Web-based interaction on feature models

Eelco van den Berg, Rafael Bidarra and Willem F. Bronsvoort

Computer Graphics and CAD/CAM Group
Faculty of Information Technology and Systems
Delft University of Technology
The Netherlands

E.vdBerg/R.Bidarra/W.F.Bronsvoort@its.tudelft.nl

Keywords: Feature modelling, Collaborative modelling, Web-based modelling, Graphical interaction

Abstract: The opportunities offered by Internet technologies have resulted in numerous new types of applications, among which so-called *collaborative systems*, i.e. distributed multiple-user systems that are both concurrent and synchronised. During a collaborative session, several users are connected to each other in order to perform activities, such as design or evaluation, together. This paper discusses how interactive modelling facilities can be offered to the clients of such an environment. Several techniques for interaction with feature models, ranging from display of sophisticated feature model images to interactive selection facilities, have been implemented in webSPIFF, a new web-based, collaborative modelling system. In particular, maintenance of model data at the clients, and their effective utilisation for enhancing user interaction and collaboration are described. The system has a well-balanced distribution of functionality between the server and the clients, and a good compromise between interactivity on the clients and network load has been achieved.

1. INTRODUCTION

Most current CAD systems are feature modelling systems that run on workstations or PC's. Such systems have become very large, expensive and complex, and require considerable computational power for many operations on models. Interaction with a system is usually only possible if the user is directly working at the CAD station, although remote interaction is sometimes possible through a high-bandwidth local area network.

This situation is no longer desirable, as nowadays more and more engineers, often at distinct locations, are getting involved in the development of products. It would be preferable if a user could remotely browse and manipulate a model, via Internet, as if he were working directly at a powerful CAD station. Even more attractive would be the support of collaborative modelling sessions, in which several users can work together on the design of a product. Typically, in such collaborative sessions, each participant would be provided with his own application-specific view on the product model, as well as specific participation privileges.

Only recently, modelling systems have been proposed that in some respects satisfy the above-mentioned requirements. Among them, the research prototype systems CollIDE (Nam and Wright 1998), CSM (Chan *et al.* 1999) and NetFeature (Lee *et al.* 1999) present some promising solutions to model synchronisation issues, although leaving many concurrency problems unsolved. To the best of our knowledge, the only commercial system currently offering some collaborative facilities is OneSpace (CoCreate 2000). However, its modelling capabilities are strongly constrained by the use of the native SolidDesigner format at the server, into which all models are converted.

The above concepts combine very well with the increasingly popular concept of Application Service Providers (ASP), in which clients remotely access, via Internet, specialised applications running on a server, being billed exclusively for the service time they spend logged on at the ASP server. Such an approach has been pointed out as a very promising and affordable alternative for distributed CAD teams (Comerford 2000). The first commercial CAD ASP has recently been launched by CollabWare, based on the GS-Design modelling system (CollabWare 2000).

Here a new web-based, collaborative feature modelling system is presented. A complete description of its architecture and functionality can be found in (van den Berg 2000), including solutions to the well-known concurrency and synchronisation problems in collaborative applications. This paper concentrates on the facilities for interaction with feature models in such an environment. In particular, it describes the models that are maintained by the clients, and how these models can be effectively used for interaction.

The paper is organised as follows: first, an overview of the web-based, collaborative modelling system is given (Section 2); second, facilities for interactive visualisation of the product model, in particular of its features, are described (Section 3); third, techniques are presented for interactively selecting and using feature entities in the specification of modelling operations (Section 4); finally, a number of technical implementation issues are discussed (Section 5), followed by some conclusions (Section 6).

2. WEB-BASED, COLLABORATIVE FEATURE MODELLING

As a basis for the new collaborative system, the SPIFF system developed at Delft University of Technology was chosen, which offers several advanced modelling facilities. First, it offers multiple views on a product model, each view consisting of a feature model with features specific for the application corresponding to the view. For example, there may be different views for design and manufacturing planning of parts. All views on a product model are kept consistent by feature conversion (Bronsvoort *et al.* 1997). Second, it offers feature validity maintenance functionality. This can guarantee that only valid feature models, i.e. models that satisfy all sorts of requirements, are created by a user (Bidarra and Bronsvoort 2000). Third, it offers advanced feature model visualisation techniques, which visualise much more specific feature information than other systems do. For example, feature faces which are not on the boundary of the resulting object, such as closure faces of cylindrical holes, can be visualised too (Bronsvoort *et al.* 2000). All these facilities are computationally expensive, and require a sophisticated product model, including a cellular model with information on all features in all views.

webSPIFF, the new web-based, collaborative feature modelling system introduced here, has a client-server architecture; see (Lewandowski 1998) for a recent survey on such architectures. All real feature modelling computations, such as feature conversion, feature validity maintenance and feature model visualisation, are executed on the product model on a server that runs the SPIFF system, and their results are exported to the clients. Interaction with the feature model is possible on the clients, but as soon as real feature

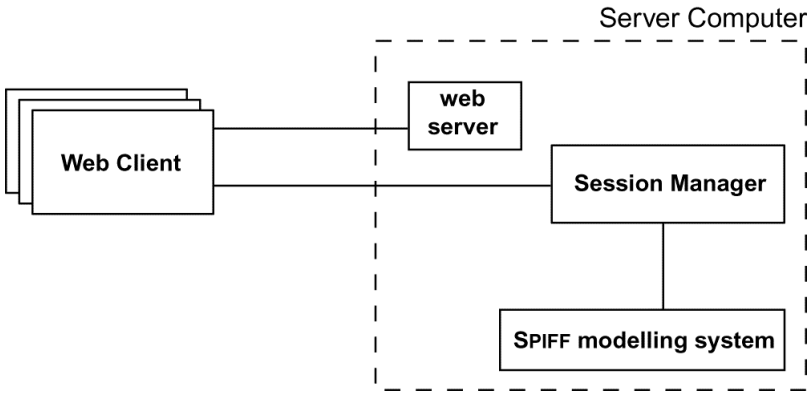


Figure 1. Architecture of webSPIFF

modelling computations, such as the ones mentioned above, are required, the server is activated. Such a client-server architecture is very suitable to solve the main problems that occur in collaborative modelling systems, such as concurrency and synchronisation problems.

webSPIFF consists of several components, as depicted in the global architecture diagram of Figure 1. On the server side, two main components can be identified: the SPIFF modelling system, which provides all modelling functionality; and the Session Manager, which provides functionality to start, join, leave and close a modelling session, and manages all communication between SPIFF and the web clients. The web server component is only used to provide the initial connection with a new client.

The clients of webSPIFF make use of standard web browsers. When a new client connects to the web server, a Java applet (Sun Microsystems 2000) is loaded, implementing a simple user interface, from which a direct connection with the Session Manager is set up. Different web clients can connect from various locations, local through a network or remote via Internet, in order to start or join a modelling session.

The Session Manager stores information about the ongoing sessions and their participants. There is a separate SPIFF process for each session. The Session Manager handles the information streams between web clients and the corresponding SPIFF process.

Since several session participants can send modelling operations and queries to SPIFF at the same time, concurrency must be handled at the Session Manager. Practically, this means that parallel information streams have to be serialised.

The Session Manager can be compared to an operator, sitting behind a computer, collecting messages from several other people, and using the information in those messages to operate the SPIFF modelling system. When several messages are handed over to the operator at the same moment, he must choose which one to handle first, and he must also store the other ones during the processing of the chosen task. The Session Manager has been implemented using the Java programming language.

Using standard web browsers at the clients increases accessibility, but limits the complexity of the operations that can be implemented on them. Nevertheless, the main goal of the work described here was to make available to the clients, in an interactive way, as much as possible of the original SPIFF system functionality.

The bottom line is obviously that clients should be able to specify modelling operations in terms of features and their entities; for example, a feature, to be added to a model, should be attachable to features already in the model. Once connected to the

server, the user can join an ongoing collaborative session, or start a new one, by specifying the product model he wants to work on. Also the desired working view on the model, e.g. design or manufacturing planning, has to be specified. Information on the feature model of that view is retrieved from the server, and used to build the client's user interface, through which the user can start active participation in the modelling session.

After a feature modelling operation, with all its operands, has been fully specified, the user can confirm the operation. The operation is then sent to the server, where it is checked for validity and scheduled for execution. Notice that this can result in an update of the product model on the server, and thus of the view on the model of the other session participants. Updating each client's model data is performed by the Session Manager.

In addition to the above, the following requirements have been set on the clients' functionality:

a) the sophisticated feature model images produced by the advanced feature visualisation facilities of the server should be made available to the user, because these can be very helpful during a modelling session;
b) interactive visualisation operations, such as changing the viewing parameters, should be supported;
c) interactive support of modelling operations, such as selection of a feature or a feature face in an image of the model, should be possible.

Fulfilment of the first two requirements in webSPIFF is dealt with in the following section; that of the last requirement, in Section 4.

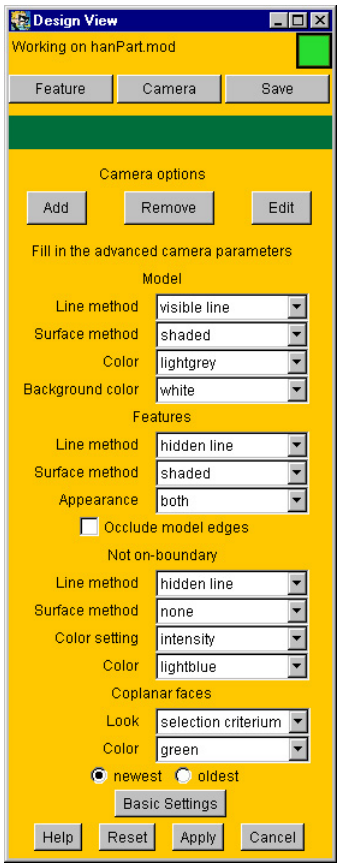
3. VISUALISATION OF THE PRODUCT MODEL

According to the requirements presented above, webSPIFF provides clients with two ways of visualising the product model. First, a sophisticated feature model image can be statically displayed. Second, an image of the model can be shown that supports interactive visualisation operations. Both techniques make use of so-called *camera* windows. A camera window is a separate window in which a graphical representation of the product model is shown. Each client may create as many cameras as desired.

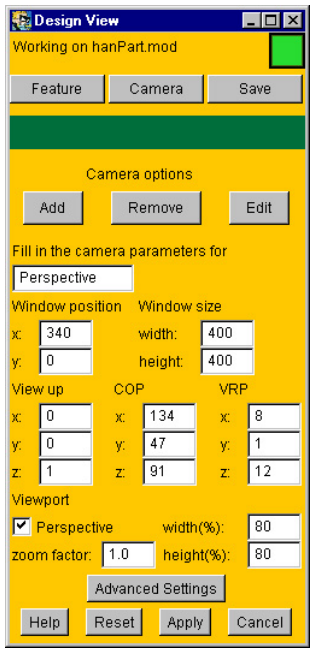
3.1 Sophisticated feature model image

The most powerful visualisation technique generates *sophisticated feature model images*, which can effectively support the user during modelling operations. They provide not only a plain visualisation of the resulting final shape of the product model, but several visualisation techniques are available to the client for customising camera images to a variety of needs (Bronsvoort *et al.* 2000). Sometimes, a user wants to have a closer look at a particular feature in a model, e.g. because he wants to fine-tune its parameters. Using different visualisation techniques for a selected feature and the rest of the model, extra insight into the selected feature is offered, e.g. on its shape and location in the model. For example, the selected feature may be visualised with shaded faces, and the rest of the model as a wire frame or with visible lines only. As already mentioned in Section 2, also additional feature information, such as closure faces of holes, can be visualised. The facilities for rendering such images make extensive use of the ACIS Modelling Kernel, and are therefore not available on the clients. Instead, the images are rendered at the SPIFF modelling system, and sent by the Session Manager to the client, where they are displayed in a camera.

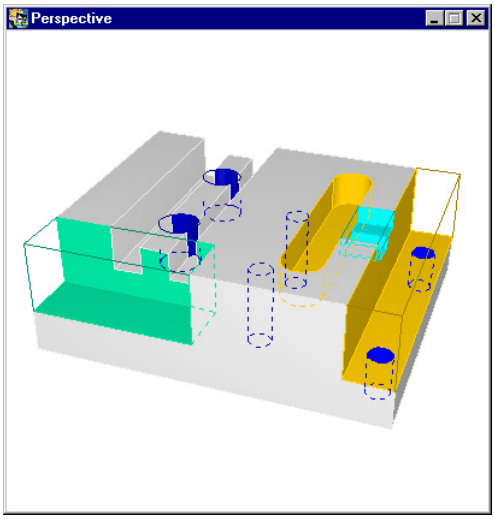
In addition to the settings for the above-mentioned techniques, several viewing parameters (such as centre of projection, view reference point, projection type, etc.) can



(a) model visualisation parameters



(b) viewing parameters



(c) camera window displaying a sophisticated feature model image

Figure 2. webSPIFF camera panels

be set per camera. The camera panels of webSPIFF offer the clients functionality to specify and modify any camera parameter, by means of menus, and control- and checkboxes; see Figure 2 (a) and (b). Interactive specification of the viewing parameters will be elaborated in Subsection 3.2.

The image in a camera has to be updated whenever the client modifies any of its viewing parameters. Similarly, the image no longer reflects the current state of the model when any session participant has modified the model. In both cases, the image is regenerated on the server and resent to the client(s). Both GIF and JPG image formats provide satisfactory results; see Figure 2(c) for a sophisticated image example. Sending

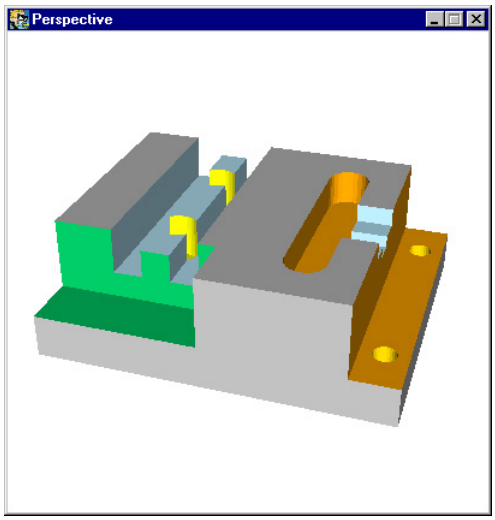


Figure 3. Camera window displaying the visualisation model of the part used in Figure 2

an image from the server to a client is therefore very cheap, both in terms of network load (approximately 10 Kbytes) and display time at the client.

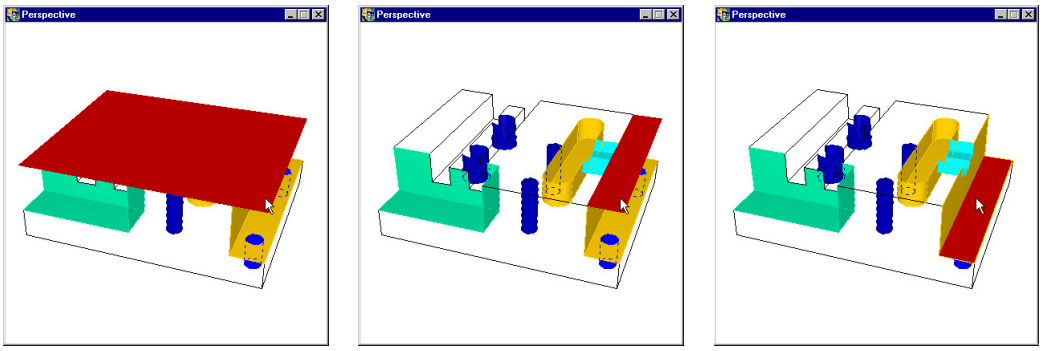
3.2 Interactive visualisation model

As described in the previous subsection, changing the viewing parameters of a camera can be done by specifying values for them using the camera panels. This is convenient, for example, when the user wishes to position the viewing camera at an exact location. Often, however, it is much more practical to be able to freely position and orient the viewing camera in 3D space in an interactive way, using the mouse, as usual in most CAD systems. As the mouse moves, the viewing parameters are then modified continuously according to the mouse events generated, creating a smooth animation. Rendering a sophisticated feature model image at the server, and sending it back to a client, takes some time, which makes it impossible to update the sophisticated image in real time: the time elapsed between arrival of two successive images at the client would simply be too long, hindering smooth interaction.

The requirement for graphical interaction led to the introduction of a *visualisation model*, which is a polygon mesh, generated at the server in VRML format (Ames *et al.* 1997), and sent to the client, where it is loaded into a Java3D scene. Unlike the cellular model on the server, it contains no information about features, except possibly for different colour attributes on faces originating from different features. Figure 3 shows a camera window with an image of a visualisation model.

Maintaining the visualisation model at the clients, viewing cameras can be interactively oriented and positioned in virtual space as follows. As default, a sophisticated feature model image is shown in a camera window, while the visualisation model is hidden. When a mouse button is pressed on the camera, the sophisticated feature model image disappears, and the visualisation model is displayed instead. The user can then interactively adjust the viewing parameters, until the camera has the desired position and orientation. After being confirmed by the user, the new camera parameters are sent to the server, which in turn generates a new sophisticated feature model image according to the new parameters. This is then delivered back to the client, where it is refreshed in the camera window, hiding the visualisation model again.

The visualisation model only needs to be regenerated by the server and updated at the clients whenever any user modifies the product model. Sending the VRML file to the



(a) *(b)* *(c)*

Figure 4. Selection of the step bottom face using the selection model

clients is reasonably cheap in terms of network load (in the order of 100 Kbytes for a moderately complex feature model).

4. INTERACTIVE SELECTION OF FEATURE ENTITIES

As explained in Section 2, an essential characteristic of the SPIFF system is that its modelling operations are specified in terms of features and feature entities (e.g. faces and datums), rather than in terms of faces of the evaluated boundary representation of the product. Among other advantages, this approach avoids the well-known problems of persistent naming of model entities (Bidarra and Bronsvoort 2000).

The interface of webSPIFF clients provides a panel for the specification of feature modelling operations, presenting the required menus filled with appropriate names (e.g. of all features, or of all faces of a particular feature). The user can then browse through these names to specify the operands of feature operations, as he might do when working directly at a CAD station running the SPIFF system.

However, graphical interaction is very useful, not only in the visualisation of the product model, as described in Section 3, but also for assisting the user in selecting model entities, specifically for modelling operations. In fact, it is often much more convenient to graphically select those entities directly on an image of the visualised product model than from a menu.

For this, the *selection model* was introduced at the web clients. It consists of a set of feature canonical shapes, each of which comprises a number of uniquely named entities, in particular the feature faces. Each canonical shape is generated at the server into a separated file in VRML format, and loaded into the Java3D scene at the client.

The canonical shapes in the selection model are never fully displayed simultaneously. Instead, they are kept invisible in the camera, until the user selects a point with the mouse. At that moment, a conceptual ray is determined from the position of the selected point and the viewing parameters used to generate the image. The features entities subsequently intersected by the ray are highlighted in the order of intersection for selection, until the user confirms the selection of the desired entity; see Figure 4 for an example. Notice that in this way also feature faces can be selected that are (partly or totally) not on the boundary of the resulting object, as shown in Figure 4.(b). In Section 5, some implementation details of the selection process are further discussed.

When the camera viewing parameters have been modified, the canonical shapes of the selection model do not need to be updated at the client: the only thing needed is to change them according to the new viewing transformation, similarly to what is done to the

visualisation model. A canonical shape only needs to be regenerated by the server, and reloaded by the client, when the parameters or the position of the corresponding feature are modified, as a result of some modelling operation. Sending VRML files of the canonical shapes to the clients is again cheap in terms of network load (in the order of 5 Kbytes per canonical shape).

5. IMPLEMENTATION AND POSSIBLE EXTENSIONS

In this section, implementation of the techniques presented so far is described in some detail. In addition, some possible extensions are briefly discussed.

5.1 Using Java3D

Basically, two main alternatives were available for visualisation of and interaction on 3D models at the clients: the *Java3D API*, and VRML in combination with the *External Authoring Interface*, *VRML/EAI*.

With VRML/EAI, a Java program creates VRML code that is sent to a conventional VRML viewer. The interactive functionality of the VRML viewer is then used to position, orient and make selections in the model. The model and the Java code are not closely connected to each other, and the interactive functionality is limited to the functionality the VRML viewer offers. Although the VRML standard defines selection on objects, it does not prescribe selection on objects occluded by others. Therefore, it cannot be guaranteed that the VRML viewer available to the user is capable of selecting occluded objects. However, this functionality is very useful, especially for reasonably complicated feature models, where many faces are fully or partly occluded by other faces.

The Java3D API is obviously very closely connected to Java, since it is an extension to the Java programming language. Interactive functionality can be implemented by the programmer, and many standard functions are even already available. Selection is possible, also on faces occluded by other faces.

For these reasons, the cameras were built using the Java3D API. However, VRML is still being used in webSPIFF for transmitting models from the server to the web clients.

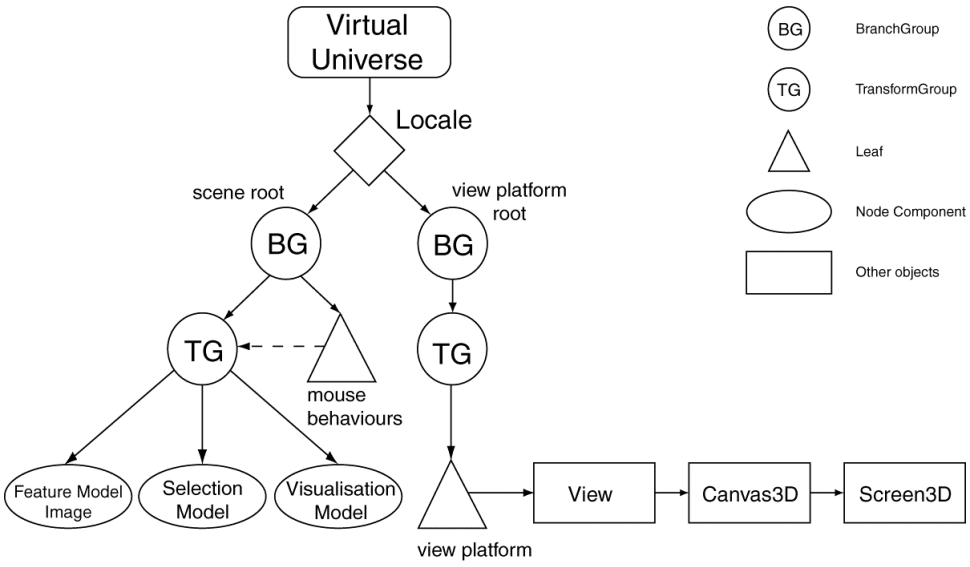


Figure 5. The Java3D scene graph on the web clients

Java3D has built-in VRML loaders at its disposal, so models defined in VRML code can be imported, converted to Java3D objects, and eventually rendered, without the restrictions VRML/EAI suffers from.

Java3D scenes are set up by defining all its components and connecting them to each other in a scene graph. Figure 5 shows the scene graph used on the clients of webSPIFF. The root of every Java3D scene graph is its virtual universe. Every scene has exactly one virtual universe. The Locale class defines the origin of the scene, usually chosen at (0, 0, 0). The left branch of the scene graph defines the geometry of the objects in a scene, whereas the right branch of the scene graph defines the viewing transformations.

Two types of groups can be identified in the scene graph. First, BranchGroups serve as the roots of a scene graph branch. They are the only components that can be attached to a Locale object. TransformGroups are groups containing a transformation that is applied to all its children.

5.2 Mouse events

Besides scene objects, the geometry-defining branch also contains mouse behaviours. These behaviours define which mouse actions will be interpreted as rotations, translations or zoom operations. When one of these actions is performed, the corresponding transformation is computed accordingly.

A synchronisation problem arises here: in the SPIFF modelling system, the user moves the camera through the scene, thus directly changing its viewing parameters, and leaving the model co-ordinates unchanged; in a Java3D scene root, however, the attached mouse behaviours do not affect the viewing transformation, but in fact transform the geometry-defining branch instead. These two approaches have to be synchronised, otherwise the webSPIFF server will not be able to generate a new sophisticated image with the viewing parameters requested by the client. The solution consists of first taking the transformation matrix from the TransformGroup parenting the selection model and visualisation model, and applying its inverse transformation to the position of the viewing camera, the view reference point and the view-up vector. The same inverse transformation is then applied to the transformation matrix in the TransformGroup, effectively resetting it to the identity matrix again. This causes the whole scene, including the viewing parameters and geometry, to be transformed as if the viewing camera has been moved around a stationary model, resulting in the same scene image as before, but with the adequate viewing parameters for the SPIFF server.

Besides defining a scene, including lights and viewing cameras, selection should be handled. In Java3D, specific capabilities have to be set in order to allow certain functionality to be applied to objects in a scene. For selection, this means that every canonical shape object must be explicitly set to allow intersection computations being applied to it.

The selection functions are called when a user clicks the left mouse button on a camera. A ray is then computed and sent into the scene, through the camera co-ordinates of the mouse event, and perpendicular to the viewing plane. If a canonical shape is intersected by the ray, the name of the feature face(s) intersected is included in a list of candidate selected faces, sorted by distance to the viewing camera. In this process, only the branch in the scene graph containing the selection model is searched for intersections.

The resulting list is processed as follows. The VRML file of a canonical shape consists of several nodes, each containing a polygon mesh of a feature face. The names of these node are the names of the feature faces, and they are used by the VRML loader of Java3D to build a hash table. The keys in this hash table are the feature face names, and the objects they refer to are the nodes representing the feature faces. Using this hash table,

the names of intersected objects in the list, returned by Java3D upon selection, can be looked up.

Whenever a feature face has been selected, its polygon mesh is highlighted on top of the sophisticated feature model image. This is done by changing its appearance parameters, setting the transparency attributes from fully transparent to fully opaque, as depicted in the example of Figure 4.

5.3 Extensions

Several possible extensions can be devised to be implemented within webSPIFF. One of them regards the use of *shared cameras* among several participants in a modelling session. In a shared camera, the same viewing parameters are permanently synchronised at all participants, e.g. allowing them to discuss some detail of the product model as if they were together, looking at the same camera. Conceptually very simple on a local network, this facility would have to be carefully optimised in order to effectively enhance user collaboration, avoiding delays in the camera refresh rate at each client. This task may be additionally complicated if *telepointers*, one for each session participant, have to be implemented on a shared camera, allowing a user's mouse cursor to be displayed at the shared camera windows of all other participants.

Additional interactive facilities, although not directly aimed at improving collaboration, could be useful at the clients, e.g. for positioning or dimensioning a feature by dragging handles on its canonical shape.

Although one of the concerns during the development of webSPIFF was to keep the network load as low as possible, one of the most obvious ways of reducing this has so far not been investigated: compression of model files. However, taking into account the relatively small sizes of the different models in webSPIFF, it can be questioned whether compressing and uncompressing a model will reduce the overall time needed to distribute model data across a network. For slow connections between fast computers, compressing data will be always profitable; only for fast connections between slow computers compression will not pay, due to the overhead introduced.

6. CONCLUSIONS

This paper discussed a number of user interaction facilities suitable for web-based, collaborative feature modelling. They have been implemented in the new collaborative modelling system webSPIFF, which uses a client-server architecture. webSPIFF provides a powerful framework for investigating many issues involved in collaborative feature modelling systems, including synchronisation, concurrency and user interaction aspects.

The proposed distribution of functionality between the server and the clients has resulted in a well-balanced web-based system. On the one hand, the full functionality of the original feature modelling system is offered by the server. On the other hand, all desirable interactive modelling functionality is offered by the clients, ranging from display of sophisticated images of feature models to interactive selection facilities. The software on the clients is quite simple, and a good compromise between interactivity on the clients and network load has been achieved.

As Internet technology rapidly improves, faster and better collaboration becomes possible. It can therefore be expected that, although the development of collaborative modelling systems is still at its early stages, such systems will soon play an important role in the whole product development process.

REFERENCES

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

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

van den Berg, E. (2000) Web-based collaborative modelling with SPIFF. MSc Thesis, Delft University of Technology, The Netherlands

Bronsvoort, W.F., Bidarra, R., Dohmen, M., van Holland, W. and de Kraker, K.J. (1997) Multiple-view feature modelling and conversion. In W. Strasser, R. Klein and R. Rau, (eds.): *Geometric Modeling: Theory and Practice - The State of the Art,* Springer, Berlin, pp. 159–174

Bronsvoort, W.F., Bidarra, R. and Noort, A. (2000) Feature model visualization. *Submitted for publication.*

Chan, S., Wong, M. and Ng, V. (1999) Collaborative solid modelling on the WWW. Proceedings of the 1999 ACM Symposium on Applied Computing, San Antonio, pp. 598–602

CoCreate (2000) Shared engineering. *http://www.cocreate.com/onespace/documentation/whitepapers/shared_eng.pdf*

CollabWare (2000) An introduction to GS–Design Beta. *https://www.prodeveloper.net/downloads/whitepaper.pdf*

Comerford, R. (2000) Software, piecewise. *IEEE Spectrum*, **37**(2): 60–61

Lee J.Y., Kim, H., Han, S.B. and Park, S.B. (1999) Network-Centric Feature-Based Modeling. Proceedings of Pacific Graphics '99, IEEE Computer Society, pp. 280–289

Lewandowski, S. (1998) Frameworks for component-based client/server computing. *ACM Computing Surveys,* **30**(1): 3–27

Nam, T.J. and Wright, D.K. (1998) CollIDE: A shared 3D workspace for CAD. Proceedings of the 1998 Conference on Network Entities, Leeds. *http://interaction.brunel.ac.uk/~dtpgtjn/neties98/nam.pdf*

Sun Microsystems (2000) The Sun Java™ Technology Homepage. *http://java.sun.com/*