A Framework for Seamless Computer Supported Cooperative Work in Three-Dimensional Design

Jun-Hai Yong

A Framework for Seamless Computer Supported Cooperative Work in Three-Dimensional Design

Jun-Hai Yong
School of Software, Tsinghua University, Beijing 100084, P.R. China
yongjunhai@tsinghua.org.cn

Abstract

A framework for computer supported cooperative design is proposed in this paper. With the system customization, the cooperation could be seamless. Namely, cooperation conflictions are detected automatically, and the designers are able to customize the system to handle the conflictions fully automatically. In this way, the designers can work on a large three-dimensional mechanical part together freely, and obtain the result quickly. The communication of CAD data is based on a feature array, which greatly reduces the network burden, and speeds up the cooperative design. The procedure of the cooperation can be replayed or modified with the feature array as well.

Keywords: Collaborative Design, Feature-Based Modelling, Computer Supported Cooperative Work; Petri Net.

1. Introduction

Global competition and rapid development of network have greatly changed production style [3]. The companies who can first provide new products to the market have great advantages [6]. Distributed CAD (Computer-Aided Design) techniques provide a means for produce products faster and cheaper [3]. Large firms such as Boeing, Ford and Kodak have used distributed CAD systems to optimize their business processes [6]. A brief survey on some distributed CAD systems is given in the paper [2]. A good review on tools for co-design, mechanisms for distributed modelling, and some other related work can be found in the paper [6]. Some other surveys on computer supported cooperative work can be found in the papers [4, 5, 8, et al].

Currently, a lot of works on distributed CAD systems and cooperative design have been carried out. The following gives some examples. Denis, Gardan and Perrin [2] propose a network architecture based on a replication process and a multi-level language to control the consistency between the client model and the server model. The purpose of them is for the fast CAD data exchanges. Li et al [6] provide a client/server framework for the feature-cased collaborative design, and focus on how to filter the varied information to avoid unnecessary re-retransfer large-size CAD files. Yong, Hu and Sun [11] use Petri net to analyze the behavior of the cooperative design. Based on the Petri net, Yong et al [10] propose a Broker-Employee architecture to deal with singular cases in CAD systems. The Broker and the employee will be inherited in this paper to enhance the stability of the collaborative design. The reason for the inheritance is as follows. The Broker-Employee architecture [10] is stable, for it is able to handle singular cases efficiently; computable, for it can be mathematically analyzed via the Petri net; and reusable, for it is defined by a few standard interfaces.

A new framework for computer supported cooperative design is proposed in this paper. Designers are able to customize their cooperative styles. When they are working on a very large CAD part, they usually require as few interruptions from other designers as possible before the part is almost done. In this way, the design time will be reduced greatly. After the initial model comes up, the designers demand a lot of interactions and communications together with some other people such as those from market department or management. This stage is necessary to improve the design quality. Another scenario for the heavy collaborative interactions happens, when a designer needs to show his ideas to others who may be at some other places. Those ideas should include not only the reasons for the design, but also the process of the design. In any collaborative design scenario, the design process is required to be as fast as possible. The idea for speeding up the design process in this paper is to detect the conflictions automatically, and to use the resources over the network as fully as possible.

The remaining part of the paper is arranged as follows. The new client/server framework is provided in Section 2. Each client or server is associated with some employees. Section 3 introduces the feature array, which greatly speeds up the data exchange. Section 4 presents the distribute mechanisms based on a Broker-Employee architecture. Some application scenarios are given in this section as well. The discussion and some concluding remarks are made in the last section.

2. System framework

As shown in Figure 1, the client/server architecture is adopted in this paper. However, it is quite different from any traditional one. A cooperative CAD system usually suffers from the data exchange and the communication. Therefore, the framework here uses hierarchy architecture to organize all computers used by the collaborative design. The whole system, which may include computers locating at different places far from each other, has the client/server architecture as shown in Figure 1. All computers in the same LAN (Local-Area Network) are grouped together with the client/server architecture, and the communications between any computer in the LAN and the computer outside of the LAN are via the local server. Thus, a local server in the LAN may serve as a client in the system. It organizes, packages, and distributes messages among computers in different LANs. If the number of the computers in a LAN is very large, computers will be classifies according to the network distance and grouped with the client/server architecture recursively. In this way, a server does not have to communicate with all the other computers in the whole system. Some messages could be dealt with directly by the local servers in LANs, and some messages with the same purpose are possibly combined into one message. Hence, the situation of the data exchange and the communication is improved.

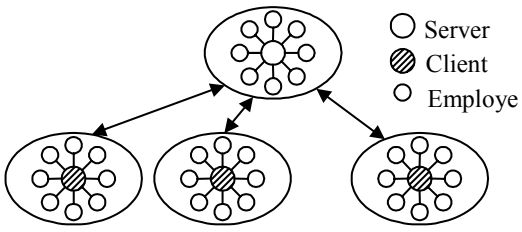

Figure 1. Network architecture.

As shown in Figure 1, every client or server is associated with some employees. This is based on the observation that there are a lot of computers unoccupied over the network for a lot of time. These free computers are very useful network resources. For example, they could help the computing for building mechanical parts. Some computing for the design requires a lot of time such as the computing for carrying out a large array feature, which requires the intersection of a mechanical part with an array of results built by a given type of feature. In this case, the computing could be dispatched into the employees. Another example is about dispatching messages. Here, the message refers to any data transferred from one computer to another one. If the message is very large, the message can be broken into several pieces, and transferred into several employees one by one. And then, each piece of the message is transferred to the destination computer by the employees. If a lot of computers require messages from a server, the server can transfer the message dispatching requirements over the employees. Thus, the communication could be improved in several times.

The above hierarchy structure is built automatically. The modelling softwares used by a server or a client are the same. Therefore, any computer can be used as a server to organize a collaborative design. If a collaborative design project is started in a computer, then this computer is called the server. Other computers, who connect the server and join the project, are called clients. Any computer involved in the collaborative project should register its associate employees at the initial stage. At the initial stage, a computer should download the feature array and some necessary environment data (for example some customized setting) from the server. Then, the system builds the hierarchy client/server architecture automatically according to the network distance. If the communication among server computers is very fast, those computers will be classified into the same group. A user does not have to know whether the computer used by him is served as a pure client or as a local server as well as a global client. If a message is sent from one computer to another one, the message is translated by the system automatically according to the hierarchy structure. Namely, the hierarchy structure is transparent to users. Users could join or leave the collaborative project freely. The system could automatically reorganize the hierarchy structure when users are changed. Thus, to avoid confusion, in the remaining part of the paper a server refers to the pure server, and a client refers to a pure client or the one who is a local server as well as global client. Before a user begins to cooperate with other users, the current modelling part should be downloaded in the local computer. However, it is not necessary to download all CAD data. A feature array is enough to rebuild the part. The feature array will be illustrated in the next section.

3. Distributed feature representation

Figure 2. An example of feature-based modelling.

Feature-based modelling becomes popular recently for three-dimensional design, especially in commercial softwares for industries [6]. Any practical part can be built by a sequence of features, if all of the features can be carried out correctly. Currently, a lot of papers

discuss how to carry out some special features. For example, the papers [1, 7 et al] propose algorithms for up to face extrusion. The paper [9] provides a method for the draft operation. Figure 2 gives an example of a mechanical part built with features. The features used are "protrusion", "cutting", "shelling" and "circle-array" features.

Li et al [6] gives a feature-representation scheme for collaborative design. This paper presents a new representation for feature. It contains more information on network resources and the process history. It is more flexible for collaborative design. The data structure of a feature is as follows.

```
class CFeature : public CObject
{
    CIDFeature                m_id;
    CVersion                    m_version;
    CTime                       m_timeCreate;
    CArrayUser               m_users;
    CAttributeFeature     m_attribute;
    CArrayDependency  m_dependency;
    CArrayEntity            m_entityAuxiliary;
};
```

All the features are stored in an array and with the order of creation time. The information of a feature is addressed as follows.

1) Every feature has a unique ID ("m_id"). When a user requests to create a feature, all the necessary information for creating the feature is sent to the server. After the sever receives the requirement, the sever will check the validation of the feature. If it is valid, the sever will create a feature, and assign it with a unique ID. At this moment, the data structure of the feature is only filled in the necessary information for building the feature and some other basic information such as the information about the user and the version. When a feature is modified, the ID will not be changed, but the version will be increased by 1. Thus, the coincidence of data in two computers can be detected via the IDs and the versions of the features.
2) In the server, there is an even queue in the even center as shown in Figure 3. Therefore, every feature has different creation time ("m_timeCreate").
3) The array of users ("m_users") records the IDs of the users, their roles with respect to the feature such as founder, or mender. Some necessary information for replaying the history of the feature is stored in this array as well. The information in the array is very useful for the "redo/undo" operations.
4) Each feature has some attribution information ("m_attribute"). The basic attribution information is the information for building the feature. Thus, with the information, some faces or some portions of the faces are removed from or added into a three-dimensional mechanical part while the feature is built. An example of this kind of information is the height of a box. A user could change the size of the box, by modifying the height, i.e., the information of the feature. Examples of some other attribution information are color information and material information.
5) Each feature may be associated with some axis, vertices, edges or faces, which are the results of some other features. Therefore, a feature may depend on some other features, which is recorded in the array "m_dependency". Since features can be modified or deleted, some associated information of some depended features may disappear. In order to improve the stability and to keep the validity of the feature even if the depended features disappear, the associated information of the feature is stored in the array "m_entityAuxiliary".

The modelling of a part is the process of building features. The feature manipulations in the paper [6] have four types: *Added, Deleted, Updated* and *Unchanged*. In the framework of this paper, they have six types: *Added, Deleted, Updated, Rebuilt, Suspended* and *Modified*. Here, the feature manipulations, addition and deletion, are similar to those in the paper [6]. When a feature is updated, all the features who depend on this feature will be updated. When the feature manipulation "rebuilt" is required, all the features will be rebuilt. When a feature is modified, some of the attribution information of the feature is modified, and then the feature is updated. When a feature is suspended, the feature will not be updated or rebuilt in the following feature manipulation until the "suspended" is cancelled. The "suspended" feature manipulation is necessary for modelling a large part. Some features, which temporarily do not affect the following design process, could be suspended. Thus, the entities (such as vertices, edges and faces) are temporarily reduced, and the time for building and rendering features is shortened.

4. Distribution and cooperation

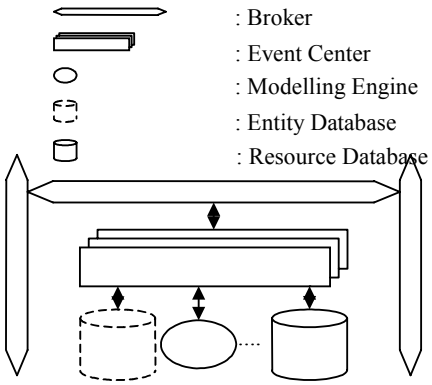

Figure 3. Framework of the three-dimensional distributed design in a computer.

In the framework of this paper, the distributed design environment is based on the Broker-Employee

architecture proposed in [10]. The paper [10] addresses the Broker-Employee architecture with a Petri net. Therefore, it has the same properties with any model represented by a Petri net. Additionally, it provides a model for dealing with singular cases without changing the interfaces of a Broker. Furthermore, the paper [10] provides the hierarchy structure for the Broker-Employee architecture as well. Here focuses how to create the distributed design environment.

When there is any requirement or any operation, the message is sent as an event to the event center via the Broker as shown in Figure 3. The event center decides how to filter events according to the system's customization, the computer(s) in which the even should be dealt with, and how to distribute the event, For example, the event center divides an event into several small events and distribute the small events to some employees, according to the information in the resource database. The resource database records the necessary information of all the available computers, such as how fast a computer is, the average time required for a simple communication between two computers, and the position of a computer in the hierarchy client/server architecture. The information in the resource database provides the fundamental information for the event center to make a reasonable or optimal decision on the event division or the event distribution. And the event center solves the minimum time problem for handling events with the event division operation.

If the event is to build the feature in the local computer, the modelling engine will build the feature and update the local entity database. The entity database stores all geometry, topology, and attribution information for a mechanical part. Since the system does not exchange the CAD data directly, each computer, which is not an employee computer, should build features to obtain the CAD data by itself or its employees.

Conflictions often happen during the cooperation if little control is performed. To manually judge conflictions and deal with them one by one are time-consuming tasks. If the current task is to build a large mechanical part as fast as possible, it is better to have no manual work on handling conflictions before the part is almost done. In three-dimensional design, the conflictions could be aware in the event center or by the modelling engine. For example, when a user wants to build a new feature based on an old feature, which is already deleted by another user, such the confliction can be picked out in the event center. In this example, the confliction can be dealt with simply by ignoring the new feature or creating the new feature with its auxiliary associated structure if it is available. The system could customize the choice. Thus, the system deals with conflictions fully automatically with the users' customization based on the idea that losing some features saves some time with respect to handle the Conflictions manually.

When the part is almost done, users would like to discuss some details on the part. At this time, almost all the conflictions should be handled manually. For example, the system is customized that only one user manipulates the part each time, and that all users have the same view direction on the part. With the customization, a token is used to control who operates the system. Since the Broker and the system could be presented with a Petri net, this could be done simply be setting the number of the token 1. At this scenario, some more types of the messages should be dispatched than those in the previous collaborative scenario which purpose is to reduce design time. For example, the messages, which could change the view directions of a part, should be dispatched.

5. Discussion and conclusions

A cooperative design framework is proposed in this paper. It focuses on collaborative modelling three-dimensional mechanical parts over network. The CAD data exchange is carried out via features. The features can be represented as a feature tree according to the dependency of features. The distribution mechanism is based a Broker-Employee architecture, rather than Agents. The reason is as follows. An Agent could make its own decision according to the environment, and may lead to some different objectives. However, a mechanical part should be accurate, i.e., the modelling result should be definite. Thus, the decision scheme is fixed in the framework of this paper.

The communication of CAD data is based on a feature array. The size in bytes of the feature array is much smaller that the size of the corresponding CAD data. Thus, the network burden is greatly reduced, and the cooperative design becomes more flexible. The feature array records the history of the design process as well. Therefore, the procedure of the cooperation can be replayed or modified.

The proposed framework cooperative design could be seamless with the system customization. With the customization, the cooperation conflictions are detected and handled fully automatically. However, the system will ignore the features, which have conflictions. Those features can be stored in the feature array with the confliction status. Thus, users are able to modify them to eliminate the confliction.

Acknowledgements

The research was supported by the National Science Foundation of China (60403047), and the Chinese 973 Programs (2002CB312106). The author was supported by a Foundation for the Author of National Excellent

Doctoral Dissertation of PR China (200342), and a project sponsored by SRF for ROCS, SEM.

References

[1] X. P. Chen and C. M. Hoffmann. Towards feature attachment. Computer-Aided Design, 27(9):695-702, 1995.

[2] L. Denis, Y. Gardan, and E. Perrin. A framework for a distributed CAD system. Computer-Aided Design, 36(9):761-773, 2004.

[3] J. Y. H. Fuh and A. Y. C. Nee. Distributed CAD for supporting Internet collaborative design. Computer-Aided Design, 36(9):759-760, August 2004.

[4] S. Gao and F. He. Survey of distributed and collaborative design. Journal of Computer Aided Design & Computer Graphics, 16(2):149-157, 2004.

[5] B. Hu, Z.-K. Lin, Y.-C. Guo, and S.-X. Lin. Research on work mode in cooperative design. Journal of Computer Aided Design & Computer Graphics, 10(4):349-354, 1998.

[6] W. D. Li, S. K. Ong, J. Y. H. Fuh, Y. S. Wong, Y. Q. Lu, and A. Y. C. Nee. Feature-based design in a distributed and collaborative environment. Computer-Aided Design, 36(9):775-797, 2004.

[7] Y. Peng, H. Zhang, J.-H. Yong, and J.-G. Sun. Up to face extrusion algorithm for generating B-rep solid. Lecture Notes in Computer Science, 3314:1195-1200, 2004.

[8] R. Tang, P. Xi, and T. Ning. Overview on the state-of-art of feature-based collaborative design software. Journal of Computer Aided Design & Computer Graphics, 15(1):15-20, January 2003.

[9] D.-M. Yan, J.-H. Yong, H. Zhang, Y.-J. Chen, and J.-G. Sun. An algorithm for draft operation in solid modeling. In International Conference on CAD/Graphics, pages 386-387, Macao, China, 2003.

[10] J.-H. Yong, Y.-J. Chen, S.-M. Hu, and J.-G. Sun. A method for dealing with singularity cases in CAD systems based on Petri net. Journal of Computer Aided Design & Computer Graphics, 12(12):941-946, 2000.

[11] J.-H. Yong, S.-M. Hu, and J.-G. Sun. A cooperative CAD system based on Petri net. In International Conference on Computer Aided Design & Computer Graphics, pages 308-312, Shanghai, China, 1999.