diagrams get large and sprawl. The object model notation is no exception, so we allow a diagram to be broken into several sub-diagrams. The complete object model is just the union of all the sub-diagrams.*
>
> *If two class boxes, with the same name, appear in the same object model, they denote the same class. The attributes of the class are formed by taking the union of all the attributes of the individual boxes. Thus each appearance need only contain the attributes needed for its immediate context.*
>
> *Relationships are treated in a similar way. Two diamonds with the same name, connecting the same classes, and with the same role names denote a single relationship that is formed by taking the union of all the attributes."*
>
> (pg. 22-23, Coleman et al. 1994)

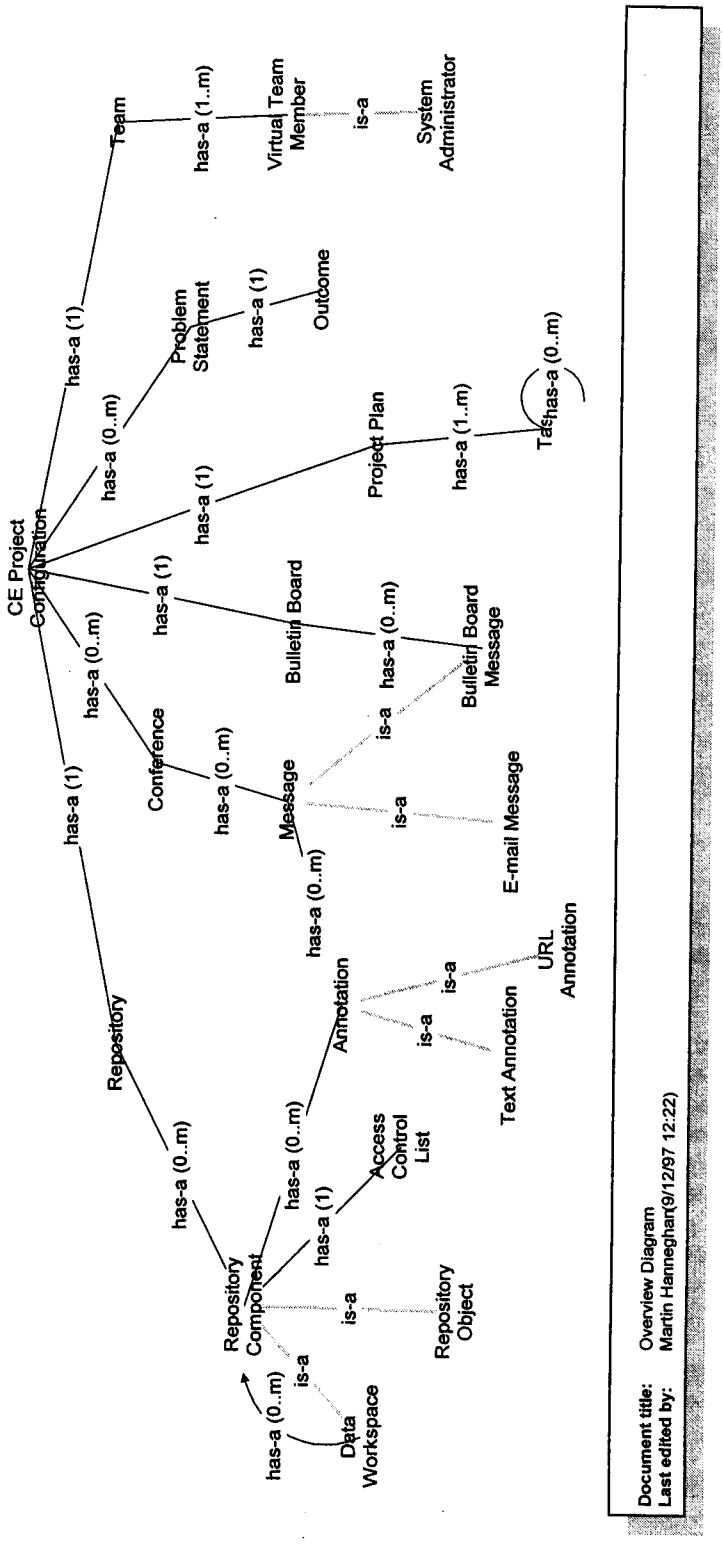
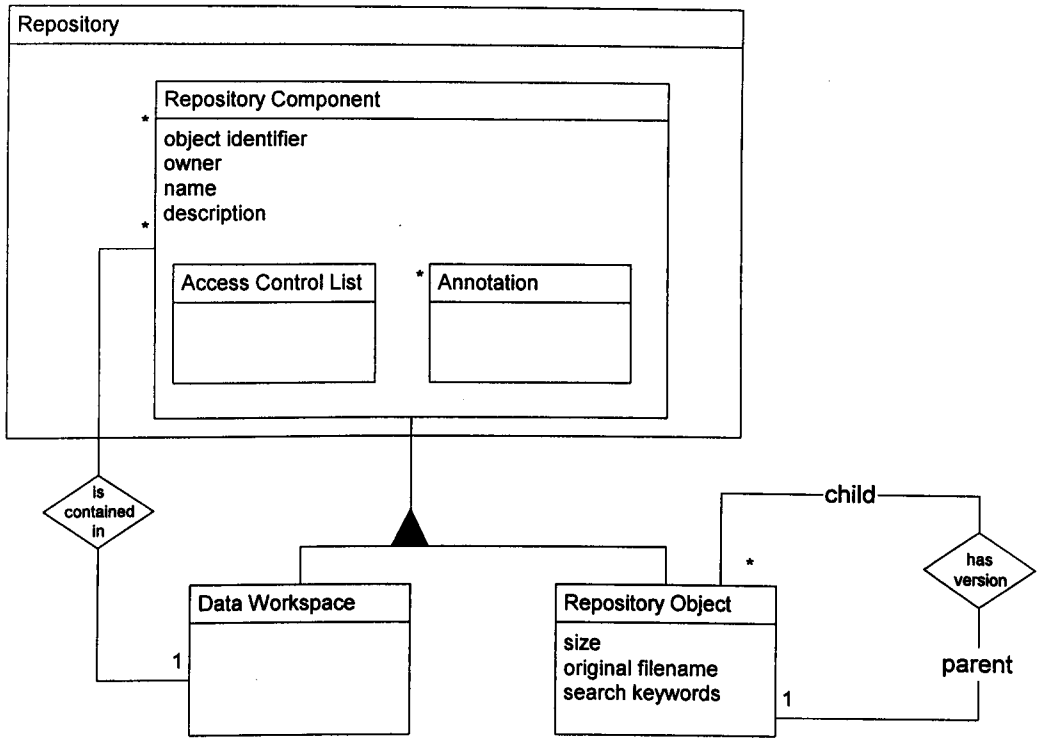


Figure C 1. Overview of object model.



Description:

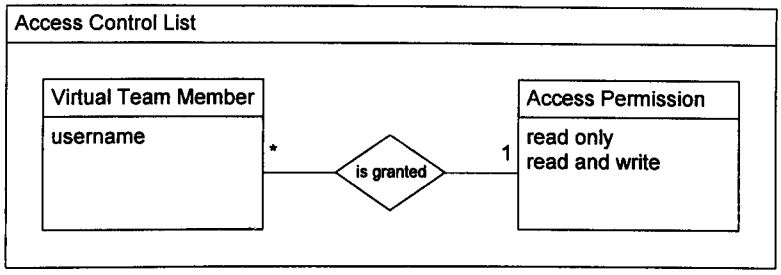
A **Repository** is the place where actual configuration items are stored for the life-time of the project. The **Repository** is capable of holding two different classes of object: **Data Workspaces** which are folders used to conveniently group like items together and **Repository Objects** which are actual data objects created during the project.

Data Workspaces can contain any number of **Repository Components** (of which **Data Workspace** and **Repository Object** are sub-classes). **Repository Components** contain an **Access Control List** and any number of **Annotations** and have the attributes: *object identifier* which uniquely identifies each component; *owner* which is the username of the team member who created the item; and a *name* and *description* used for descriptive purposes.

Repository Objects have the additional attributes: *size* (in bytes), *original filename* (which is used when team members edit the object) and *search keywords* (which can be used to quickly locate objects of a given type). One final aspect of **Repository Objects** is that they can contain child versions of themselves (i.e. they maintain a version tree).

Document title:	Repository
Last edited by:	Martin Hanneghan(2/12/97 12:32)

Figure C 2. Repository object model.

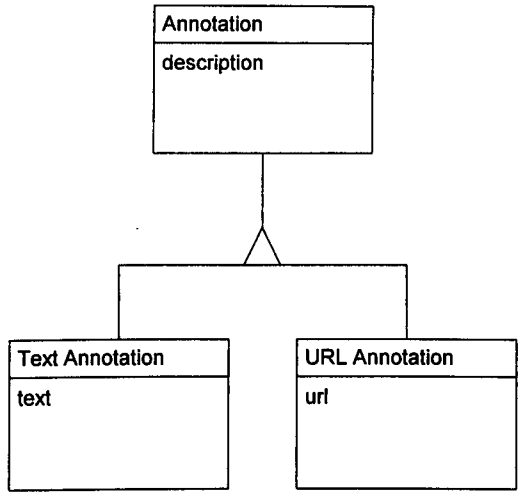


Description:

An **Access Control List** is the means by which the CONCERT environment controls access to **Repository Components**. It is a list of the **Virtual Team Members** who can access a given object and the **Access Permission** or type of access they are allowed: i.e. *read only* or *read and write* access.

Document title:	Access Control List
Last edited by:	Martin Hanneghan(2/12/97 12:24)

Figure C 3. Access Control List object model.



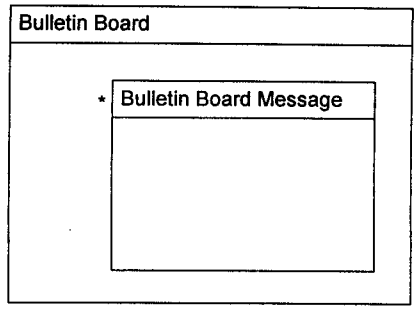
Description:

An **Annotation** is an additional descriptive reference used to support a given point of view. An **Annotation** can be one of a number of sub-types: e.g. **Text Annotation**, i.e. paragraph(s) of plain *text*; or **URL Annotation**, i.e. the annotation refers to information that is available at the given *URL* (Uniform Resource Locator) on the Internet.

Other possible type of Annotation include graphical annotations, audio annotations and video annotations (not shown here).

Document title:	Annotations
Last edited by:	Martin Hanneghan(2/12/97 12:31)

Figure C 4. Annotations object model.

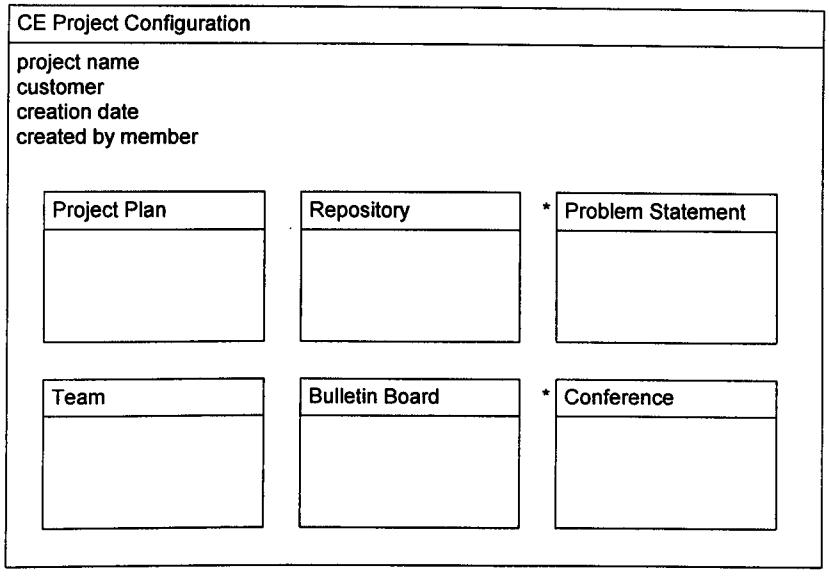


Description:

Bulletin Boards are used for storing temporal information or notices generated within a given **CE Project Configuration**. It can contain zero or more **Bulletin Board Messages**. Each **CE Project Configuration** has one **Bulletin Board**.

Document title:	Bulletin Board
Last edited by:	Martin Hanneghan(5/12/97 11:28)

Figure C 5. Bulletin Board object model.



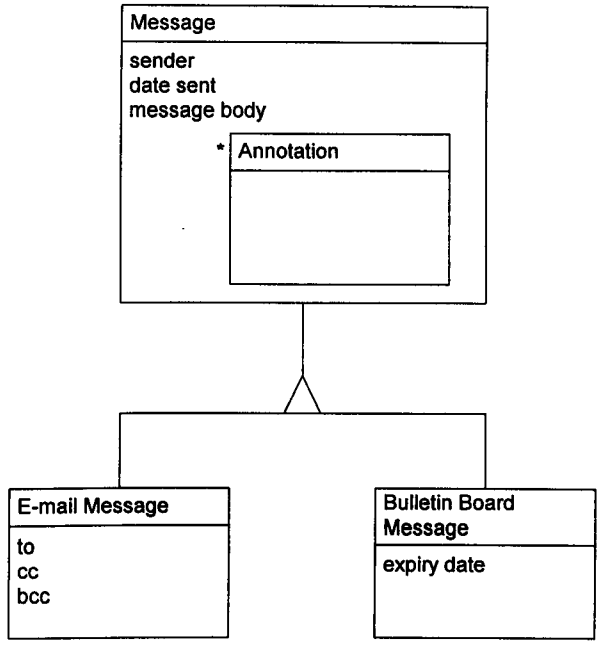
Description:

A **CE Project Configuration** is the one single place which holds all the configuration information for a given CE project. It is here that the **Project Plan**, the **Repository** for all configuration items, the details of this project **Team,** the project **Bulletin Board**, zero or more related **Problem Statements** and zero or more **Conferences** are stored.

A **CE Project Configuration** is for a specified *customer* and is given by its creator (*created by member*) a unique descriptive *project name* at its *creation date*.

Document title:	CE Project Configuration
Last edited by:	Martin Hanneghan(3/12/97 11:16)

Figure C 6. CE Project Configuration object model.



Description:

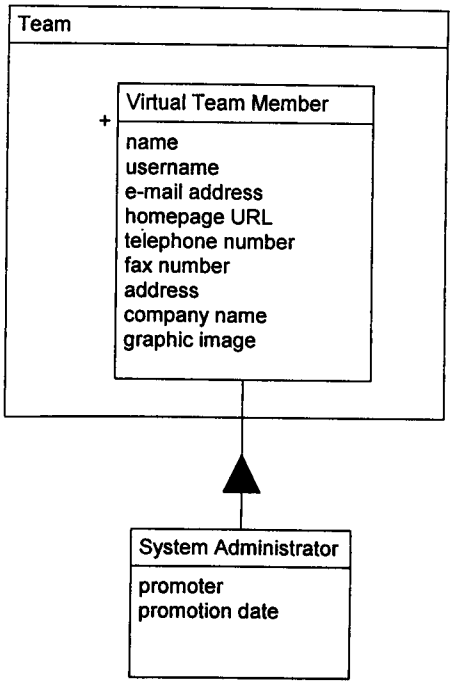
This model show the various types of **Message** that exist within the CONCERT environment. Each **Message** contains information about the username of the *sender*, the *date sent* and the actual *message body*. It can also contain zero or more **Annotations**.

The **E-mail Message** sub-class contains additional information such as who the message is *to*, the *cc* recipients (carbon-copy) and *bcc* recipients (blind carbon-copy).

The **Bulletin Board Message** sub-class contains the additional attribute *expiry date* which is used to invalidate the message after a given time.

Document title:	Messages
Last edited by:	Martin Hanneghan(2/12/97 15:54)

Figure C 7. Messages object model.



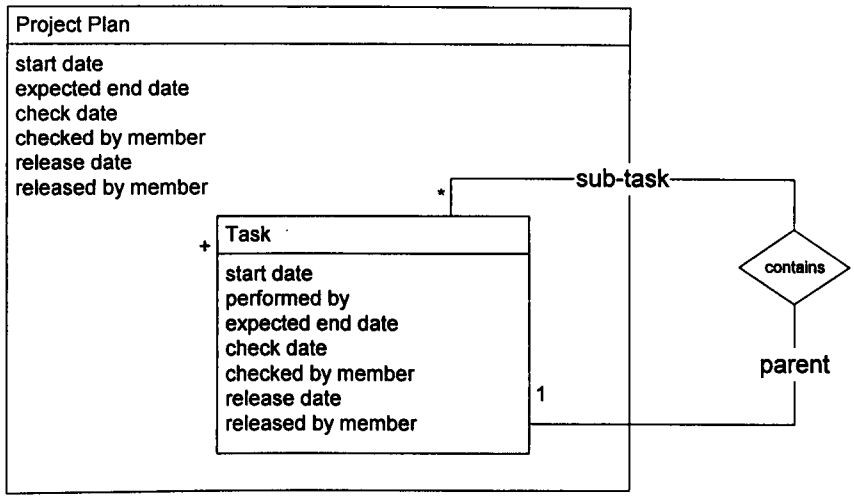
Description:

A **Team** is composed of one or more **Virtual Team Members**. Each **Virtual Team Member** records details of their *name*, CONCERT *username*, *e-mail address*, *homepage URL*, *telephone number*, *fax number*, *address*, *company name* and an optional *graphic image*. These details are used as contact information by both users of the CONCERT environment and the system itself.

One special sub-class of **Virtual Team Member**, the **System Administrator** has additional privileges which allow him / her to perform system functions within the CONCERT environment. Due to the nature of the functions that a **System Administator** can perform, only other **System Administators** can promote **Virtual Team Members** to this role (hence we have *promoter* and *promotion date* attributes for each administrator).

Document title:	Team and Virtual Team Members
Last edited by:	Martin Hanneghan(2/12/97 13:11)

Figure C 8. Team and Virtual Team Members object model.



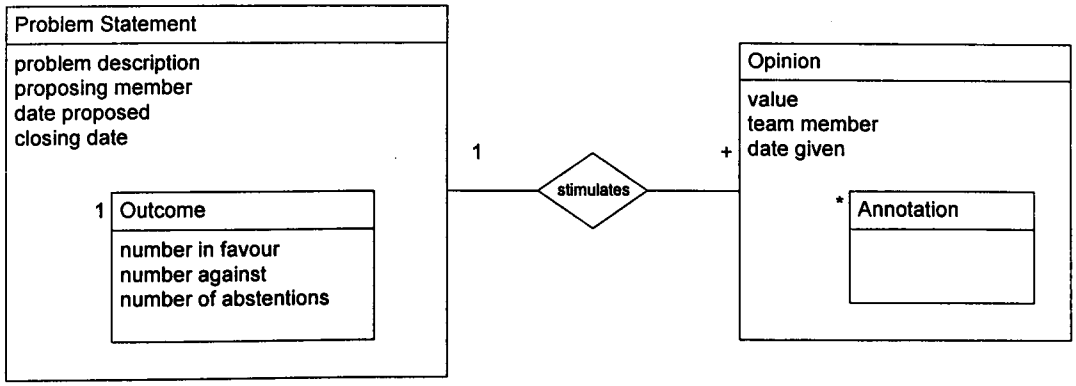
Description:

The **Project Plan** records the series of **Tasks** that need to be be undertaken in order to complete a given CE project. In accordance with ISO9000 requirements both the **Project Plan** and individual **Tasks** must be both checked and released by independent **Virtual Team Members** (i.e. by someone whom the actual task was not *performed by*).

Each **Task** can contain an optional number of sub-**Tasks** and so on ad infinitum.

Document title:	Project Plan
Last edited by:	Martin Hanneghan(3/12/97 11:37)

Figure C 9. Project Plan object model.



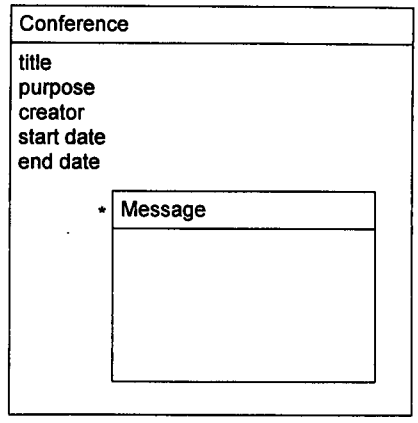
Description:

This object model describes the decision making process within the CONCERT environment. A **Problem Statement** is proposed by a **Virtual Team Member** (*proposing member*). Other **Virtual Team Members** within that **Team** can then express their **Opinion** on the statement, i.e. by specifiying a *value*: either whether they are in favour, against or abstain. Each **Opinion** may optionally contain a number of **Annotations** to support a given *value*.

Each **Problem Statement** records a *problem description*, the *date proposed* and a *closing date* when no more votes shall be allowed and the totals should be counted. A **Problem Statement** will produce one **Outcome** which records the *number in favour*, *number against* the motion, and the *number of abstentions*.

Document title:	Problem Statements
Last edited by:	Martin Hanneghan(3/12/97 11:30)

Figure C 10. Problem Statements object model.



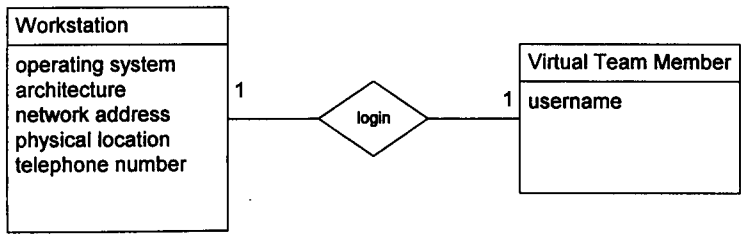
Description:

Conferences are used as a means of allowing multiple **Virtual Team Members** to communicate synchronously. They do this by adding **Messages** to a particular conference. Each **CE Project Configuration** can contain any number of **Conferences**.

A **Conference** has a *title* and a *purpose* which describes the context for the discussion. It also has a *creator* (which is the *username* of the **Virtual Team Member** who started the **Conference**) and a *start date* and an *end date* (when it has concluded).

Document title:	Conference
Last edited by:	Martin Hanneghan(5/12/97 11:31)

Figure C 11. Conference object model.



Description:

In order to use the CONCERT environment, **Virtual Team Members** must login from a **Workstation**. This is the physical computer terminal which is used to access the software components of the environment.

Each unique **Workstation** has attributes such as the *operating system* it uses, the hardware *architecture* and *network address*, the *physical location* of the machine and a *telephone number* that can be used to contact the operator of that **Workstation**. These details can be used by the CONCERT environment and other **Virtual Team Members** for communication purposes.

Document title:	Workstation
Last edited by:	Martin Hanneghan(5/12/97 13:33)

Figure C 12. Workstation object model.

1. Hanneghan, M., Merabti, M., and Colquhoun, G. (1998). "A Viewpoint Analysis Reference Model for Concurrent Engineering," *to appear in* Computers In Industry: An International Journal.

2. Hanneghan, M., Merabti, M., and Colquhoun, G. (1998). "CONCERT: A Middleware-Based Support Environment for Concurrent Engineering," in Proceedings of 2nd International Symposium on Tools and Methods for Concurrent Engineering (TMCE'98), Manchester Metropolitan University, UK, I. Horváth and A. Taleb-Bendiab, eds., pp. 446-455.

3. Hanneghan, M., Colquhoun, G., and Merabti, M. (1997). "A Scalable Intranet-Hosted Support Environment For Concurrent Engineering," in Proceedings of IEE Colloquium on Internet Technology and the Integrated Enterprise, Savoy Place, London, J. Boardman, ed. Institution of Electrical Engineers, Digest No: 97/149, pp. 9/1-9/4.

4. Hanneghan, M., Merabti, M., and Colquhoun, G. (1996). "The World-Wide Web as a Platform for Supporting Interactive Concurrent Engineering," in Proceedings of Advanced Information Systems Engineering - 8th International Conference, CAiSE'96, Heraklion, Crete, Greece, P. Constantopoulos, J. Mylopoulos, and Y. Vassiliou, eds., Springer-Verlag, Lecture Notes in Computer Science, 1080, pp. 301-318.

5. Heß, P., Hubel, H., Merabti, M., Colquhoun, G., and Hanneghan, M. (1996). "Concurrent Engineering Effizient Nutzen - Unterstützende Ansätze aus der Datenverwaltung (Using Concurrent Engineering Effectively - Supporting Approaches from Data Management)," VDI-Z Integrierte Produktion, 138(3), pp. 51-54.

6. Hanneghan, M., Merabti, M., Colquhoun, G., and Mills, B. (1996). "Computer-Aided Concurrent Engineering Using the Internet," poster presented at Graduate School Promotion Launch, Liverpool John Moores University, Liverpool.

7. Hanneghan, M., Merabti, M., and Colquhoun, G. (1995). "The Design Of An Object-Oriented Repository To Support Concurrent Engineering," in Proceedings of OOIS'95 - The 1995 International Conference on Object-Oriented Information Systems, Dublin, Ireland, J. Murphy and B. Stone, eds., Springer-Verlag, pp. 200-215.

8. Hanneghan, M., Colquhoun, G., and Merabti, M. (1995). "Concurrent Engineering Support Environments," poster presented at 11th International Conference on Computer Aided Production Engineering, Institute of Mechanical Engineers, London.

9. Hanneghan, M., Colquhoun, G., and Merabti, M. (1995). "Computers in Concurrent Engineering: An Enabling Technology," poster presented at IEE Colloquium on 'Concurrent Engineering: Getting it right first time!', Institution of Electrical Engineers, London.