13

Supporting Multidisciplinary Teams in Concurrent Engineering

J.H.Maxfield, L.T.P.Fernando and P.M.Dew
The Keyworth Institute,
Virtual Working Environments Laboratory,
School of Computer Studies,
University of Leeds,
Leeds LS2 9JT, England.
Tel: +44 532 336819
email: {max, ltpf, or dew}@uk.ac.leeds.scs

Abstract

This paper describes an architecture for supporting real-time collaboration in concurrent engineering within a distributed virtual environment. This architecture integrates a multiperspective distributed virtual environment with a standard product model and supports interaction among users through shared objects. The initial implementation of this architecture supports accurate assembly modelling and kinematic simulation for virtual prototypes and runs on a network of SGI Indy workstations over an ATM network. This environment enables designers to test the assembly and disassembly task within a distributed virtual environment. The realistic manipulation of the assembly models within the virtual environment is supported by constraint-based 3D manipulation techniques. The shared objects, that support collaboration, encapsulate product information in a standard format based on the developing draft ISO standard STEP (a standard for the exchange of product data).

Keywords

Distributed virtual environments, concurrent engineering, STEP, constraint-based assembly modelling, shared objects.

INTRODUCTION

The pressures for modern design and manufacturing companies to remain competitive in today's world markets has led many to investigate the adoption of concurrent engineering to reduce the lead time for new products and improve their quality. Concurrent engineering is a systematic approach to the integrated concurrent design of products and related processes,

J. Rix et al. (eds.), *Virtual Prototyping*

including manufacture and support. When using concurrent engineering specialist knowledge and expertise from downstream tasks of a sequential design process, such as manufacturing and maintenance, is introduced during the early design phases. The largest percentage of design and manufacturing costs are allocated during the first stages of a project. As a result decisions made during these early stages are the most difficult and expensive to correct at a later time. The basic philosophy behind concurrent engineering is to encourage the consideration of as many product development issues as possible, during these crucial early phases. Such considerations should result in fewer unexpected problems during subsequent development and consequently fewer design iterations, reduced development time and costs together with a better quality design.

Given sufficient resources, experts from several product development stages can be introduced by simply making the individuals available for consultation during the design process. An effective way of managing such consultation is through the creation of multidisciplinary teams. When employing such teams, the members will usually need to convene in a single location for regular meetings to review progress and make important decisions. Such meetings will normally be arranged in advance to give, the individuals involved, time to prepare necessary documentation and travel to the location of the meeting. During the meetings the experts from different areas of product development will be able to offer advice and suggestions from their own perspectives and can ensure that important issues are not overlooked. However, the physical co-location of a team is no longer a trivial issue because many companies are now exploiting the opportunity to trade in the global world market and consequently are becoming more decentralised in their activities. The travel, time and expense lost as a direct consequence, can inhibit the regularity and spontaneity of team interactions and act as a direct barrier to the successful implementation of concurrent engineering techniques.

High speed networks, distributed virtual environments and multimedia communications are now commercially available and have a potential to support interaction between geographically dispersed users in a much more dynamic and synchronous nature than traditional file exchange and electronic mail. Such advances have made it possible to develop collaborative working systems supported by networked computers in which geographical dispersion is transparent. A team that conducts its work in such a way is *virtually co-located* and thus called a *virtual team*, since interactions only require the participants to be available at the same time, but not necessarily at the same place.

This paper discusses a system architecture that supports interaction between members of a geographically dispersed multidisciplinary virtual team who are engaged in product development activities. We call such a system a *Distributed Virtual Engineering Environment.* In particular the environment allows the team to interact, reach a common understanding of the problems quickly and then make decisions regarding those problems from multiple perspectives using a distributed virtual environment. Each user accesses the distributed virtual environment through a user configurable interface which has been designed to allow the easy integration of further engineering applications to support different engineering perspectives. The applications interoperate using shared objects that encapsulate product information in a standard format based on the developing draft ISO standard STEP (a standard for the exchange of product data).

BACKGROUND

Concurrent engineering has received a great deal of attention in the engineering and management research communities since its introduction over a decade ago and much work has been published on its advantages over more traditional sequential design processes. Major research projects such as DARPA DICE (Cleetus 1993), SHARED (Wong *et al* 1993) and PACT (Cutkosky *et al* 1993) are all addressing issues involved in computer support for managing and co-ordinating multifunctional, cross-disciplinary teams in this context. However these projects have mainly concentrated on the asynchronous activities of such teams. Although this is important, teams must also be able to communicate effectively during meetings. This is especially difficult if the team is geographically dispersed. Certain issues that still need to be addressed for synchronous working in concurrent engineering have been indicated in several papers (Prasad *et al* 1993 and Clausing 1993). In summary these issues are :

- The establishment of virtually co-located multidisciplinary teams with integration of and mapping between individual view points.
- The ability to share and exchange standard product data and tools.
- The ability to perform collaborative decisions in a single trade-off space with a common understanding of the problems.

Some concurrent engineering projects are beginning to address some of these issues. The SHARE (Toye *et al* 1993) project is addressing negotiation and trade-off in real-time through videoconferencing. Support for synchronous collaboration in virtual teams has also been explored within the DICE project through the use of MONET, a teleconferencing system, and COMIX, a system for transparently sharing X-Windows applications (Cleetus 1993). However, these projects are not investigating the use of distributed virtual environments for supporting collaboration over a virtual prototype of a product. Neither do any of the collaborative tools that are used tackle the problems of real-time multiperspective meetings as required by multidisciplinary teams.

There now exists many commercial and non-commercial tool kits for the creation of distributed virtual environments, for example dVS (Division), World Tool Kit (Sense8), MR-Tool Kit (University of Alberta), DIVE (Swedish Institute of Computer Science). Many research projects are using such tool kits to develop distributed or single user virtual environments for specific engineering applications. For example, Bayliss *et al* (1994) are investigating a virtual manufacturing environment consisting of a machine shop in which engineering components can be made. The VirtuOsi project is investigating the organisational issues involved in the formation of a virtual factory (Benford 1994). NASA have conducted a successful experiment in which they used a virtual prototyping system called Preview to assist the correction of the Hubble Space Telescope (Hancock 1993).

These projects are just a few examples of many that are investigating support of specific engineering applications in virtual environments. However, the effective integration of these engineering applications in a multiperspective distributed virtual environment has not been addressed. The integration of international product data standards such as STEP in distributed or even single user virtual environments has not yet progressed beyond the ability to access IGES or DXF (AutoCAD) geometric definition files. Another limitation in current virtual environments is the lack of support for accurate positioning of objects in 3D space. At present,

most of these systems employ crude collision detection techniques based on bounding boxes. Such techniques fail to provide powerful and accurate 3D manipulation methods necessary for exploiting the virtual environment technology for engineering applications such as solid and assembly modelling. Techniques such as accurate constraint-based 3D manipulation (Fa *et al* 1993) are essential for supporting the realistic manipulation of solid models within virtual environments, especially when dealing with the integration of engineering application.

THE DISTRIBUTED VIRTUAL ENGINEERING ENVIRONMENT

The distributed virtual engineering environment has been developed to satisfy a number of requirements which will be discussed in section 3.1. The environment is based on a number of concepts which are outlined in section 3.2. A description of the detailed architecture and its current implementation is presented in section 3.3.

Requirements

A fundamental requirement for synchronous collaboration between participants of any distributed meeting, is the ability to share information in *real time*. This shared information will then form the basis for discussion within the meeting. A second fundamental requirement that is specific to collaboration in multidisciplinary teams is the ability to make collaborative decisions in a single trade-off space with a common understanding of the problems and with integration of and mapping between individual *perspectives*. A perspective defines a context within which the shared information can be manipulated in a meaningful way by an individual.

In addition, several requirements have been considered during the implementation of the distributed virtual engineering environment. These are that the architecture should :

- Be open and extensible so that different perspectives and engineering applications can be integrated easily.
- Have the ability to support user collaboration through sharing product data in real time.
- Be built on emerging and established standards where possible, including the product modelling standard STEP.
- Have the potential to be scalable for large meetings with a dynamic number of participants.
- Provide support for quality of service over high speed networks such as ATM, for real time interaction and communication.

A Conceptual View Of The Architecture

This section introduces a number of fundamental concepts that are used in the DVE environment to satisfy some of the requirements outlined in the previous section. For each concept, a high level description of the mechanisms employed by the DVE environment is also presented. A conceptual view of the DVE architecture is illustrated in figure 1.

The users *perspective* defines what information is meaningful to them and how they will manipulate that information. The DVE environment uses the concept of an *information mask* to define what information is meaningful to a user with a given perspective. The mask acts

conceptually as a filter that defines what subset of the information in the shared space a user can actually access. The mask can also be used to filter an entire product model to define what information the user may add to the shared information space.

The perspective also defines how the user will manipulate the information they access. To support this, each user of the DVE environment has a user interface that they may *configure* with the operations they wish to perform and the visualisation style they wish to use. These operations are actually performed on the shared information by a set of distributed engineering applications that are invoked and controlled in the background automatically by the DVE environment. The users may modify their perspectives at any time during a meeting by changing the configuration of their interface. This will imply a change in the users information requirements and consequently a change to the users information mask. Such changes may also affect in the set of background engineering applications that support their perspective. The concept of the perspective and how it is supported within the DVE environment is illustrated in figure 2(a).

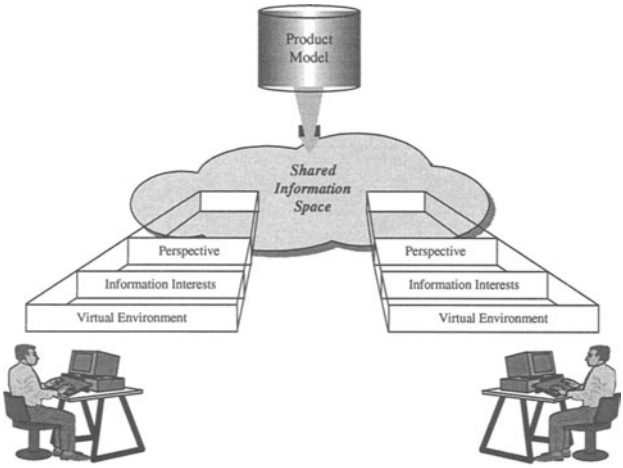
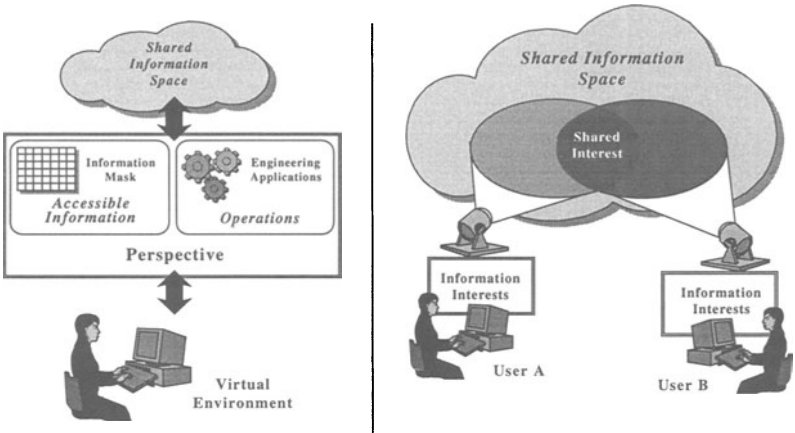


Figure 1 A Conceptual View of the DVE Environment

As a meeting progresses the discussion may change focus and more information may be added to the shared space from the product model. The shared information space can potentially contain a huge amount of information which may be meaningful to a user, and therefore satisfy their information mask, but may not all be of interest. The DVE environment allows a user to choose which subsets of the available information they are interested in and *register* this interest with the environment. Users may change their interest as the meeting progresses by registering an interest in further sets of information or discarding an interest in a particular set of information. Collaboration can occur when users register an interest in the same information, i.e. their particular interests intersect. Users may still collaborate over information that they do not share an interest in, if the information they view is related. In this case, changes can be propagated through the relationships using a constraint manager that will

maintain a relationship (or dependency) graph for the information within the shared space. The users will therefore still be able to see changes that affect their information.



(a) The Perspective is composed of accessible information and operations.

(b) Users register an interest in information held in the shared space. Users dynamically share information that they share an interest in.

Figure 2 The Fundamental Concepts Employed by the DVE Environment

Internally the DVE environment maintains either a *passive* or *interactive* interest in an particular area of the shared information space for each user. A *passive* interest is automatically registered in all information that the user accesses. An *interactive* interested is registered in any information that the user attempts to modify or manipulate in any way. By distinguishing between passive and interactive interests in this way, locking techniques can be employed by the environment to eliminate any chance of inconsistency in the shared information space. The concepts and mechanisms of interest registration are illustrated in figure 2(b).

A Detailed View Of The Architecture

A layered view of the detailed DVE architecture is illustrated in Figure 3. At the heart of this architecture is the shared information space called the *Product Data Sharing System* (PDSS). This system makes extensive use of a library of sharable objects that can be instantiated and populated with product information. These objects and the mechanisms employed by the PDSS for sharing these objects are discussed in the following sections. The users access the shared information through a graphical user interface that visualises the information in a virtual environment. They may then directly interact with the information and perform operations supported by a set of engineering applications. The following sections discuss this interface and the underlying mechanisms that allow a user to configure their interface to suit their required perspective. Finally this section also discuess the integration of constraint based 3D manipulation within the architecture.

The Product Data Sharing System
This section discusses the structure of the sharable objects that are instantiated from an object library when required. It also descibes the distribution and sharing mechanisms used within PDSS.

The shared product information must be accessible by many different applications and therefore a neutral, usage independent representation for the information is important. A *product data model* is an integrated set of data schemata that describe such a standard format and content for storing product data. An instance of the product data model will contain data regarding a specific product and is called a *product model.* The most important issue here is that the information in a product model is in a neutral format which is totally independent of the way in which it is used. This makes a product data model an ideal format for the storage of product information within the objects used by the DVE environment.

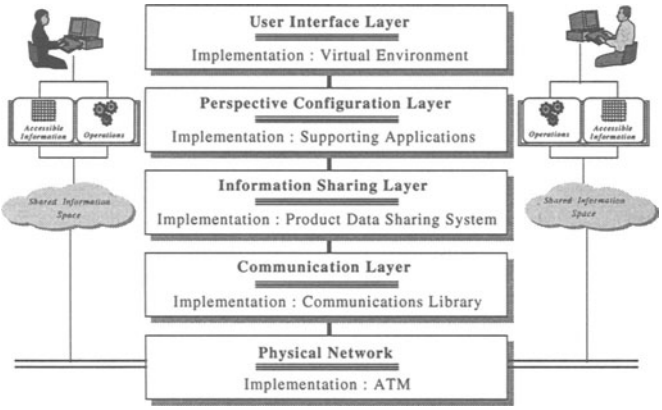


Figure 3 A Layered View of the DVE Environment

The objects generated and used by the DVE system, encapsulate product information and methods for validating the correctness of that information. This is held in an area of the object called the *payload.* Each object in the library consists of a payload (the information that will be shared) and useful methods for encoding and decoding the payload to assist in sharing the object. These methods are derived from a basic object type by all of the objects as illustrated in figure 4.

Given objects that encapsulate product information structured this way, it is feasible for this information to be exchanged, without any translation, with a product model database, if a common product data model is used to structure the information in both. Therefore DVE environment users can add information to the environment from a persistent product model database during the meeting. If the users also wish to keep any changes made as a result of decisions made in the meeting, then the modified information can be transferred back to the product model database at the end of the meeting.

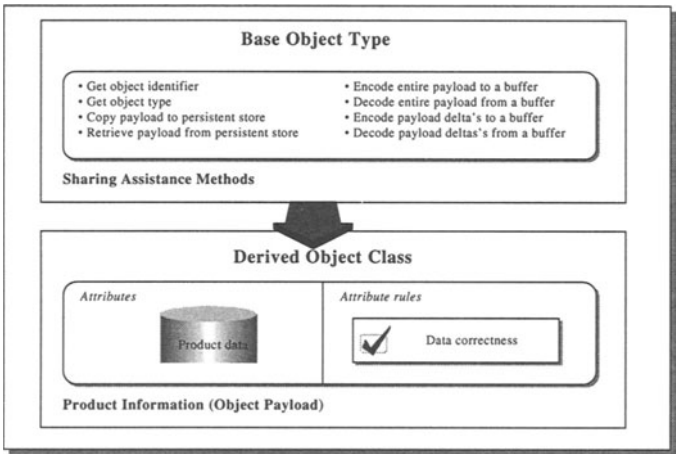


Figure 4 The Structure of a Typical Shared Object

At the heart of the PDSS is an *Object Manager*. The manager controls the instantiation, population, shared access and destruction of the objects. It does not distinguish between user interfaces and engineering applications but treats all entities that wish to share product information as *clients*. The object manager will distribute copies of the objects to its client on request. Each client has a *PDSS wrapper* that allows the client to register an interest in some product information using *registration* services and then view and edit the information using a standard set of *enquiry* and *modification* services. The wrappers will also inform the interface or application of any changes to its' local information made by another client, through a set of *notification* services. The PDSS wrappers effectively hide the sharing mechanisms and communicate with the object manager through a standard PDSS protocol which is currently built on top of a network and platform independent communications library. The internal components of the PDSS are illustrated in figure 5.

The object manager and PDSS wrappers all maintain local copies of the objects. The responsibility for maintaining consistency in the contents of the local copies is shared by the wrappers and the object manager. The PDSS protocol implemented in both the wrappers and object manager defines the interaction that is required to retrieve objects and maintain consistency in them. The main protocol services are summarised as :

- **Initialisation** : Each application and user interface must become a *client* of the object manager to gain access to the shared product information. During the initial exchange the each client will give the object manager a copy of its interest mask. The object manager will then send the client a high level list of components and assemblies that are currently available either, in the environment as virtual prototypes already, or available to retrieve from the product database.

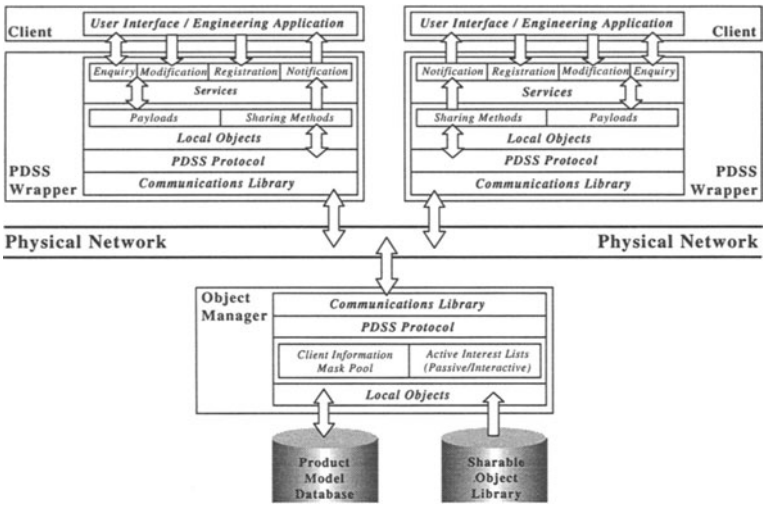


Figure 5 The Sharing Mechanisms

- **Registering interest** : When a user registers an interest in a component or assembly the object manager will first check whether the product information that represents it has already been added to the environment by another user. If it has not then the manager will retrieve the product information from the database and instantiate and populate the required objects to hold the data. The object manager applies the users information mask to the set of objects and gathers together the objects that will be meaningful to the client. These will then be dispatched to the client, where a local copy will be maintained in the PDSS wrapper.
- **Changing the contents of an object** : If a client modifies the product information in any way then the resulting changes or *delta's* in the payloads of the shared objects are propagated to the object manager by the clients' PDSS wrapper. The manager will propagate the deltas on to any other clients that are interested in the modified objects and then update its own local copy. Each interested client will receive the payload deltas, apply them to its own local object copies, and inform the client of the change using a notification service.

The Configurable Virtual Environment and Supporting Applications

The DVE system supports collaborative working between its users through 3D virtual environment interfaces. Each interface provides a view into the shared product information space managed by the PDSS. The interface visualises components and assemblies as 3D solid models and allows a user to manipulate the models in 3D and alter their viewing position by moving around in the environment.

The current implementation of the user interface uses non-immersive technology and does not represent the user in the environment in any way. Therefore to allow users to indicate

regions or objects of interest in the environment to other users during a discussion, the interface provides the necessary tools for creating, manipulating, highlighting and labelling shared 3D pointers. Each user can construct a different perspective or personal interface for the virtual environment by configuring the basic interface to support a variety of engineering applications. Multiple perspectives of the same shared information can be supported by the DVE environment because the information is in a neutral and standard format which is accessible by all, but independent of any applications used by a client of the PDSS.

The user configures the virtual environment by choosing from a list of applications those that they wish to be supported by their virtual environment interface. Each application that chosen by a user relates to an engineering application that is integrated into the DVE architecture and invoked as a background process by the DVE environment. The user interface provides an *application toolbar* from which the user may select and adjust operations, modes and ranged values for each application selected. An operation is represented as a push button within the toolbar and may require parameters to be selected within the virtual environment. Examples of operation parameters that can be selected in the current environment include solids, faces, edges, vertices, or points on a face, edge or points in open space. Given the correct type and number of parameters, the operation can be requested by the user. A user selects the modes of an application using toggle buttons that can be switched on or off. Ranged values such as sizes, scales, and tolerances, are adjusted using sliders. Since the current implementation of the user interface is non-immersive it uses standard push buttons, toggle buttons, and sliders provided by the windowing environment to build the application toolbar. A picture of the toolbar (right) and virtual environment (left) taken from the current implementation is shown if figure 6.

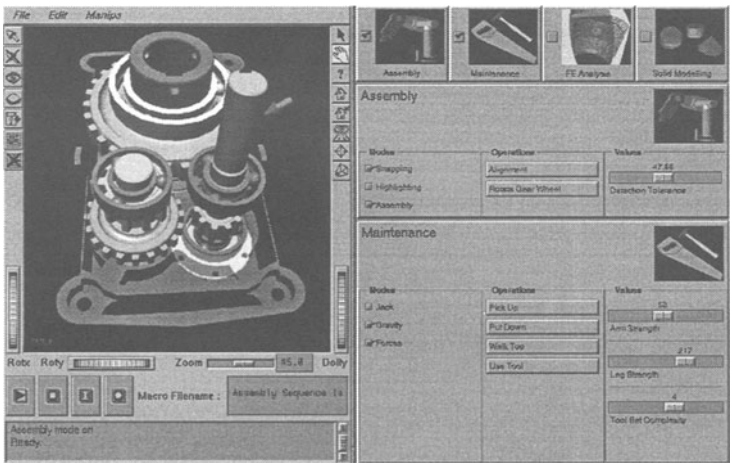


Figure 6 The Configurable Virtual Environment Interface

Figure 7 illustrates how the PDSS is used to support an individuals perspective within the virtual environment interfaces with support from engineering applications. When a user chooses a particular engineering application to support their perspective from the application toolbar, the DVE system invokes the appropriate background application to perform the

operations required. The user interface and applications exist as different clients of the PDSS and use a special set of shared objects to communicate with each other. These objects are called *transient objects* because the lifetime of the information they contain is only as long as of the time the user spends within the DVE environment. The user interface informs each selected application of the users interests and requests operations, mode changes, and adjustments to ranged values using a *transient object* that is shared by the user interface and the application itself. Using this technique, an application can be shared between many users, if required, by simply sharing the applications' transient object.

The DVE architecture supports the integration of a diverse range of engineering applications. To integrate an application a configuration file must be created that describes how to execute the application and the operations, modes and ranged values that will be offered to the user in the toolbar. In addition to this file, the application must be made aware of the PDSS wrapper services so that it can access and manipulate the information held in the shared objects.

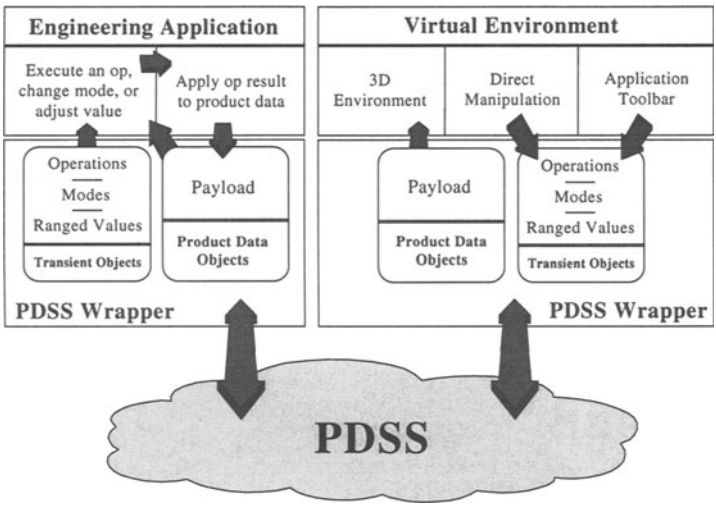


Figure 7 Virtual Environment and Supporting Application Interoperation.

Interactive Constraint Based 3D Manipulation

The management of geometric constraints within the shared information space is supported by a *Constraint Manager.* Constraint based 3D manipulation techniques, previously developed at Leeds (Fa *et al* 1993), are supported within the distributed virtual environment through the integration of this Constraint Manager within the DVE architecture.

The Constraint Manager automatically registers an interest in the information that the users are accessing and monitors the relative positions of the geometric solids as they are manipulating within the virtual environment. *Automatic constraint recognition techniques* are used within the Constraint Manager to recognise geometric constraints between geometric

entities from the users' 3D manipulations. Constraints such as against, coincidence, cylindrical fit, gear fit and screw fit are automatically recognised using this technique. Once the geometric constraints are recognised, a technique referred to as *allowable motion* is used to manipulate the under constrained models without invalidating the previously satisfied constraints. For example, when the user position a block on top of a larger block through an against constraint, the allowable motion of the block is derived to be translation and rotation with respect to the bottom block. In the case of a gear fit between two gears, the rotation of one gear is transmitted to the second gear through a coupled rotation. The combination of automatic constraint recognition and allowable motion techniques supports the accurate positioning of solid models in 3D space using 3D input devices, such as a dataglove or spaceball, to assemble complex solid models. Once the objects are assembled, the kinematic behaviour of assemblies are automatically simulated using the allowable motions of the solid models. The Constraint Manager therefore supports the highly interactive virtual environment in which the users can carry out assembly modelling and kinematic simulation of virtual prototypes in a realistic manner.

IMPLEMENTATION AND RESULTS

The current prototype version of the DVE architecture runs on a network of Indy workstations over an ATM network. The architecture has been implemented using C++ and the current virtual environment interface is based on Inventor 2. The shared object library used in PDSS is generated by a compiler that has been developed by the authors to convert an Express schema into a set of C++ classes. The initial implementation uses part of the product data schema that has been developed within a project called MOSES, by the department of Mechanical Engineering at Leeds University (Henson 1994).

The current prototype version of the DVE architecture supports an assembly modelling perspective through the Constraint Manager. It enables engineers to register an interest in assembly models and perform assembly and disassembly operations within the shared information maintained within PDSS. A video conferencing system is run along-side with the virtual environment to support the communication between the members of the product development team. Several case studies are currently being used to test the feasibility of the environment for supporting collaboration among team members including a complex speed reducing gear box.

CONCLUSIONS AND FUTURE WORK

In this paper we have discussed an architecture for supporting multiperspective collaboration over complex interrelated product information for geographically dispersed virtual teams in concurrent engineering. The current implementation of the DVE environment supports assembly modelling and kinematic simulation of virtual prototypes. When used in conjunction with other collaborative tools, the DVE environment allows geometric and assembly problems to be explained clearly and assists a virtual team in reaching a common understanding of problems quickly. The environment will then also allow the users to discuss and test precise solutions in real time. The architecture is currently being evaluated and refined using case studies.

Work is now underway within the Keyworth Institute to integrate further engineering applications into the system to support a wider range of perspectives. These applications include the JACK (Badler *et al* 1993) human factor modeller and a solid modelling kernel based on Parasolid. JACK will allow us to demonstrate a maintenance engineers perspective more effectively by allowing a user to consider the ergonomic and spatial issues involved in real human maintenance, in more detail. The integration of a solid modelling kernel will allow parametric variation of the virtual prototypes in the DVE environment.

A more powerful communication architecture is being developed to support the DVE architecture which will deliver Quality of Service over ATM, synchronisation and directory services (X.500) supporting access to distributed data and users. Experiments involving the scalability of the architecture will study the replication of components of the PDSS and the management of multiple, but related meetings. We are also planning to integrate multimedia information sharing. This will allow us to integrate other collaborative tools into the system and support multimedia annotation of the product information, which can then be stored as interactive minutes for a meeting held in the DVE system.

ACKNOWLEDGEMENTS

The authors would like to thank the members of the DVE project and the Keyworth Institute of Manufacturing and Information Systems Engineering for supporting this work. Thanks also go to Brian Henson and Martin Ashworth within the Department of Mechanical Engineering.

REFERENCES

K. J. Cleetus. (1993) The Virtual Team Framework. In *Concurrent Engineering : Tool and Technologies for Mechanical Systems Design.* NATO ASI Series, Springer-Verlag.

A. Wong and D. Sriram (1993) Shared : An Information Model for Cooperative Product Development. Technical report, Intelligent Engineering Systems Laboratory, MIT, Department of Civil Engineering, MIT, Cambridge, MA, USA, July.

M. R. Cutkosky *et al* (1993) Pact : An Experiment in Integrating Concurrent Engineering systems. In *IEEE Computer*, pages 28-37, January.

B. Prasad, R. S. Morenc, and R. M. Rangan (1993) Information Management for Concurrent Engineering : Research issues. in *ISPE Concurrent Engineering : Research and Applications*, pages 3-20.

D. P. Clausing (1993) World Class Concurrent Engineering. *In Concurrent Engineering : Tool and Technologies for Mechanical Systems Design.* NATO ASI Series, Springer-Verlag.

Toye *et al* (1993) Share : A Methodology and Environment for Collaborative Product Development. In proceedings of the *IEEE Workshop on Enabling Technologies : Infrastructure for Collaborative Engineering.* IEEE Press.

G.M. Bayliss, A. Bowyer, R.I. Taylor, and P.J. Willis (1994) Virtual Manufacturing. In *CSG 94*. Information Geometrics, April.

Steve Benford, John Bowers, Stephen Gray, David Leevers, Tom Rodden, Michael Rygoland, and Vaughan Stanger (1994) The Virtuosi Project. In *VR94*, February.

D. Hancock (1993) 'Prototyping' the Hubble Fix. In *IEEE Spectrum, Special Issues on Virtual Reality, tools, trends and applications*, pages 34-39, October.

M. Fa, T. Fernando, and P. M. Dew (1993) A Virtual Environment for Interactive Constraint-based Solid Modelling. *Eurographics93*, September.

N. I. Badler, C. B. Philips, and B. L. Webber (1993) *Simulating Humans: Computer Graphics, Animation and Control.* Oxford University Press, edition 1, June.

B. Henson (1994) An Assembly Data Model. Technical Report, moses-misc-10, MOSES Group, Department of Mechanical Engineering, University of Leeds, Leeds, LS2 9JT, April.

BIOGRAPHY

John Maxfield is a PhD research scholar working on the Distributed Virtual Engineering Project within the Keyworth Institute of Manufacturing and Information Systems Engineering. His research interests include computer supported collaborative working over high speed networks, object oriented software engineering and distributed virtual environments for engineering applications. John received a first class honours degree in Computer Science from the University of Leeds in 1992.

Dr Terrence Fernando is a lecturer in the School of Computer Studies at the University of Leeds. His research interests include interactive constraint-based solid modelling and distributed virtual environments for concurrent engineering. He leads these research activities within the Virtual Environment Research Group. Terrence received his PhD in Computer Science from the University of Manchester Institute of Science and Technology (UMIST) in 1987.

Professor Peter Dew is the Professor of Computer Science at the University of Leeds and Deputy Director of the Keyworth Institute of Manufacturing and Information Systems Engineering. His main research interests include virtual working environments for virtual organisations and scalable, parallel and distributed computing. Peter is head of the division of computer science where he leads groups in virtual working environments and scalable systems and algorithms