# A Distributed Feature-based Environment for Collaborative Design

Wei-Dong Li‡, Yi-Qiang Lu, Hong Zhou
Singapore Institute of Manufacturing Technology
71 Nanyang Drive, Singapore, 638075
‡ Email: wdli@gintic.gov.sg, Tel: +65-6793-8354, Fax: +65-6791-6377

Soh-Khim Ong, Andrew YC Nee, Jerry YH Fuh, Yoke-San Wong
Department of Mechanical Engineering, National University of Singapore
10 Kent Ridge Crescent, Singapore, 119260

## ABSTRACT

This paper presents a client/server design environment based on 3D feature-based modelling and Java technologies to enable design information to be shared efficiently among members within a design team. In this environment, design tasks and clients are organised through working sessions generated and maintained by a collaborative server. The information from an individual design client during a design process is updated and broadcast to other clients in the same session through an event-driven and call-back mechanism. The downstream manufacturing analysis modules can be wrapped as agents and plugged into the open environment to support the design activities. At the server side, a feature-feature relationship is established and maintained to filter the varied information of a working part, so as to facilitate efficient information update during the design process.

**Keywords:** *Distributed environment, collaborative design, feature-based modelling*

## 1. INTRODUCTION

For a complex design task, a design team is usually engaged, and the communication and collaboration among members in the team are crucial to enable the design to be carried out effectively. With the rapid development of IT, it is possible and imperative to develop a distributed design environment, in which geographically-distributed design systems can be integrated and a design team can be set up within the Internet/Intranet. In this environment, design information among the team can be exchanged across physical and temporal boundaries efficiently. With the integration of manufacturing analysis modules in the environment, downstream manufacturing processes of the product life-cycle can be considered and evaluated in the initial design phase, which will lead to better engineered products with higher quality, less iterations and more cost competitiveness.

Recently, some research and developments have been carried out in this area. Different collaborative design mechanisms and system architectures have been designed. A summary is listed in Table 1.

In this paper, a distributed feature-based modelling environment to support collaborative design is described. A designed part is modelled at the server side, and the clients are responsible for design parameter input, visualization and selection of operations for the part. The information communicated between the clients and the server is wrapped as events, and an event-driven and call-back mechanism is developed to efficiently update information in the environment. The environment is open and scaleable, and it can dynamically integrate downstream manufacturing analysis agents without re-initializing the whole system. At the server side, a feature-feature relationship is established to differentiate the varied information of a part during its creation or edition process, so as to efficiently update information in the network.

## 2. SYSTEM FRAMEWORK

The system framework is illustrated in Figure 1. The three parts in the system are as follows:

(1) Design clients;
(2) A collaborative server; and
(3) Intelligent downstream manufacturing analysis and optimization agents.

The detailed functions of each part in the environment are described in Table 2. The organization for designing a part collaboratively in the environment is depicted in Figure 2.

## 3. DISTRIBUTED MECHANISM

### *3.1 Remote Method Invocation (RMI) protocol*

Java RMI is a simple and yet powerful protocol for distributed object design. Java RMI-based objects can be quickly deployed and managed across networks, and the RMI mechanism is a much easier and lighter weight approach to distributed object design. In this research, the establishment of the distributed design environment is based on the Java RMI mechanism.

According to the Java RMI mechanism, through defining remote interfaces, methods and objects can be used for remote calling and transmission. The remote interfaces defined in the environment are given in Figure 3.

### *3.2 Event-driven and call-back mechanism*

In order to enable clients to update design information only when the server has a new event to communicate, instead of routinely pinging the server for information and creating a network backlog, a call-back mechanism is employed to develop a high-performance and robust server. The working process for the call-back mechanism is described as follows:

(1) In a working session, a list is created to store the references of clients that have joined the session.
(2) With an input of parameters for a feature, a parameter event is generated in a client. Through invoking one of the server methods - *push_Event()*, such an event is received and handled by the server. After an object event is created and

ready for broadcasting from the server, each client recorded in the reference list is activated to receive the event by invoking one of the clients' methods - *receive_Event()*. A similar process can be carried out between the server and the required CAPP agent.

A common event interface, which extends Serializable

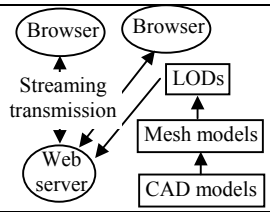
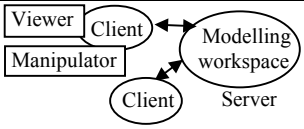
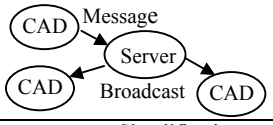
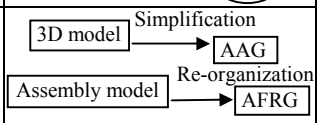
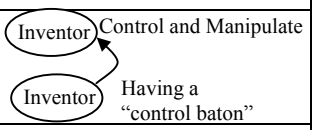
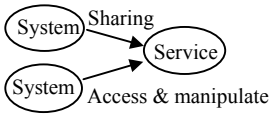
| Distributed Strategies | Characteristics | R&D Examples | Diagrams |
|---|---|---|---|
| Visualization and annotation of 3D CAD models | • The tools are primarily for visualization, annotation and inspection in a web environment. They do not support a real co-modelling activity.<br>• The CAD models are simplified as polygon mesh models for visualization.<br>• The mesh models are divided as different Levels of Details (LODs) for incremental transmission and display. | VizStream™, Hoops Stream Toolkit™, van den Berg *et al* |  |
| Co-design of 3D CAD models | • The tools provide collaborative facilities to support real co-design activities on 3D CAD models.<br>• The exchanged messages for maintaining the consistency of the co-design environment can be 3D models or creation/edition commands for 3D models.<br>• The sizes of the CAD models are usually huge. In some research, new kinds of feature or assembly representation schemes have been developed for distributed applications. | AlibreDesign™, GS-Design™, OneSpace™ |  |
| | | Nam and Wright, Qiang *et al* |  |
| | | Lee, Shyamsundar and Gadh |  |
| Services sharing for other systems | • The services or sub-modules of a system can be shared and manipulated by other systems.<br>• For the Inventor™ collaborative tool, at any one time, only an Inventor system has the "control baton" to design, and the controlled Inventor system is an observer. The "control baton" can be acquired and exchanged. | Inventor™ collaborative tools |  |
| | | Begole *et al*, Pahng *et al* |  |

**Table 1.** Summary of reported research and systems for distributed design

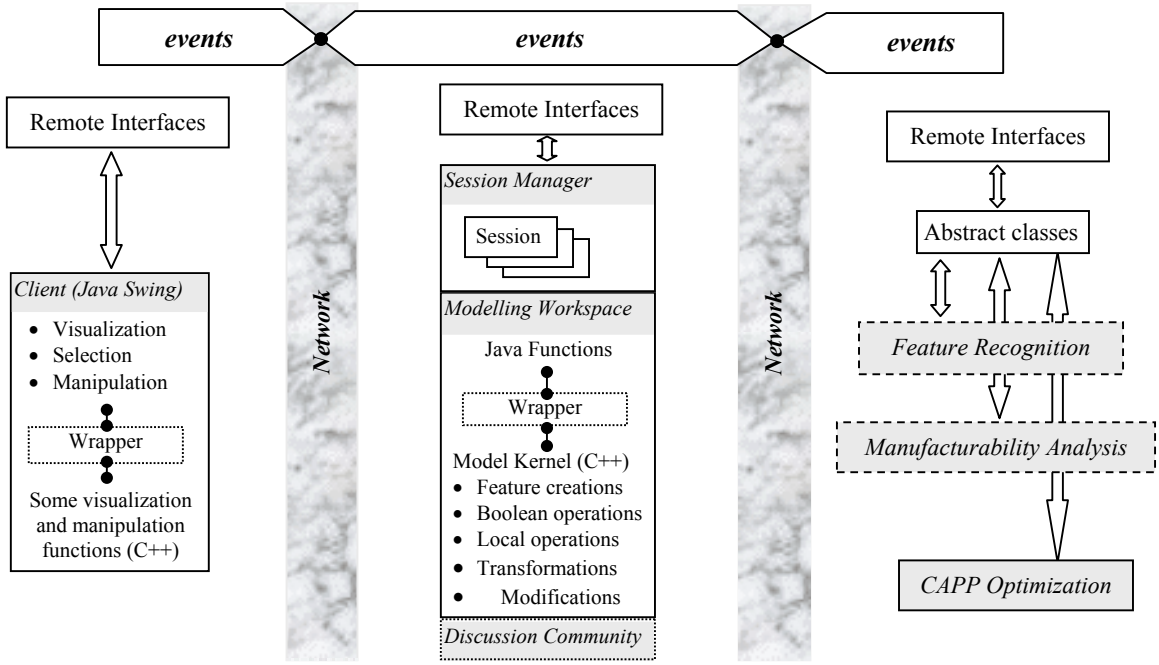



**Figure 1.** System framework of the distributed design environment

| Clients | Collaborative Server | Analysis Agents |
|---|---|---|
| *Parameter Input/Edition Interface*<br>• Input/edition of parameters for features<br>• Selection of entities in features<br>• Information queries for features and parts | *Session Manager*<br>• Dynamical generation of working sessions;<br>• Sharing of parts for clients within a working session | *Intelligent analysis and optimization modules*<br>• Machining feature recognition<br>• Manufacturability analysis.<br>• C APP optimization<br>• Providing abstract classes of machining feature recognition and manufacturablity analysis for future integration |
| *Visualization Environment*<br>• JFC-swing components to build the environment<br>• Visualization and manipulation of parts | *Modelling workspace*<br>• Acceptanace of parameters or entities from the clients<br>• Modelling parts based on OpenCasCade™ solid kernel<br>• Through Java Native Interface (JNI), the native API functions of the OpenCasCade™ can be invoked and manipulated by Java application | |
| | *Discussion Community*<br>• Provision of e-room for discussion and information (text and multimedia) exchange among clients | |
| *Communication*<br>The communication between each part is through an event-driven and call-back mechanism based on the Java RMI protocol | | |

**Table 2.** The functions in each part of the distributed environment

class for communication in the network, is defined. Based on this interface, four types of events have been implemented as follows:

(1) Parameter event for a design feature. This event, which is generated in a client, is used to wrap the parameters for a feature or a set of selected entities for local operations on an existing feature. This event is dispatched to the server for creating a feature represented as a B-Rep object.
(2) Object event for design features. This event wraps the features in the server to be sent back to the clients for visualization and manipulation.
(3) Object event for a design part. The feature objects of a design part are wrapped in this event, which is dispatched from a client to the CAPP service provider for analysis.
(4) Process plan event for a design part. This event is generated by the CAPP service provider to bind the generated process plans for a requested client.

The four types of events are defined in Table 3.

## 4. INTELLIGENT ANALYSIS AGENTS

The downstream manufacturing analysis modules can be wrapped as intelligent agents to be plugged in the system to support concurrent engineering design.

A three-layer architecture, including remote interfaces, abstract classes and class implementation, is designed for this purpose. With the definitions of the abstract classes for agents, some agents can be integrated later without re-initializing the whole system.

Currently, a hybrid genetic algorithm and simulated annealing CAPP optimization agent has been designed and integrated [Li *et al*, 2002(a)]. With such an agent, the activities of selecting machining resources, determining setup plans, and sequencing machining operations can be considered simultaneously so as to achieve the global lowest machining cost according to a combined evaluation criterion of machine costs, cutting tool costs, machine changes, tool and setup changes.

The other two agents, namely machining feature recognition and manufacturability analysis [Li, 2001], will be integrated later.

## 5. FEATURE RELATIONSHIP AND MANIPULATION

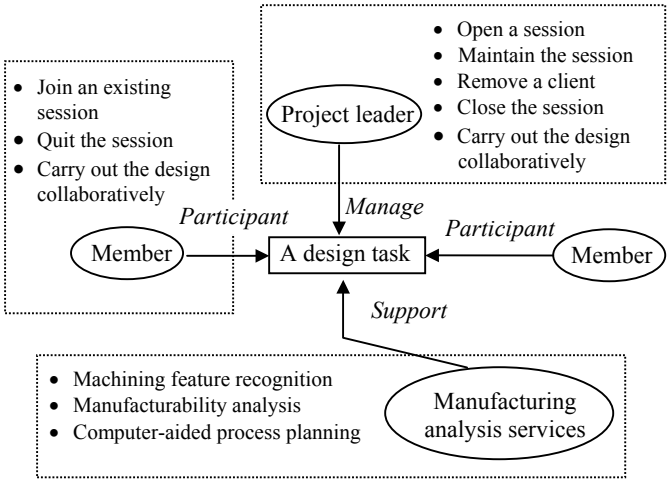



**Figure 2.** Organization of a design task in the environment

### *5.1 Feature-to-feature relationships*

In the environment, a client dynamically and interactively edits features in a part. Since a feature is usually associated with other features in the part, the edition operation on a feature might cause variations in other features in the part. In order to speed up the transmission of the designed model efficiently via the network with limited bandwidth, the varied information during the edition process should be differentiated in the server side and synchronized with the unchanged information in the client side. For this purpose, a feature-feature relationship should be studied.

The relationships between features can be categorized as interacting and non-interacting relationships. The interacting relationships between features include adjacency, overlapping, nesting and constraining [Li *et al*, 2002(b)]. The relationships are defined in Table 4, in which the symbols are as belows. Several cases for these definitions are illustrated in Figure 4.

$FE_1$, $FE_2$ - Two volumetric features in a part $P$

I $\not\subset$ $\subset$ - Regularized Boolean interaction, a proper subset, not a proper subset

| Tags | Remarks |
|---|---|
| Feature_Type | Type of the feature |
| Feature_ID | Unique ID of the feature |
| Feature_Parent | Parent feature of the feature |
| Feature_Parent_ID | Unique ID of the parent feature |
| Feature_Parameters | Form parameters of the feature |
| Position_Parameters | Position parameters of the feature |

(a) *Parameter event for a design feature*

| Tags | Remarks |
|---|---|
| Feature_Types[] | Type of the features |
| Feature_IDs[] | Unique IDs of the features |
| Feature_Parents[] | Parent features of the features |
| Feature_Parent_IDs[] | Unique IDs of the parent features |
| Feature_Objects[] | Objects of the features |
| Feature_Properties[] | Properties of the feature objects |
| Del_Feat_IDs[] | Unique IDs of the deleted features |

(b) *Object event for design features*

| Tags | Remarks |
|---|---|
| Feature_Properties[] | Properties of features in the part |
| Feature_Matrix[][] | Feature relation matrix |

(c) *Object event for a design part*

| Tags | Remarks |
|---|---|
| Process Plans | Generated process plans |
| Machining Cost | Machining cost for the plan |

(d) *Process plan event for a design part*

**Table 3.** Four types of events in the distributed environment

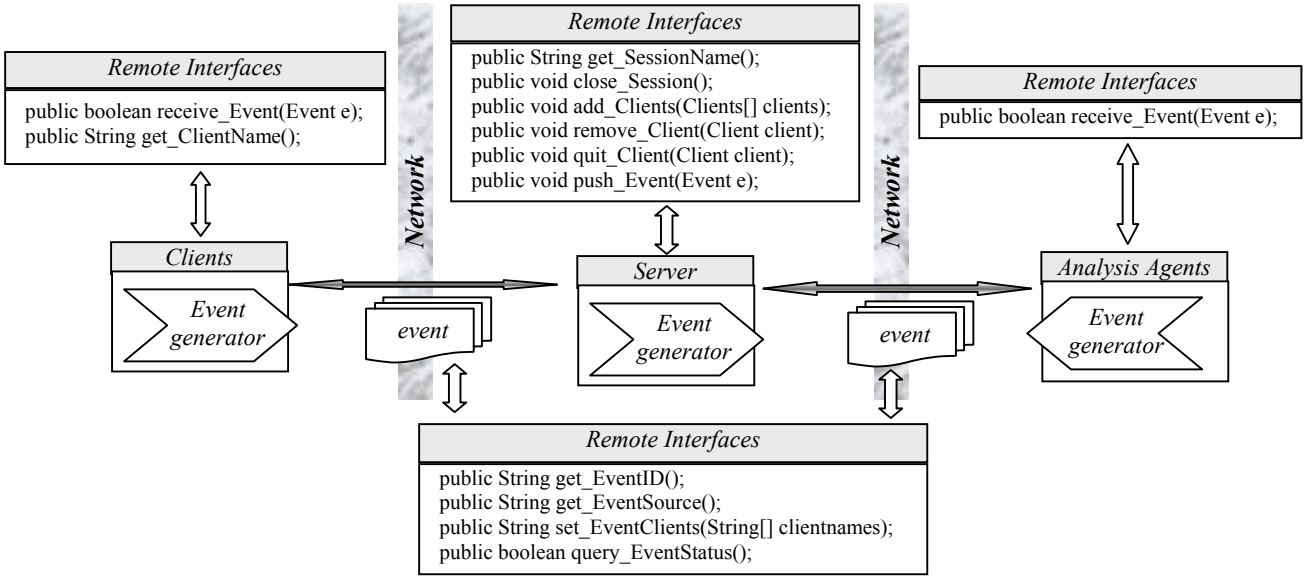



**Figure 3.** Remote interfaces in the distributed environment

$$P' = \begin{cases} P - FE_1 - FE_2 & FE_1, FE_2 \text{ are positive} \\ P - FE_1 \cup FE_2 & FE_1 \text{ is positive, } FE_2 \text{ is negative} \\ P \cup FE_1 - FE_2 & FE_1 \text{ is negative, } FE_2 \text{ is positive} \\ P \cup FE_1 \cup FE_2 & FE_1, FE_2 \text{ are negative} \end{cases}$$

$\partial$ - Boundary set of a volumetric feature

### *5.2 Feature manipulation operations*

For the different edition operations on a feature (adding features, deleting features, or modifying parameters of features), the feature-feature relationships associated with this feature are set up and maintained. The processes are described as follows:

(1) When a feature is *New_Added*, its interacting and non-interacting sets are set up according to the definitions. The features in the interacting set are *Updated*.

(2) When a feature is *Deleted*, its nesting features are *Deleted*. Other features in its interacting set will be differentiated as two types: features with changes in volume or boundary are *Updated*, or features without changes in volume and boundary are *Unchanged*. The features in the non-interacting set are *Unchanged*.

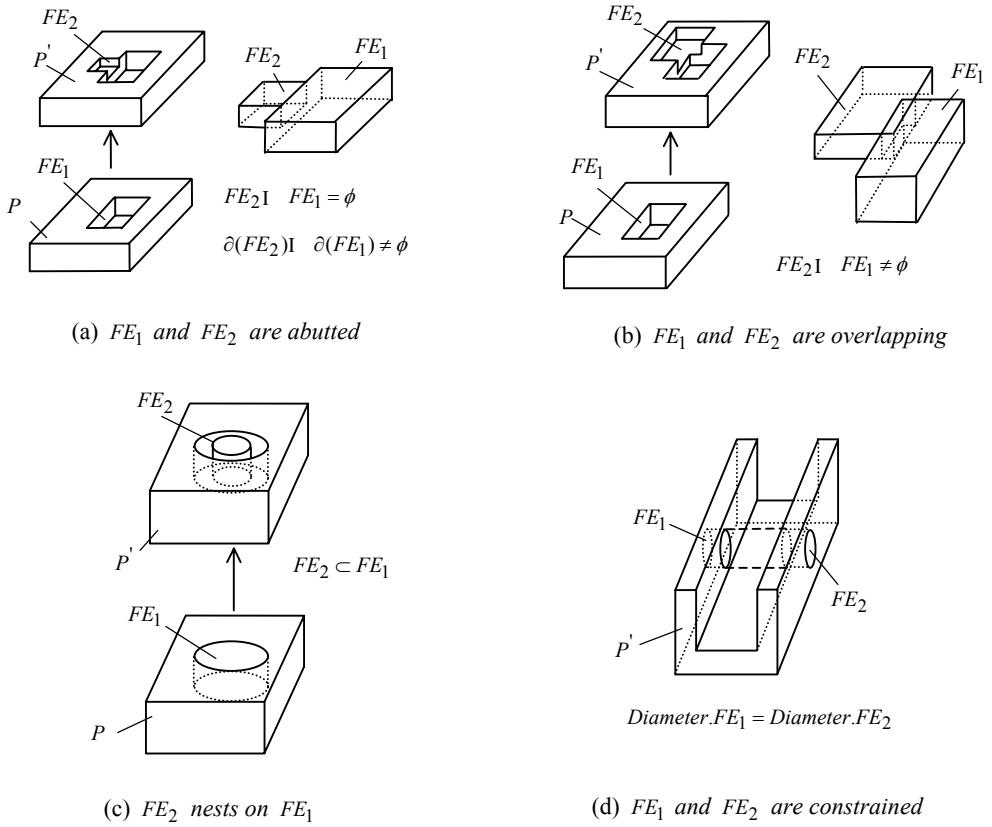
(3) When the parameters of a feature are modified, the process consists of three steps:
   (a) The features in its non-interacting set that have adjacency, overlapping or nesting relationship with this feature due to the modification are *Updated*;
   (b) The features in its interacting set will be differentiated as two types: features with changes in volume or boundary are *Updated*, and features without changes in volume and boundary are *Unchanged*;
   (c) If this feature has a constraining feature, the interacting and non-interacting features of the constraining feature will be adjusted according to the above steps.

For features that are *New_Added* and *Updated* differentiated from above process, their unique IDs, feature objects and other properties are wrapped into the object event for design features. For features that are *Deleted*, only their IDs are recorded in the event for clients to erase the relevant information. The *Unchanged* features will be kept the same in the clients. The scenario is shown in Figure 5.

### *5.3 A case study to illustrate the updated process*

| Relationships | Definitions |
|---|---|
| $FE_1$ and $FE_2$ are *adjacent* | $FE_1 \cap FE_2 = \phi$ , $\partial(FE_1) \cap \partial(FE_2) \neq \phi$ , $\partial(FE_1) \cap \partial(P') \neq \phi$ , $\partial(FE_2) \cap \partial(P') \neq \phi$ . |
| $FE_1$ and $FE_2$ are *overlapped* | $FE_1 \cap FE_2 \neq \phi$ , $FE_1 \not\subset FE_2$ and $FE_2 \not\subset FE_1$ . |
| $FE_1$ *nests* $FE_2$ | If both of $FE_1$ and $FE_2$ are either negative or positive: $FE_1 \cap FE_2 = \phi$ , $\partial(FE_1) \cap \partial(FE_2) \neq \phi$ , $\partial(FE_1) \cap \partial(P') = \phi$ . |
| | If one of $FE_1$ and $FE_2$ is negative, and another one is positive: $FE_1 \subset FE_2$ . |
| $FE_2$ *nests* $FE_1$ | If both of $FE_1$ and $FE_2$ are either negative or positive: $FE_1 \cap FE_2 = \phi$ , $\partial(FE_1) \cap \partial(FE_2) \neq \phi$ , $\partial(FE_2) \cap \partial(P') = \phi$ . |
| | If one of $FE_1$ and $FE_2$ is negative, and another one is positive: $FE_2 \subset FE_3$ . |
| $FE_1$ and $FE_2$ are *constrained* | There are some constraints between the entities in $FE_1$ and $FE_2$ . |

**Table 4.** The relationships between two interacting features - $FE_1$ and $FE_2$




(a) $FE_1$ *and* $FE_2$ *are abutted*

(b) $FE_1$ *and* $FE_2$ *are overlapping*

(c) $FE_2$ *nests on* $FE_1$

(d) $FE_1$ *and* $FE_2$ *are constrained*

**Figure 4**. Examples of relationships between interacting features - $FE_1$ and $FE_2$

An example case is illustrated in Figure 6 (a). The feature-to-feature relationships for the part are shown in Figure 6(b). There is a constraint between the widths of slots 1 and 2. With the shrinking operation of slot 2 in Figure 6(c), the relationship between the feature and the features in its non-interacting set remain the same and are labeled as *Unchanged*. The features in its non-interacting set are *Updated*. Since slots 1 and 2 have the same interacting set, with the variation of slot 1, the features in the interacting set of slot 1 will not be handled further. The final result is shown in Figure 6(d).

## 6. CONCLUSION

A distributed design environment based on 3D feature-based modelling and Java technologies has been proposed and developed. The advantages of the approach include:

(1) The environment can simulate a practical teamwork situation through creating and managing dynamic sessions, in which clients can play different roles in the design task.

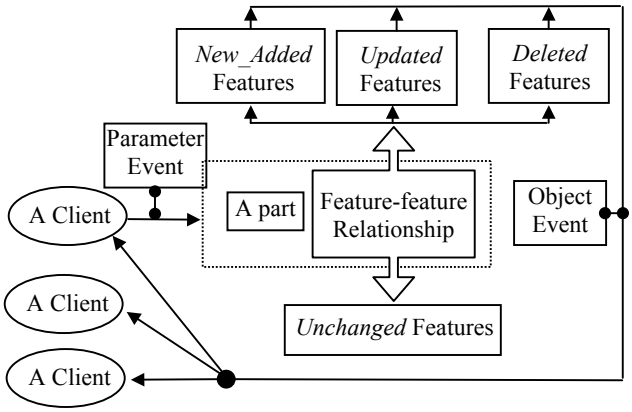



**Figure 5.** Differentiation of features during a part edition process

A high-performance and robust server is established through an event-driven and call-back mechanism;

(2) The design information based on 3D feature-based modelling technology can be exchanged and updated in the environment efficiently through a feature-to-feature relationship and manipulation method. The system is open and scaleable, and additional analysis modules can be incorporated by implementing some pre-defined abstract classes.

## REFERENCES


Alibre Design™ 2001, Alibre Inc. www.alibre.com

Begole J, Struble CA and Shaffer CA, 1997, Leveraging Java applets: toward collaboration transparency in Java. *IEEE Internet Computing*, 1-2, pp. 57-64.

GS-Design™ 2000, Technical whitepaper - an introduction to GS-Design beta, CollabWare Inc. www.prodeveloper.net/downloads/whitepaper.pdf

Hoops Stream Toolkit ™ 2001, Hoops Inc. www.hoops.com

Inventor™ collaboration tools, 2001, Autodesk Inc. www.autodesk.com

Lee JY, Kim H, Han SB and Park SB, 1999, Network-centric feature-based modeling. *Proceedings of Pacific Graphics'99*, South Korea, pp. 280-289.

Li WD, 2001, *Intelligent Synthesis of Product Design and Manufacturing Processes in a CE Environment*. Ph.D Thesis, National University of Singapore.

Li WD, Ong SK and Nee AYC, 2002(a), A hybrid GA-SA approach to optimize process plan for prismatic parts. *International Journal of Production Research*, in press.

Li WD, Ong SK and Nee AYC, 2002(b), Recognizing manufacturing features from a design-by-feature model. *Computer-Aided Design*, in press.

Nam TJ and Wright DK, 1998, CollIDE: a shared 3D workspace for CAD. *Proceedings of the 1998 Conference on Network Entities*, Leeds, UK.

OneSpace™ 2000, Technical whitepaper - sharing engineering, CoCreate Inc. www.cocreate.com/onespace/documentation/whitepapers/shared_eng.pdf

OpenCasCade™ 3D Modelling Kernel V3.0, 2001, www.opencascade.com

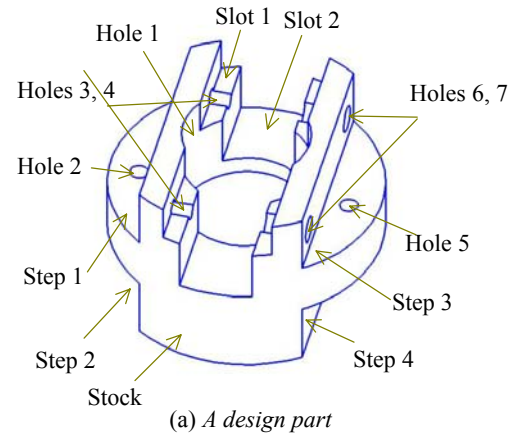
Pahng GD, Bae S and Wallace D, 1998, Web-based collaborative design modeling and decision support. *Proceedings of 1998 ASME Design Engineering Technical Conference*, Atlanta, US.

Qiang L, Zhang YF and Nee AYC, 2001, A distributed and collaborative concurrent product design system through the WWW/Internet. *International Journal of Advanced Manufacturing Technology*, 17(5), pp. 315-322.

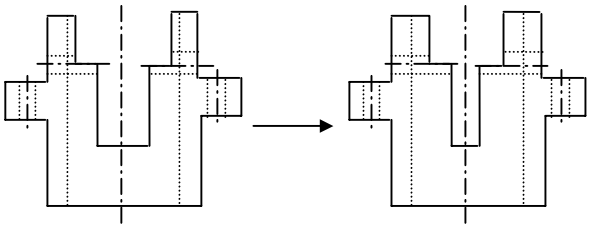
Shyamsundar N and Gadh R, 2001, Internet-based collaborative product design with assembly features and virtual design spaces. *Computer-Aided Design*, 33, pp. 637-651.

van den Berg E, Bidarra R and Bronsvoort WF, 2000, Web-based interaction on feature models. *Proceedings of the Seventh IFIP WG 5.2 Workshop on Geometric Modelling: Fundamentals and Applications*, Parma, Italy.

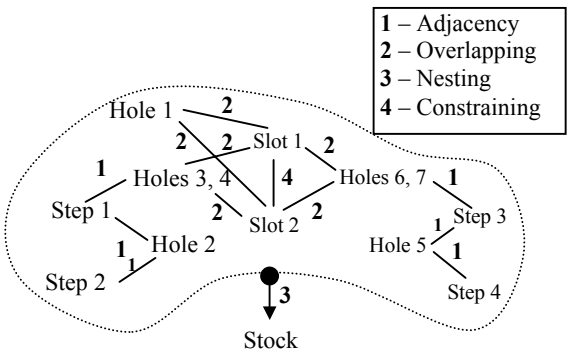
VizStream ™ 2001, RealityWave Inc. www.realitywave.com


(a) *A design part*



(c) *The width of Slot 2 is shrunk*




(b) *The feature-feature relationship*

*Unchanged* features: Steps 1, 2, 3, and 4, Holes 2 and 5
*New_Added* features: No
*Deleted* features: No
*Updated* features: Stock, Slots 1 and 2, Holes 1, 3, 4, 6 and 7

(d) *The unchanged and changed features*

**Figure 6.** A case part containing the differentiated features