A Collaborative Framework for Integrated Part and Assembly Modeling

Rafael Bidarra, Niels Kranendonk, Alex Noort and Willem F. Bronsvoort
Faculty of Information Technology and Systems
Delft University of Technology
Mekelweg 4, NL-2628 CD Delft, The Netherlands
Telephone: +31 15 2784564
http://graphics.tudelft.nl
(Bidarra/Noort/Bronsvoort)@its.tudelft.nl

ABSTRACT

An ideal product modeling system should support both part modeling and assembly modeling, instead of just either of them as is the case in most current CAD systems. A good basis for such integration is multiple-view feature modeling, as it allows focusing on different aspects of the product, while at the same time maintaining the consistency among all model views.

This paper presents a framework that supports synchronous collaborative sessions via the Internet, among members of a distributed development team, with such a modeling system. The framework provides facilities for creating a hierarchical product structure, with single and compound components, and meanwhile assigning tasks to team members. The actual design of a single component is supported by a web-client specialized in part design, whereas the specification of assembly relations among components is supported by a web-client specialized in assembly design.

All clients make use of the same server, which runs a multiple-view feature modeling kernel and maintains the complete product model, guaranteeing the consistency between the part design and the assembly design views. In addition, the server keeps all clients up to date and manages all communication.

Categories and Subject Descriptors

J.6 [Computer-Aided Engineering]: Computer-aided design (CAD), Computer-aided manufacturing (CAM); I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling — modeling packages

General Terms

Algorithms, Design

Keywords

Collaborative design, integrated product modeling, part modeling, assembly modeling, feature modeling

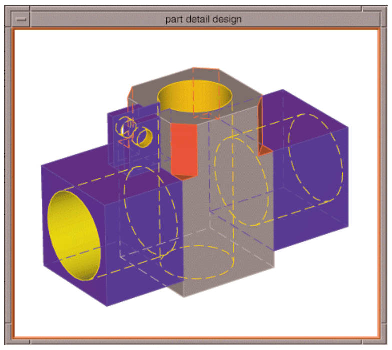
1. INTRODUCTION

Current modeling systems adequately support either modeling of parts or modeling of assemblies, usually by a single user only. In particular for complex products, this leads to a lot of undesirable exchange of product data, back and forth, between different users and systems. Future modeling systems should therefore support integrated modeling of parts and assemblies by a team of development engineers.

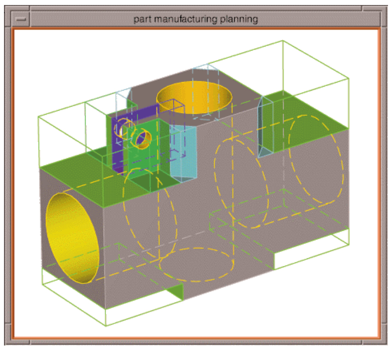
Multiple-view feature modeling is a good basis for integrated modeling of parts and assemblies [15]. It offers different views on a product for parts and assemblies, each view containing a specific feature model. For each part, there is a part detail design view, describing the part in terms of design form features, and a part manufacturing planning view, describing the part in terms of manufacturing form features. In addition, there is an assembly design view for the whole product, describing one or more assemblies in terms of components and connection features between these components. The part and assembly views are kept consistent, i.e. changes in one view on a part are propagated to the other view on the part, but also to the assembly design view, and changes in the assembly design view are propagated to the part views that are involved. In this way, real integration of part and assembly modeling is achieved.

The approach to integrated modeling just described can be very helpful to a single engineer who is responsible for the development of a complete product [6]. In practice, however, usually several engineers are involved in the development of a product. This is obvious for complex products consisting of many parts, but can even be the case for relatively simple products. In both cases, one can think of different experts for the design of the parts, the manufacturing planning of the parts, and the assembly design of the complete product. Indeed, product development teams nowadays involve more and more engineers. Three essential characteristics of the working procedure of such teams can be summarized as follows:

- team members, often geographically distributed, need to work on (at least part of) the same product data;
- due to outsourcing of some components, team members may even be scattered over different companies, each one with its own design practice, CAD tools and data formats, further complicating the previous aspect;
- collaboration among team members plays an increasingly important role in solving design conflicts as early as possible



(a)



(b)

Figure 1. Geometry camera windows for detail design view (a) and manufacturing planning view (b) on a part that is used in a bicycle frame.

in the design stage, but travel time and costs, and other well-known inconveniences, prohibit frequent physical meetings.

Due to their interdependencies, it is not likely that conveniently supporting these characteristics can be achieved by three separate, independent solutions. Instead, synchronous collaborative modeling sessions via the Internet are gaining attractiveness, as they indeed allow several team members to remotely coordinate their design work and discuss design issues of mutual relevance.

However, systems that support real collaborative design of parts or of assemblies are hardly available, let alone systems that support collaborative, integrated design of parts and assemblies.

In this paper, a collaborative framework is presented that does support integrated design of parts and assemblies. It supports synchronous collaborative sessions via the Internet, among several members of a product development team. There are specialized web-clients for part detail design and part manufacturing planning, and for assembly design. All clients make use of the same server, which runs a multiple-view feature modeling kernel, and takes care of all communication. This paper focuses on the top-down product structuring and task assignment and on the actual collaborative part and assembly modeling, but also describes the most salient aspects of the client-server communication.

Section 2 summarizes the underlying approach to integrated part and assembly modeling. Section 3 discusses the state of the art in collaborative modeling and introduces the new collaborative framework. Section 4 elaborates the product structuring and task assignment, and Section 5 the collaborative part and assembly modeling. Section 6 describes the data exchange between the clients and the server. Section 7 presents some conclusions.

2. INTEGRATED PART AND ASSEMBLY MODELING

Noort *et al*. [15] have recently presented an integrated approach to part and assembly modeling. This approach is the basis for the collaborative modeling framework presented in this paper, and is therefore summarized in this section.

Most commercial modeling systems now adequately support part modeling. These systems are typically feature modeling systems, allowing a user to store functional information on the parts in a model. However, such systems offer only limited facilities to represent assembly information [8]. The relations between the components in an assembly usually have to be specified using low-level relations, such as mate and align. In addition, these systems usually provide only a single interpretation of the product for both part and assembly design, whereas part and assembly design focus on different aspects of the product. There are some research systems that adequately support modeling of assembly aspects of a product, but their models are, in turn, less suitable to support part design [9, 10, 24].

The main problem of having separate part and assembly modeling systems is that part-oriented requirements cannot be automatically checked during assembly design, and vice versa. To be able to perform such checks, information has to be exchanged from one system to the other, sometimes by hand, possibly leading to inconsistency of the models in the two systems. A solution to this problem is the approach to integrate part and assembly modeling summarized here. It supplies the user with the functionality of both a part modeling system and an assembly modeling system, maintains integrated part and assembly models, and thus solves the problems of data exchange and inconsistency.

This approach is based on the *multiple-view feature modeling* concept, which provides specialized interpretations of a product for different product development phases by means of views. Each *view* has its own feature model of the product, with features relevant for the corresponding development phase. Here, there are a detail design view and a manufacturing planning view for each part, and an assembly design view for the whole product. All views are kept consistent, i.e. changes in one view are automatically propagated to the other views. The three types of views, and the way these are kept consistent, will be shortly described now.

The feature model of a part detail design view contains instances of form feature classes present in the feature library for part detail design. A form feature class contains a generic feature shape, and possibly several constraints that have to be satisfied for all instances of the class, e.g. for a hole feature that the radius should

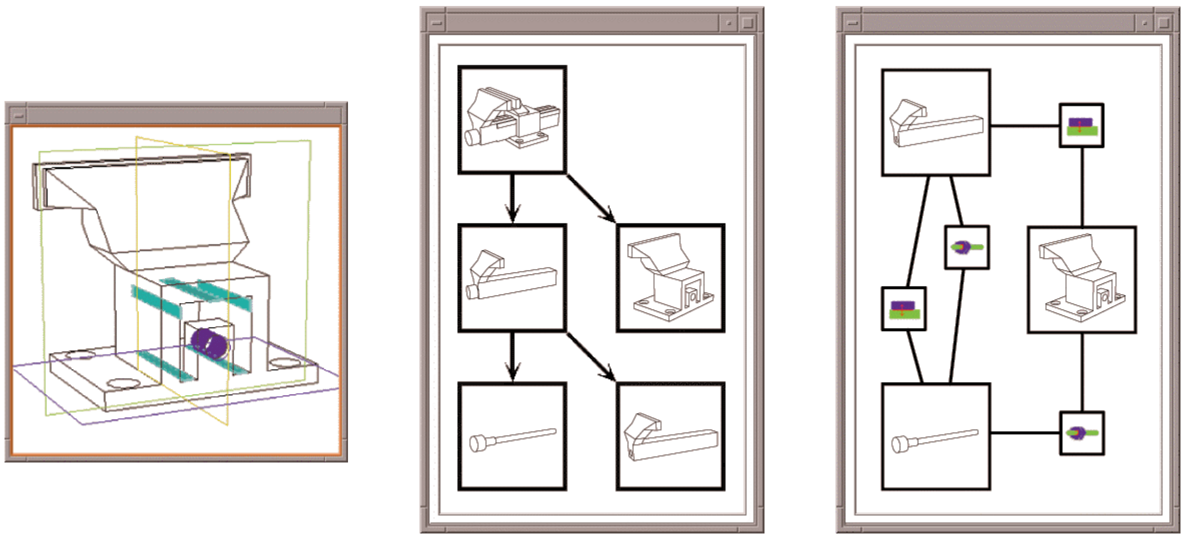

(a) (b) (c)

Figure 2. Windows of a geometry camera for a component of a bench vice (a), a hierarchical graph camera (b) and a relational graph camera (c) for the complete bench vice assembly.

be within some interval and that its entrance face should remain open. In addition to these feature constraints, there can be *model constraints* on one or several form feature instances, e.g. that two slots should be at some prescribed distance.

To visualize the feature model of a part detail design view, one or more geometry cameras can be used. A geometry camera can use several line visualization and shading techniques, in various combinations [5]. It can provide insight into the feature model by visualizing all sorts of engineering information, e.g. highlighting all features of a specific class, all closure faces of subtractive features or all intersections of features. See Figure 1(a) for a geometry camera window for a part detail design view.

To create the feature model of a part detail design view, modeling operations are available to add, remove and change the parameters of a feature or a model constraint. After each operation, the validity of the model is checked on the basis of all feature and model constraints in the model. If the model is no longer valid, i.e. any of these constraints is violated, the user is assisted in making it valid again [3].

The feature model of a part manufacturing planning view is similar to that of a part detail design view, but with instances of form feature classes present in the feature library for manufacturing planning. Whereas the library for part detail design contains both additive (material adding) and subtractive (material removing) features, the library for manufacturing planning contains only subtractive features. Again geometry cameras can be used to visualize the feature model of a manufacturing planning view; see Figure 1(b) for a geometry camera window for the manufacturing planning view for the same part as shown in Figure 1(a). Although the manufacturing planning view of a part will usually be derived from the part detail design view, it is also possible to directly modify the feature model of the manufacturing planning view to improve manufacturability of the part. For the latter, the same modeling operations are available as for the part detail design view.

The feature model of the assembly design view contains components and connection features between components; a set of components connected by connection features forms an assembly. A component is either a single component or a compound component. A single component represents a part in the assembly design view. A compound component encapsulates an assembly for further assembly modeling operations, by hiding its internal structure of components and connection features, and dealing with the boundary of the assembly only. A connection feature is an instance of a connection feature class present in the feature library for assembly design. A connection feature class contains a description of the types of the form features needed on the components for the connection, several constraints that specify the relations between the components, such as the internal freedom of motion, but also the way the connection can be established [10]. A connection feature instance determines the relative position and orientation between the components involved. Examples of connection features are a rib-slot and a pin-hole connection feature.

A component contains the reference geometry of the component, i.e. the boundary of the part or assembly related to that component, and in addition the form features of the connection features on the component. So, only the regions of the component that are relevant for assembly are described by form features, but the rest of the component is not.

To visualize the feature model of the assembly design view, both geometry cameras and graph cameras can be used [5]. A geometry camera shows the reference geometry of a component with lines, and the form features of the connection features on the component with shaded faces (see Figure 2(a)). Graph cameras show the structure of an assembly. A hierarchical graph camera shows the

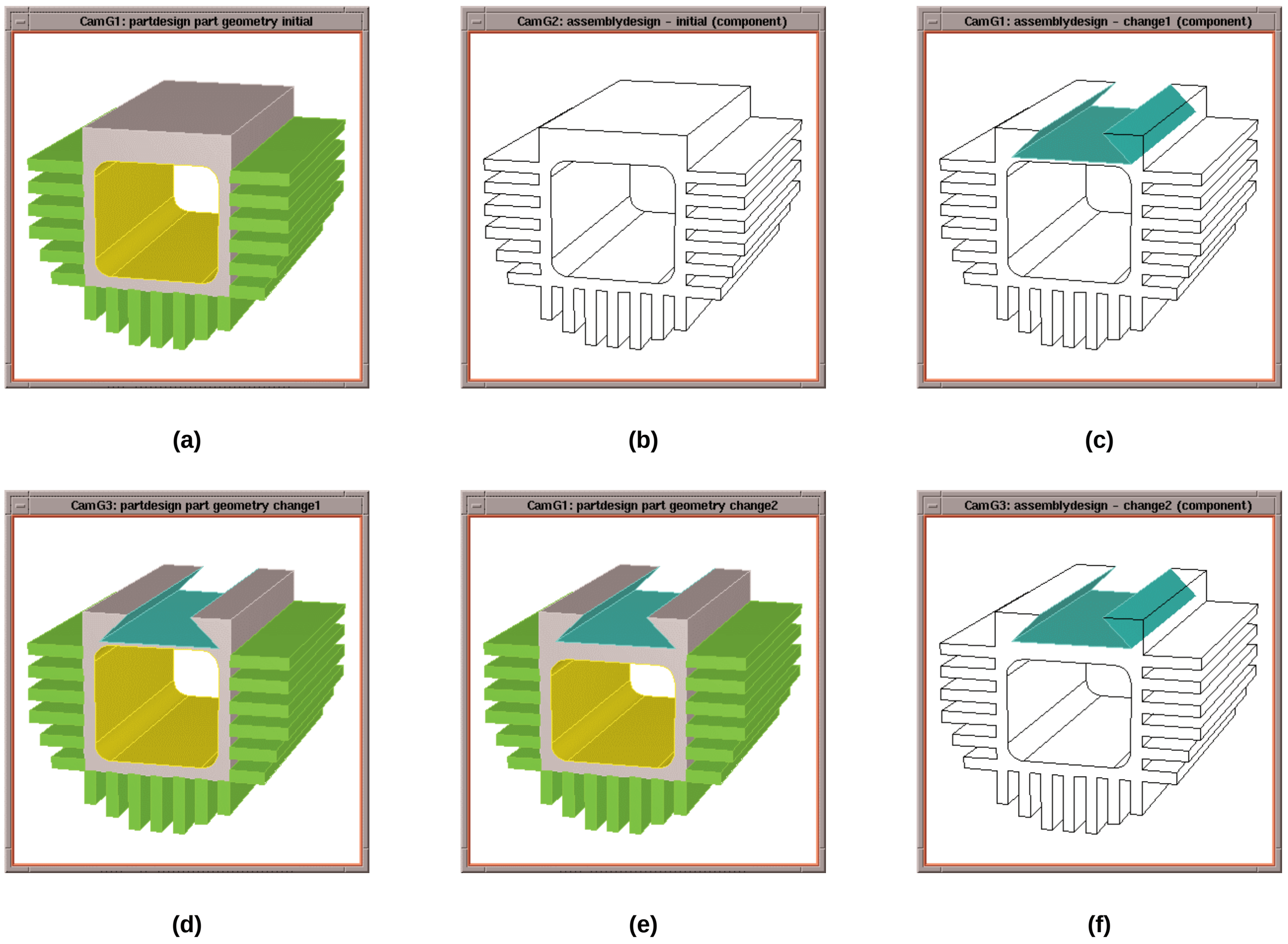

CamG1: partdesign part geometry initial
CamG2: assemblydesign – initial (component)
CamG1: assemblydesign – change1 (component)
(a)
(b)
(c)
CamG3: partdesign part geometry change1
CamG1: partdesign part geometry change2
CamG3: assemblydesign – change2 (component)
(d)
(e)
(f)

Figure 3. If a detail design view on a part (a) and the assembly design view on the related component (b) become inconsistent because a form feature for a connection feature is added to the component (c), then the feature model of the part is updated accordingly (d). The resulting invalid feature model of the part is made valid again by reducing the size of the passage (e), thus increasing the distance between the passage and the new form feature, and the reference geometry of the component is updated accordingly (f).

hierarchy of an assembly with its components and their subcomponents (see Figure 2(b)). A relational graph camera shows the connections between the components (see Figure 2(c)).

To create the feature model of the assembly design view, operations are available to add a connection feature between different components, to change the parameters of a connection feature, to remove a connection feature, to make a compound component out of an assembly, and to turn a compound component back into an assembly. Adding a connection feature between components requires the appropriate form features, e.g. a pin and a hole for a pin-hole connection feature, to exist on the components. If such a form feature does not yet exist in the assembly design view, it has to be created. If the shape for the form feature already exists on the reference geometry of the component, the form feature can be created by feature recognition; otherwise the form feature can be created by adding the form feature, including its shape, to the reference geometry.

The part detail design views, the part manufacturing planning views and the assembly design view are kept consistent, by automatically propagating changes made in one view to the other views. The detail design view and the manufacturing planning view on a part are kept consistent by feature conversion [12]. The part detail design views and the assembly design view are kept consistent by linking the part models to the related single components in the assembly model. When a form feature for a connection feature is added to some component, and this changes the shape of the component, this change is propagated to the feature model of the related part in its detail design view by feature conversion again. When a part detail design view is changed, the reference geometry of the related single component in the assembly design view is updated. See Figure 3 for an example of keeping several views consistent.

Integration of detail design and manufacturing planning of parts, and assembly design of the whole product, can be very profitable. It enables requirements on parts to be taken into account during assembly design, by propagating changes made in the assembly design view to the relevant part detail design views, and indirectly to the related part manufacturing planning views, where requirements on the part can be checked. The other way around, it also enables requirements on assemblies to be taken into account during part design, by propagating changes made in the part detail design or the part manufacturing planning view to the assembly design view, where requirements on the assembly can be checked. For the checking, or more in general for the modeling, it is advantageous to provide specialized views on a product for part and assembly modeling, because these focus on those product aspects that are relevant for their type of modeling.

The concept of integrated part and assembly modeling can already be very useful to support a single engineer in the development of a complete product, but will become even more valuable when it is made available to a team of engineers involved in the development of a product. This is the subject of the rest of this paper.

3. COLLABORATIVE PRODUCT MODELING

In this section, we first analyze proposals that have emerged so far in the area of collaborative product development, and then describe the main characteristics of our new collaborative framework.

3.1 State of the Art

All three aspects of the working procedure in current development teams mentioned in Section 1 are very poorly supported, if at all, by current CAD systems. So far, only a small number of tools have been developed that somehow support collaborative design activities. For example, tools for collaborative model annotation and visualization via Internet are now becoming available, providing concepts such as shared cameras and telepointers [11, 23]. However, such tools are primarily focused on inspection, using simple polygon mesh models, and do not support real modeling activities. In other words, they are valuable assistants for teamwork, but no real CAD systems.

Some recent efforts have explored the possibility of enhancing existing CAD systems with collaborative facilities. For example, several commercial CAD systems are now offering functionality for multi-user, token-based *asynchronous* manipulation of a CAD model [16, 17].

To the best of our knowledge, there are only two commercial system currently offering some *synchronous* collaborative modeling facilities: OneSpace [7], with a client-server architecture; and Alibre Design [1], with a peer-to-peer architecture, involving Alibre CAD stations. Both systems are severely constrained by the model format into which they convert all shared CAD models.

The requirements for concurrency and synchronization in a collaborative modeling context lead almost inevitably to the adoption of a *client-server* architecture, in which the server provides the team members not only with the indispensable communication, coordination and data consistency tools, but also with the necessary modeling facilities. In client-server systems it is important to balance the complexity of the client application and the network load. In a collaborative modeling context, client complexity is mainly determined by the modeling and interactive facilities implemented at the client, whereas network load is mainly a function of the kind and size of the model data being transferred to/from the clients. Some collaborative modeling prototype systems follow a *fat client* scheme [14, 19]. Fat clients are able to manipulate their local copy of the model data. This choice leads to good interactive and visualization facilities, but comes at the cost of a rather heavy network load due to the frequent synchronization of model data among clients. Furthermore, fat clients are typically platform-dependent applications that require more complex installation and maintenance procedures, and are therefore less practical in a multi-platform environment, in particular across various enterprises. Other prototype systems follow a *thin client* scheme [4, 13, 18]. Thin clients can profit from the use of feature models at the server, where all modeling operations are performed. A limited amount of model data, required at the clients for real-time display, navigation and interaction, is derived at the server and broadcast incrementally to the clients, thus keeping the network load at acceptable levels.

For a collaborative modeling system to be successful, it should combine a good level of interactivity with the sort of visualization typically provided by conventional CAD systems. Users will not be able to design adequately if they have to wait a long time after every operation. But increasing interactivity by just porting more and more data and functionality to the clients is not a good solution either, as synchronization problems would then become critical. A web-based client-server approach is more appropriate in such contexts.

The prototype system webSPIFF described by Bidarra *et al.* [4] is a system that follows this approach. It provides collaborative *part modeling* capabilities to its clients, who can connect to the server to work together using the design view and/or the manufacturing planning view on a part.

The server has two main components: the SPIFF modeling system and the Session Manager. The SPIFF modeling system provides all feature modeling functionality, including multiple views on a part, and advanced visualization and validity maintenance of feature models. It maintains a central product model, which includes a cellular model for the geometric representation of a part, and canonical shapes representing the individual features in each view. The Session Manager provides functionality to start, join, leave and close a collaborative session, to coordinate the session, and to manage all communication between SPIFF and the clients. In particular, the Session Manager collects all operations requested by the various clients, and schedules them for execution at the SPIFF system.

webSPIFF clients perform operations locally as much as possible, e.g. regarding visualization of, and interaction with, their feature model, and only high-level messages, e.g. for specifying modeling operations, as well as a limited amount of model data necessary for updating the client information, are sent over the network. In particular, as soon as real feature model computations have to be executed, such as required by modeling operations, by conversion between feature views and by feature validity maintenance, these are executed at the server, on the central product model, and their results are eventually exported back to the clients. An important advantage of using a central product model in this architecture is

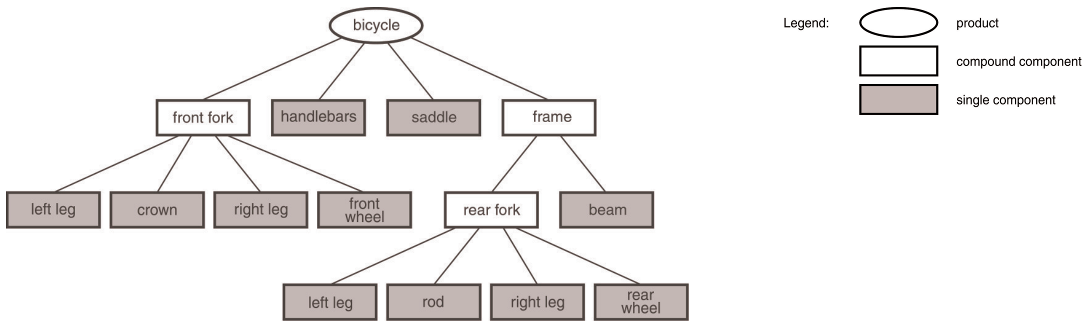


Figure 4. A simplified hierarchical product structure of a bicycle.

that inconsistencies among multiple versions of the model data at different clients are avoided.

The next subsection briefly introduces our new collaborative framework for integrated part and assembly modeling, which capitalizes on the webSPIFF facilities just described.

3.2 A New Collaborative Framework for Integrated Product Modeling

The new webSPIFF collaborative framework presented here substantially expands the capabilities referred to in the previous subsection in a number of ways.

First, webSPIFF now provides a scheme for hierarchically structuring a product into components. In this scheme, each component is assigned to one or more team members, responsible for its actual development.

Second, the webSPIFF server can now concurrently support several groups of users, each group collaboratively working on its own component of the product.

Third, besides modeling on part-oriented views, webSPIFF now also provides specialized clients with modeling capabilities for assembly design, e.g. to specify assembly relations between components, possibly developed by other team members, by means of connection features.

Finally, because at the server the SPIFF modeling system seamlessly integrates in its central product model the part- and assembly-oriented views described in Section 2, the propagation of model changes among components can now be fully exploited.

Before we elaborate in Section 5 how collaborative modeling takes place in this framework, we describe in Section 4 how it supports collaborative structuring of a product into components, and how tasks are assigned to members of the product development team.

4. TOP-DOWN PRODUCT STRUCTURING AND TASK ASSIGNMENT

To be able to assign tasks to the members of a development team, a product to be developed has to be structured in some way. We use the well-known hierarchical product structure for this. In line with the integrated approach for part and assembly design discussed in Section 2, a product consists of a number of components. Each component can either consist of a number of subcomponents or be a part; the first type of component is called a compound component, the second type a single component. Subcomponents in a component are related to each other by means of connection features. See Figure 4 for a simplified example of a hierarchical product structure.

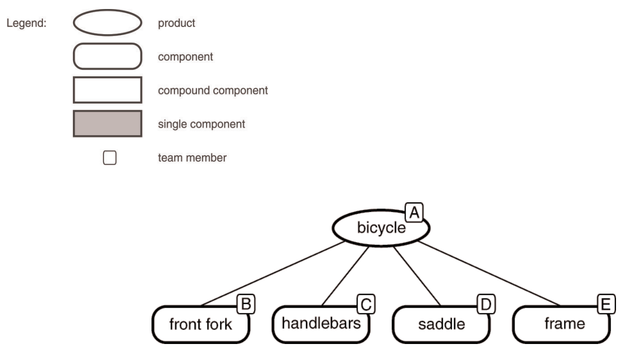
In our approach, product structuring and task assignment go hand in hand. A so-called principal product designer has to start up the hierarchical structuring process for a new product. He can do this by setting up a new product structure, i.e. giving a name to the product, specifying the main components the product consists of, and assigning these components to himself or to other team members (see Figure 5(a), left). This can be easily performed using the Product Navigator, which provides all functionality for building the product structure and visualizing it as it evolves (see Figure 5(a), right).

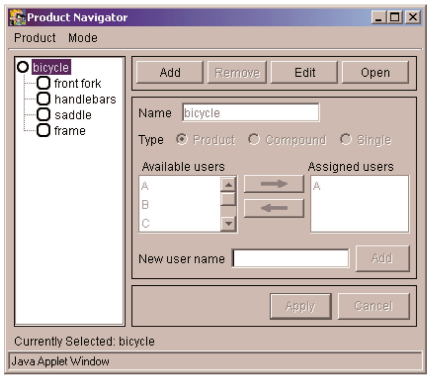
For each component, the team member to whom it was assigned has to specify whether it is a single component (see Figure 5(b)) or a compound component. In the latter case, he also has to specify the subcomponents it consists of, and assign each to a team member (see Figure 5(c)).

The product structuring continues recursively in this way, until all components at the lowest level in the hierarchy are single components (see Figure 5(d), which corresponds to the product structure of Figure 4). So the product is structured in a top-down way, creating as many levels as desired by the team members, and meanwhile the components are assigned to team members.

Notice that product structuring and task assignment are detached from the actual modeling of components. Part modeling of a single component can start as soon as it has been designated as such. Assembly modeling of a compound component can only start once its subcomponents have been modeled. In both cases, the Product Navigator reports the fact to the team member(s) to whom the component was assigned.

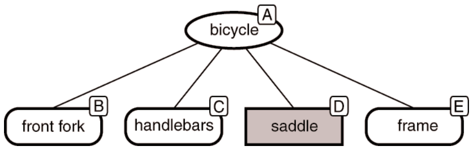
Several activities can be done simultaneously in the whole product development process. First, product structuring and task assignment can be done simultaneously for different branches of the product structure. Second, product structuring and task assignment can still be going on in certain branches of the product structure, while in other branches parts or even compound components are already being modeled. Third, modeling of different parts and compound components can also be done simultaneously. So, a product can be developed collaboratively in the sense that several team members can concurrently work on independent tasks.

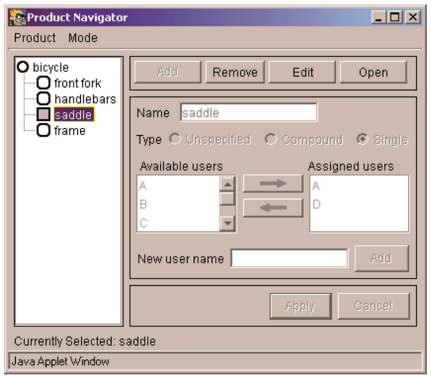




Name: bicycle
Type: [✓ Product] Compound Single
Available users: A, B, C
Assigned users: A
New user name:
Currently Selected: bicycle

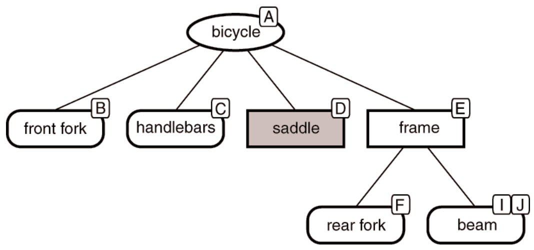
(a)

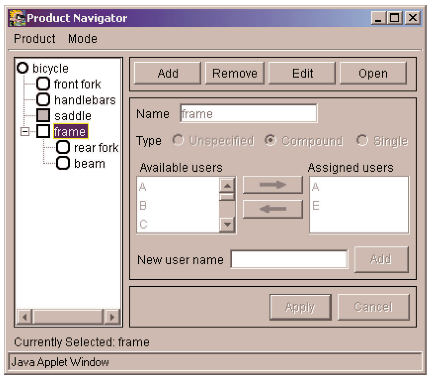




Name: saddle
Type: Unspecified Compound [✓ Single]
Available users: A, B, C
Assigned users: A, D
New user name:
Currently Selected: saddle

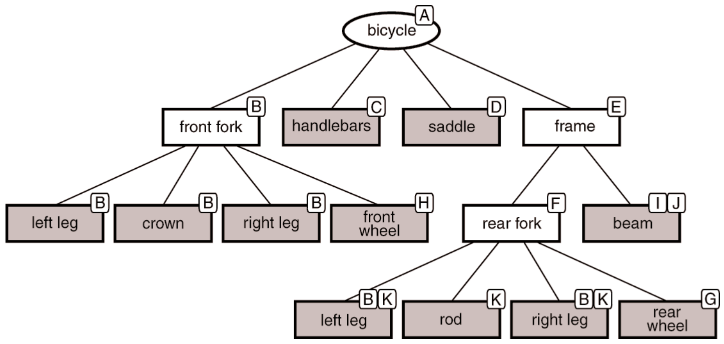
(b)

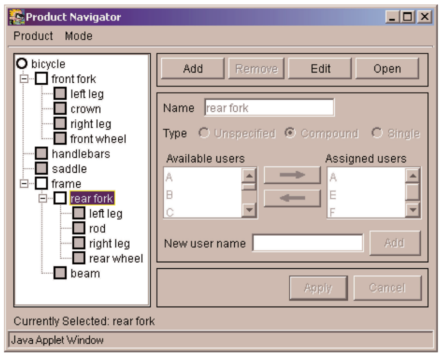




Name: frame
Type: Unspecified [✓ Compound] Single
Available users: A, B, C
Assigned users: A, E
New user name:
Currently Selected: frame

(c)

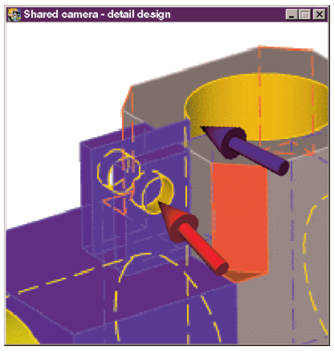




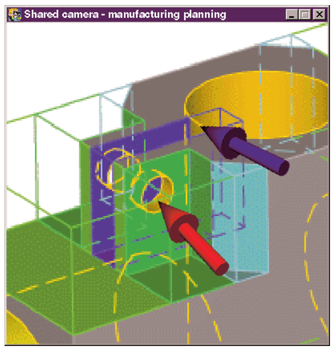
Name: rear fork
Type: Unspecified [✓ Compound] Single
Available users: A, B, C
Assigned users: A, E, F
New user name:
Currently Selected: rear fork

(d)

Figure 5. Product structuring and task assignment: new product structure (a), specification of a single component (b), specification of a compound component (c), and final product structure (d).



(a) detail design view



(b) manufacturing planning view

Figure 6. Shared cameras of two team members with different views on the same part.

However, the possibilities of collaboration go much further. In particular, it is possible that several team members are collaboratively working on the development of a same component. The concept of the modeling scope of a team member is important here. It is defined as the set of components he has modeling rights for, and contains the components assigned to him, supplemented with all their subcomponents in the product structure. So for team member E in Figure 5(d), it consists of the set {frame, rear fork, left leg, rod, right leg, rear wheel, beam}. This notion of modeling scope is based on the assumption that the development team is also hierarchically structured, and that a team member should have modeling rights for all subcomponents which in the end constitute a component assigned to him, regardless of whether these were assigned to him or to other team members. A team member can also extend the modeling scope of another team member, by granting the latter modeling rights for a component in his own modeling scope. For example, in order to exploit design similarities between the two wheels, team member B could grant modeling rights for the front wheel to team member G (not depicted in Figure 5(d)).

Two or more team members with modeling rights for a component can collaboratively model the component, in the sense that they have access to the corresponding feature model and can modify it in a synchronized way. The next section discusses how this actual modeling is done.

5. BOTTOM-UP PART AND ASSEMBLY MODELING

As mentioned in the previous section, a team member can start modeling a part as soon as it has been assigned to him. On the other hand, modeling of a compound component by the team member to whom it was assigned, can only start as soon as its subcomponents have been created. So the actual modeling activity is a bottom-up process, starting at the leafs of the hierarchical product structure.

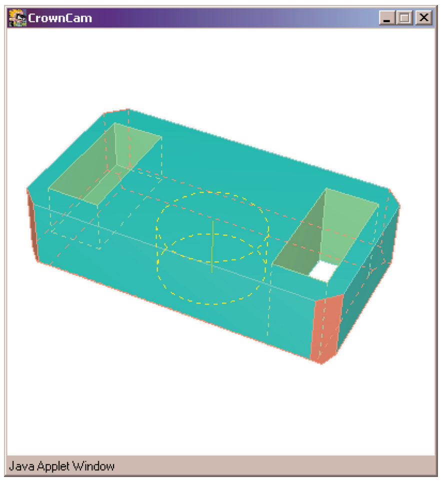
If two or more team members have modeling rights for the same part or compound component, they can collaboratively work on it. This is called *collaborative part modeling* and *collaborative assembly modeling*, respectively.

5.1 Collaborative Part Modeling

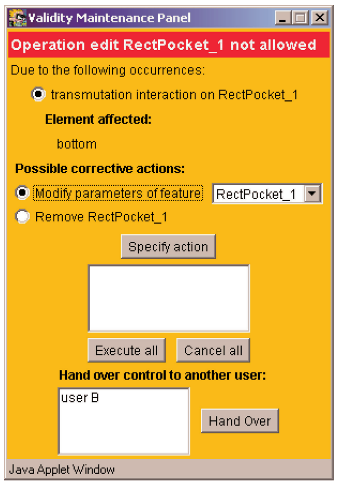
Team members with the appropriate modeling rights can work together on the same part, performing modeling operations available in their view: detail design view and/or manufacturing planning view; see Section 2. Typically each team member will have a geometry camera displaying the feature model of his view. After any of the team members has performed a modeling operation on the part, all the others will also have the model updated in their cameras. Because all of them have the same modeling rights, several modeling operations on the part may be concurrently specified and sent to the server. Such concurrency is handled at the server, by serializing the operations.

It may well occur, however, that modeling operations are conflicting, e.g. in the sense that an operation unintentionally cancels the effect of another operation. For this reason, webSPIFF encourages team members to coordinate their actions using some conferencing facility (phone, chat channel, etc.). In addition to this, webSPIFF provides team members with so-called shared cameras. All participants in a shared camera share the same viewing parameters on the visualized product geometry, possibly in different views (see Figure 6). These parameters are permanently synchronized, so that every time one user interactively modifies them, the shared cameras of the other users are automatically updated. webSPIFF also provides each user of a shared camera with a personalized telepointer [22]. The telepointers of all participants are constantly updated in all shared cameras. In this way, e.g. when discussing some model detail, participants in a shared camera can always trace back where each interlocutor is pointing at.

When a modeling operation results in an invalid part model at the server, i.e. one or more constraints are no longer satisfied, the Session Manager takes the role of coordinating the validity recovery process. In the example of Figure 7(a), the crown part model becomes invalid as a result of increasing the depth of a pocket, which is in fact turned into a passage feature. Initially, the team member who issued the operation is presented a validity maintenance panel (see Figure 7(b)), where useful information on the particular invalid situation is given, together with validity recovery hints [3]. This team member can specify corrective modeling actions himself and/or hand over the validity



(a) invalid crown part model


Validity Maintenance Panel

Operation edit RectPocket_1 not allowed

Due to the following occurrences:

● transmutation interaction on RectPocket_1

Element affected:

bottom

Possible corrective actions:

● Modify parameters of feature RectPocket_1

○ Remove RectPocket_1

Hand over control to another user:

user B

(b) validity maintenance panel

Figure 7. Collaborative validity maintenance.

maintenance panel to a colleague involved, until they agree on the corrective actions and issue their execution.

At any moment during a collaborative modeling session, a team member may invite a colleague to join a discussion on a part in his modeling scope, either simply as an advisor, e.g. as participant in a shared camera, or as a participant with full modeling rights.

5.2 Collaborative Assembly Modeling

Collaboration is also possible in the assembly design view, among team members who have modeling rights on some compound component. As explained in Section 2, modeling operations in assembly design view consist of creating, modifying or removing connection features, which specify how components should be connected to each other. For example, in the model structure of Figure 5(d), the front fork lies in the modeling scope of team members A and B, and therefore they can establish together the connections between the two legs and the crown of the front fork (see Figure 8(a) and (b)).

An important aspect here is that establishing a connection feature may require the creation of the respective form features on the components involved. As explained in Section 2, because of the integration of all views, these component changes are propagated downwards in the hierarchy to the respective parts, where new features are also created. It may occur that one such form feature causes the part it is located on to become invalid. In these cases, the collaborative validity maintenance scheme mentioned in Subsection 5.1 assists the team members involved in recovering validity again.

Similarly to what was described in the previous subsection, team members can discuss assembly issues, e.g. where and how to create a connection feature, using shared camera and telepointer facilities. In the assembly design view, these facilities are available for both geometry and graph cameras. In addition, if assembly considerations require adjustments in any of the single components, the team members can either switch to that component's part design view and directly adjust it, or invite the team member(s) to whom that part has been assigned to join the discussion and perform these adjustments. Because of the integration of all views, changes performed on a part are now propagated upwards in the hierarchy to the compound components which contain the part. So, for example, in the assembly model of Figure 8(b), if users A and B find that the legs of the front fork are too close to each other, they can decide to extend the crown in its part design view, after which they can check whether the modification satisfies the front fork requirements in assembly design view (see Figure 8(c)).

6. CLIENT-SERVER DATA COMMUNICATION

The clients of webSPIFF make use of standard web browsers. When a team member connects to the webSPIFF server, a Java applet is loaded. This so-called Product Navigator implements a simple user interface, from which a direct connection with the Session Manager is automatically set up. Different team members can connect from various locations, in order to start or join a modeling session. As became clear from Subsection 3.1, the choice for thin clients requires a good communication and synchronization mechanism with the webSPIFF server. This section describes in some detail the model data required at the clients, and how the Session Manager deals with data synchronization.

The Session Manager has been implemented using the Java programming language [21], and it makes extensive use of the Remote Method Invocation (RMI) facilities for the communication with the webSPIFF clients.

6.1 Model Data at the Clients

Several visualization and interactive facilities have been implemented at the clients. First, a variety of images of a component can be displayed in cameras: for a geometry camera, a geometry image of the component; for a graph camera, its hierarchical or relational graph. As pointed out in Section 2, these

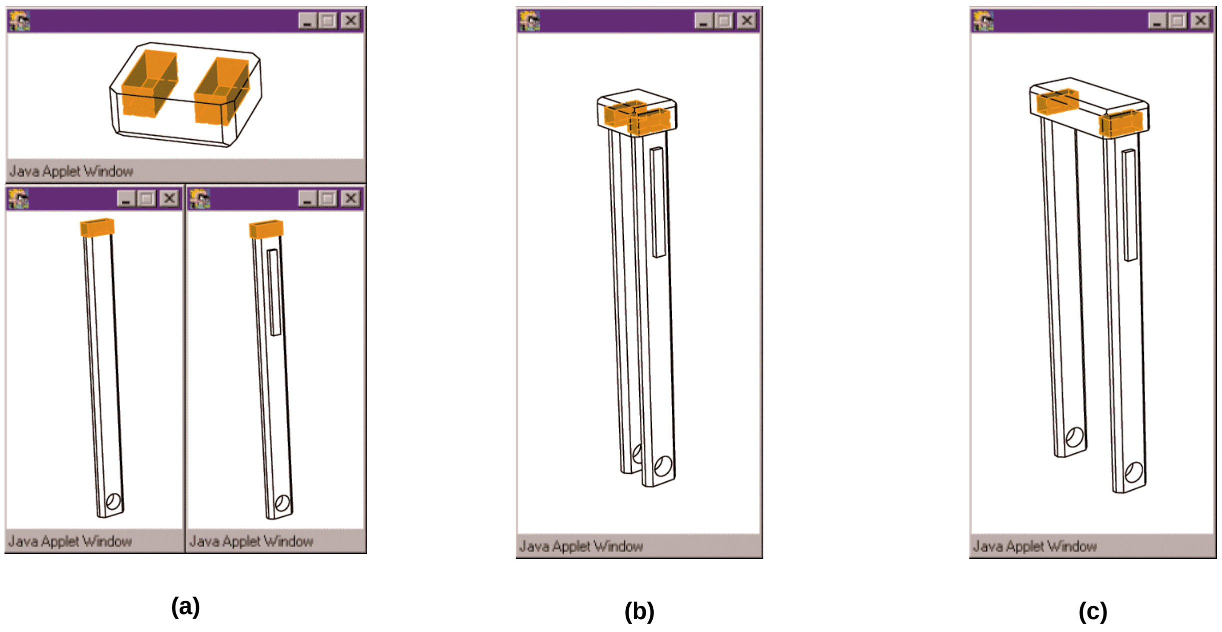


Figure 8. Three components (a) are connected into the front fork assembly of a bicycle (b), of which the crown part is subsequently adjusted to make room for the front wheel (c).

images provide very powerful visualizations of a feature model. Second, interactive modification of viewing parameters, e.g. rotation of a model, can be performed on a geometry camera. After the desired viewing parameters have been interactively set, they are sent to the server, where a geometry image corresponding to the new viewing parameter values is rendered and sent back to the client, where it is displayed. Finally, both types of cameras provide facilities for interactive specification of modeling operations, e.g. assisting the user in selecting features by having them picked on an image of a component.

To support all these facilities, the webSPIFF clients need to locally dispose of model data. However, as explained in Subsection 3.1, a central product model is maintained at the server. Therefore, client model data has to be derived at the server from this central model. Besides text data, necessary for the client's user interface, including an XML description of the product structure for the Product Navigator, model data at the clients can be classified into the following three categories:

- **Model images**

  Both geometry and graph images are rendered by SPIFF, and distributed in GIF format. A separate image is needed for each camera, and it must be updated every time the model or the camera settings are changed.

- **Visualization model**

  The visualization model represents the global shape of a component, and is generated by SPIFF in VRML format [2]. It supports interactive modification of viewing parameters for all geometry cameras. Real-time rendering is locally feasible, whereas rendering of a smooth sequence of geometry images at the server and transmitting it to the clients would be unfeasible in real time. All geometry cameras on a particular client, including any shared cameras, use the same visualization model, but each camera displays it with its own viewing parameters.

- **Selection model**

  The selection model is a collection of three-dimensional objects representing the canonical shapes of all features in a given view of a component. Its purpose is to support interactive selection of feature faces on a geometry image, during the specification of modeling operations. Again, the selection model is identical for all geometry cameras on a client, each applying its own viewing parameters. The selection model is also generated by SPIFF in VRML format.

Model data at the clients is never modified directly by the clients themselves. Instead, it is updated by the webSPIFF server, as will be explained in the following subsection.

6.2 Data Synchronization

Two important types of processes run in the *Session Manager* at the webSPIFF server (see system architecture in Figure 9). First, it maintains a *Client Manager* for each client, managing all communication with it. A Client Manager receives messages from its client, interprets them, and either processes a message itself, if possible, or propagates it to the SPIFF modeling system. Second, the Session Manager maintains an *Event Manager*, including an event queue that schedules the tasks received from all Client Managers, which have to be passed on to the SPIFF modeling system.

Several types of tasks can be distinguished at the Client Managers. First, session operations have to be handled. These involve starting a session, logging into and out of a session, and closing a session. Second, modeling operations can be received that have to be forwarded to the SPIFF modeling system, which,

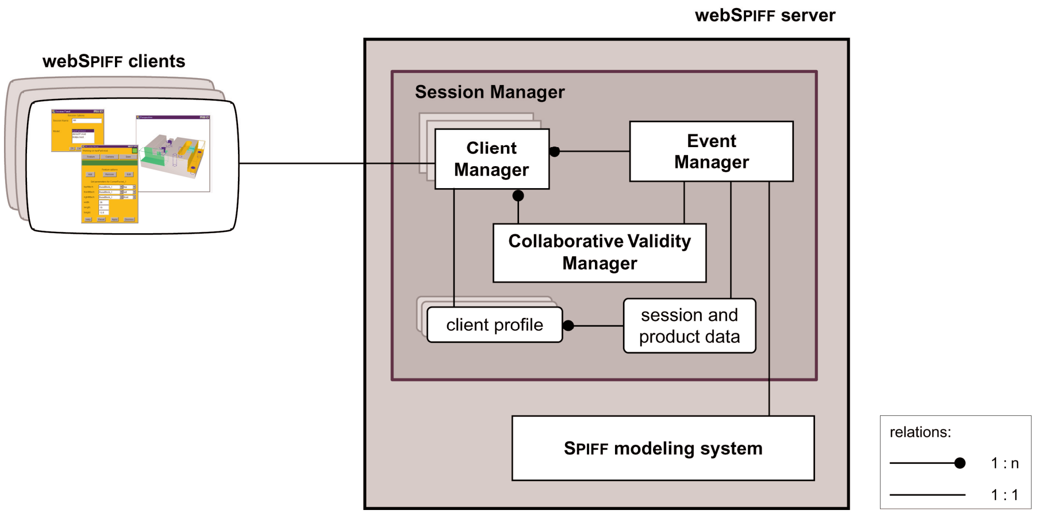


Figure 9. Simplified webSPIFF architecture

after executing them, returns their result to the Session Manager. Sending these results back to the clients is handled by the Event Manager, through the respective Client Managers. Third, camera operations have also to be forwarded to the SPIFF modeling system. Fourth, queries about a feature model are directly answered by the Session Manager, which keeps a record of up-to-date product data in order to reduce response times.

When the webSPIFF server executes a camera operation, a new model image is generated. Since the corresponding feature model remains unaffected by camera operations, the server only needs to send the new model image back to the client that requested the camera update.

However, after a modeling operation has been executed on a component, its feature model has been changed, so relevant updated model data will eventually have to be sent to all session participants using that component. Messages are then sent to each client with the corresponding updates, consisting of, possibly several, new model images, a new visualization model, and an incremental update of the selection model (containing only new and/or modified feature canonical shapes). Also, additional information on the feature model is included in the messages, such as an updated list of its feature instances. Separate messages have to be created per client, since geometry cameras typically have different settings per client, and therefore require different geometry images. Of course, the feature model information will also be different for users of different views. In order to reduce "collaboration noise", due to too many distracting updates, clients may analyze updates, and possibly decide to postpone processing them until some convenient synchronization time.

Finally, implementing the shared cameras in webSPIFF was straightforward, because every client already disposes locally of its own visualization model, introduced in the previous subsection. The only requirement here is the propagation of the changing viewing parameters, together with the coordinates of the telepointers, to all participants involved, so that these can be adjusted for the local shared camera. This is effectively handled via the Session Manager, which receives from every client the modified parameters, and forwards them to the other users.

7. CONCLUSIONS

A new collaborative framework has been presented that supports integrated design of parts and assemblies. Collaborative sessions via the Internet, among several members of a product development team, are possible. A product can be hierarchically structured, with compound and single components, and meanwhile tasks can be assigned to team members. During the actual modeling, each team member can have his own specific view on the product, in particular a part detail design view, a part manufacturing planning view, or an assembly design view. All these views are kept consistent, by using a central product model.

The collaborative framework not only offers possibilities to simultaneously work on independent tasks in a product development process, but also synchronous facilities to really collaborate on the design of a same component. Together with the integration of part and assembly modeling, this means a major step forward in collaborative modeling.

The chosen client-server architecture and the functionality of the clients imply that both the computational requirements for the clients and the network load are low. The server, on the other hand, can become rather heavily burdened for complex products and large development teams. However, we believe that problems arising in this respect may be solved by using a distributed server approach, because independent components of a product can be handled by different servers, running their own modeling kernel, but coordinated by a single Session Manager.

A useful extension would be to incorporate *conceptual design* and *assembly planning* views in this collaborative framework. The former would allow a more flexible specification of components and interfaces prior to the definition of detailed geometry; the latter would bring in assembly sequencing and other assembly considerations. Together, these would represent another important step towards a collaborative and integrated product modeling system covering the entire product life cycle.

8. REFERENCES

[1] Alibre (2002) Alibre Design. Alibre Inc., Richardson, TX. *http://www.alibre.com*

[2] Ames, A., Nadeau, D. and Moreland, J. (1997) The VRML 2.0 Sourcebook. Second Edition, John Wiley & Sons, New York

[3] Bidarra, R. and Bronsvoort, W.F. (2000) Semantic feature modelling. *Computer-Aided Design,* **32**(3): 201–225

[4] Bidarra, R., van den Berg, E. and Bronsvoort, W.F. (2001) Collaborative modeling with features. CD-ROM Proceedings of the 2001 ASME Computers and Information in Engineering Conference, ASME, New York. *See also: http://www.webSPIFF.org*

[5] Bronsvoort, W.F., Bidarra, R. and Noort, A. (2002) Feature model visualization. To be published in: *Computer Graphics Forum*

[6] Bullinger, H.-J. and Warschat, J. (1995) Concurrent Simultaneous Engineering Systems. Springer-Verlag, Berlin

[7] CoCreate (2002) One Space Suite Solution. CoCreate Software, Inc., Fort Collins, CO. *http://www.cocreate.com*

[8] Deneux, D. (1999) Introduction to assembly features: an illustrated synthesis methodology. *Journal of Intelligent Manufacturing,* **10**(1): 29–39

[9] Heisserman, J. and Mattikalli, R. (1998) Representing relationships in hierarchical assemblies. CD-ROM Proceedings of the 1998 ASME Design Engineering Technical Conferences and Computers in Engineering Conference, ASME, New York

[10] van Holland, W. and Bronsvoort. W.F. (2000) Assembly features in modeling and planning. *Robotics and Computer Integrated Manufacturing,* **16**(4): 277–294

[11] Kaon (2001) HyperSpace-3DForum. Kaon Interactive Inc., Cambridge, MA. *http://www.kaon.com*

[12] de Kraker, K.J., Dohmen, M. and Bronsvoort, W.F. (1997) Maintaining multiple views in feature modeling. Proceedings of Solid Modeling '97 – Fourth Symposium on Solid Modeling and Applications, Hoffmann, C.M. and Bronsvoort, W.F. (Eds.), ACM Press, New York, pp. 123–130

[13] Lee, J.Y., Kim, H., Han, S.B. and Park, S.B. (1999) Network-centric feature-based modeling. Proceedings of Pacific Graphics '99, Kim, M.-S. and Seidel, H.-P. (Eds.), IEEE Computer Society, Los Alamitos, CA, pp. 280–289

[14] Nam, T.J. and Wright, D.K. (1998) CollIDE: A shared 3D workspace for CAD. Proceedings of the 1998 Conference on Network Entities, Leeds

[15] Noort, A., Hoek, G.F.M. and Bronsvoort, W.F. (2002) Integrating part and assembly modelling. To be published in: *Computer-Aided Design,* **34**(13)

[16] Parametric (2001) Pro/ENGINEER 2001*i.* Parametric Technologies Corporation, Waltham, MA. *http://www.ptc.com*

[17] SDRC (2001) I-DEAS. SDRC, Milford, OH. *http://www.sdrc.com*

[18] Shyamsundar, N. and Gadh, R. (2001) Internet-based collaborative product design with assembly features and virtual design spaces. *Computer-Aided Design,* **33**(3): 637–651

[19] Stork, A. and Jasnoch, U. (1997) A collaborative engineering environment. Proceedings of TeamCAD '97 Workshop on Collaborative Design, Atlanta, GA, pp. 25–33

[20] Spatial (2002) ACIS 3D Modeling Kernel, Version 7.0. Spatial Technology Inc., Westminster, CO. *http://www.spatial.com*

[21] Sun (2002) The Sun Java™ Technology Homepage. Sun Microsystems Inc., Palo Alto, CA. *http://java.sun.com*

[22] Valin, V., Francu, A., Trefftz, H. and Marsic, I. (2001) Sharing viewpoints in collaborative virtual environments. CD-ROM Proceedings of the IEEE Hawaii International Conference on System Sciences, Maui, Hawaii

[23] WebEx (2002) WebEx Meeting Center. WebEx Communications Inc., San Jose, CA. *http://www.webex.com*

[24] Whitney, D.E. and Mantripragada, R. (1998) The Datum Flow Chain: A systematic approach to assembly design and modeling. CD-ROM Proceedings of the 1998 ASME Design Engineering Technical Conferences and Computers in Engineering Conference, ASME, New York