Available online at www.sciencedirect.com
ScienceDirect
Robotics and Computer-Integrated Manufacturing 25 (2009) 334–347
Robotics and Computer-Integrated Manufacturing
www.elsevier.com/locate/rcim

Multi-agent collaborative 3D design with geometric model at different levels of detail

Chih-Hsing Chu*, Ping-Han Wu, Yu-Chiung Hsu

Department of Industrial Engineering and Engineering Management, National Tsing-Hua University, Hsinchu 300, Taiwan

Received 16 April 2006; received in revised form 17 September 2006; accepted 11 January 2007

Abstract

This paper presents a scheme for collaborative 3D design using product model at various levels of detail (LODs). Design features are selectively hidden at each level from certain participants, depending on their actual needs and individual accessibility in the collaboration. A tree data structure represents the feature hierarchy of CAD construction, the link between feature and LOD, and 2D mesh data for display control of each feature. An XML/XSLT-based approach is proposed to enable real-time visualization of different LOD models in distributed environment. A collaborative design system is implemented using multi-agent technologies, which focuses on function design of each agent, interactions among agents, the client–server structure, and generation of the LOD data using the XML/XSLT approach. A scenario of synchronous 3D mold assembly demonstrates that geometric categorization of product model provides an operational mechanism for assuring security of information sharing in engineering collaborations over the Internet. It also validates the effectiveness of the agent technologies for automating complex engineering activities.

Keywords: Collaborative product development; Levels of detail; Agent; Design collaboration; Geometric model

1. Introduction

One crucial issue for most companies facing global competition is to assure individual core competences. This leads to the trend of outsourcing on a worldwide basis. Medium- and small-sized enterprises (SMEs) greatly rely on close collaborations with customers, suppliers, and strategic partners in their product development chain for competing with large companies. Such a business operation is often referred to as collaborative product development (CPD), which has been widespread in various industrial sectors. Product data sharing becomes a challenging task in CPD with three major concerns [1]. First, it is necessary to overcome the spatial and temporal separations among the people involved. Moreover, the issue of interoperability across heterogeneous software systems has to be effectively resolved so that distributed resources can be integrated. More importantly, the individual design know-how of the participating companies must be protected while assuring continuous exchange of appropriate product information. General functions in product data management (PDM) fail to fulfill these requirements. New operational mechanisms for data manipulation and accessibility control are required to manage the complexity when the product development tasks are conducted in a distributed manner.

Previous studies [2,3] have shown that multiple-resolution techniques offer an effective approach to achieving a compromise between shape preservation and data size. The underlying principle is mesh refinement, or simplification, adopted for polygon meshes with different LODs. Most of the related research was intended for either computer graphics or CAE applications [4–6]. A little work is focused on engineering design or CAD. On the other hand, the concept about LODs in geometric models has been identified as a requisite for modern product development [7,8]. The geometry-based model is not only utilized at the design stage, but also serves as useful communication means for the other activities in a product life cycle. These tasks may have different requirements and/or limitations in

*Corresponding author.
E-mail address: chchu@ie.nthu.edu.tw (C.-H. Chu).

0736-5845/$ - see front matter © 2008 Elsevier Ltd. All rights reserved.
doi:10.1016/j.rcim.2007.01.005

the use of the geometric model from a design-centric task. Therefore, product information should be properly categorized according to geometry users and their specific needs. Following this idea, framework for access control in computer-aided design environments (FAÇADE) was developed for information assurance (IA) within collaborative design based on role-based viewing [9,10]. The implementation result demonstrates the effectiveness of this system for protecting engineering information in collaborative design with multiple participants while ensuring the availability, integrity, authentication, and access control of shared data. This work adopts traditional mesh refinement in LOD control of geometric model. The similar idea has been extended to data retrieval in distributed product development [11].

Distributed design is a complex engineering activity that requires effective communication among interdisciplinary teams. Differences in system architectures and information representations become a main barrier to this. The proactiveness and autonomy provided by agents may alleviate some of the problems [12]. Multi-agent system (MAS) offers a solution containing non-centralized, mutually cooperating elements suited to the distributed design and manufacturing environment [13]. Ant colony coordination mechanisms were implemented using the multi-agent technologies for handling dynamic changes and disturbances. Distributed intelligent design environment (DIDE) was proposed as a distributed software architecture for integrating multidisciplinary engineering tools in an open environment, realized through a set of asynchronous cognitive agents [14]. An agent-based cooperative intelligent design environment (CLOVER) was implemented to improve the interoperability among distributed and heterogeneous resources [15]. The objective of this research is to establish an agent-based infrastructure using widely accepted standards including ISO standards and standard propositions. Agent technologies also have been utilized in various engineering applications such as the AEC sector [16], mechanical design [17–19], and process planning [20,21].

Multiple-resolution techniques have been identified as an effective approach to achieving good performance in terms of shape preservation and data transfer. Most mesh-based methods were focused on graphics rendering with less emphasis on design intents. On the other hand, several studies concern with compression and de-compression of B-Rep model in distributed environment. Song and Lee [22] proposed a wrap-around operation for producing multi-resolution representations from a B-Rep model, which can be simplified according to the resolution level. A software framework was developed for streaming the B-Rep model through the network. This streaming technique requires rendering environment implementing the warp-around procedure and other specialized CAD functions. Lee [23] proposed a technique for the generation of multi-resolution solid models in terms of features. The focus of this work is to overcome the modeling problem that the sequence of constructing a B-Rep model by Boolean operations cannot be arbitrarily changed. It offered geometric algorithms for the creation of the same model with different construction sequences, or to be more specific, for preserving the same model by adding extra modification operations.

The previous LOD methods derived from mesh refinement treat CAD model as monolithic in their geometry categorization. As a result, they fail to control display/hide of design information in an explicit manner. Separation of part geometry has to be executed based on design semantics so that the result can meet practical needs in collaborative design. To overcome this problem, we develop a representation scheme that provides geometric model at different LODs in terms of design feature. This is achieved by integration of feature-based model and 2D mesh data. The display of a design feature is controlled by its switch face(s). This work also implements a prototyping system based on the scheme using multi-agent technologies. This system allows a group of users geographically dispersed to perform synchronous collaborative 3D assembly over the Internet via autonomous agents. The user leading the collaboration process can hide certain design features from some collaborators and grant them only accessibility to the corresponding models. An XML/XSLT-based approach is proposed to generate the LOD geometric models in distributed environment. A software framework characterizes the function design of each agent, interactions among agents, the client–server structure adopted by the system, and other implementation issues. A scenario of synchronous 3D mold assembly with multiple users demonstrates the effectives of the proposed LOD concept. It also verifies the practicality of multi-agent technologies for automating complex engineering activities.

2. Levels of detail (LOD) in a geometric model

The recent study argues that users' requirements should be used to specify configurable product view [7]. Different types of product illustrations are presented depending on individual roles in the organization. This idea is not only limited to the design stage, but it is also applicable to the entire product life cycle. The content of product information needs to be properly tailored to account for different people, time, and tasks. Categorization of geometric model becomes even more important in CPD, in which not only the designers geographically dispersed are involved, but other stakeholders such as customer, suppliers, and strategic partners also participate in the collaboration process. Different participants have to access product information at various levels in terms of design semantics. Individual know-how can be well protected in this manner. For instance, a product owner certainly does not want to disclose the complete product model when conducting design review with a supplier only responsible for the development of a single part of the product. The supplier should be permitted only to the part being outsourced and

possibly some other parts interfacing with it. In brief, design information of concern should be removed from geometric model and thus becomes obscure from irrelevant people.

Therefore, we claim that a product model needs to be categorized into geometry variants for different users in collaborative design, considering their roles, time involved in the collaboration process, and individual tasks. Fig. 1 depicts a schematic for this idea, with the axis-indicating role (R), time (T), and task (K), respectively. A tuple (R,T,K) corresponds to a geometric model consisting of part or all of the design features in the complete product model. A leading user can choose certain design contents and explicitly specify them to be hidden from some other users. LOD also indicates data assess control at different levels. Another benefit is to reduce data transfer during the collaboration process. Product data to be exchanged can be significantly simplified by eliminating unnecessary or hidden information. This is one urgent issue that remains unsolved in the current collaborative design research [24,25].

2.1. Switch face for controlling design feature

Generation of multileveled geometric models is based on the integration of design features and 2D meshes [26]. Such an approach holds under a few assumptions. First, feature representation is given as prior information. Moreover, the CAD model to be divided is constructed with a series of feature creation operations. A single operation corresponds to only one feature creation. Next, the effective feature volume formed in each operation can be generated by sweeping a 2D profile along a given direction. These assumptions ensure display control of each feature with the switch face(s). Fig. 2 illustrates the process of "covering" a pocket feature created within a block. The first step is to obtain the effective volume resulted from adding the pocket, i.e. a cube comprising six faces. The next step is to remove all the surfaces making up the feature volume. This results in an opening that makes the model visually invalid. A new face, referred to as a switch face, is added to correct this situation. The pocket disappears from the original model when the switch face is turned on. An additive feature like a protrusion shown in Fig. 2(b) is hidden in the similar manner. Certain feature volumes require multiple switch faces to control their display.

2.2. Generation of geometric models with multi-LODs

The LOD models concerned in this work are primarily designed for visual communication in the viewing-based collaboration [25]. Participants utilize viewing tools, namely thin client, with very limited CAD functions in the process. Instead of B-Rep, such tools adopt polygonal models consisting of 2D meshes for shape representation because of their better interoperability across heterogeneous software. The mesh data need to be divided into groups according to the feature creation process. Our previous work developed a geometric algorithm for this

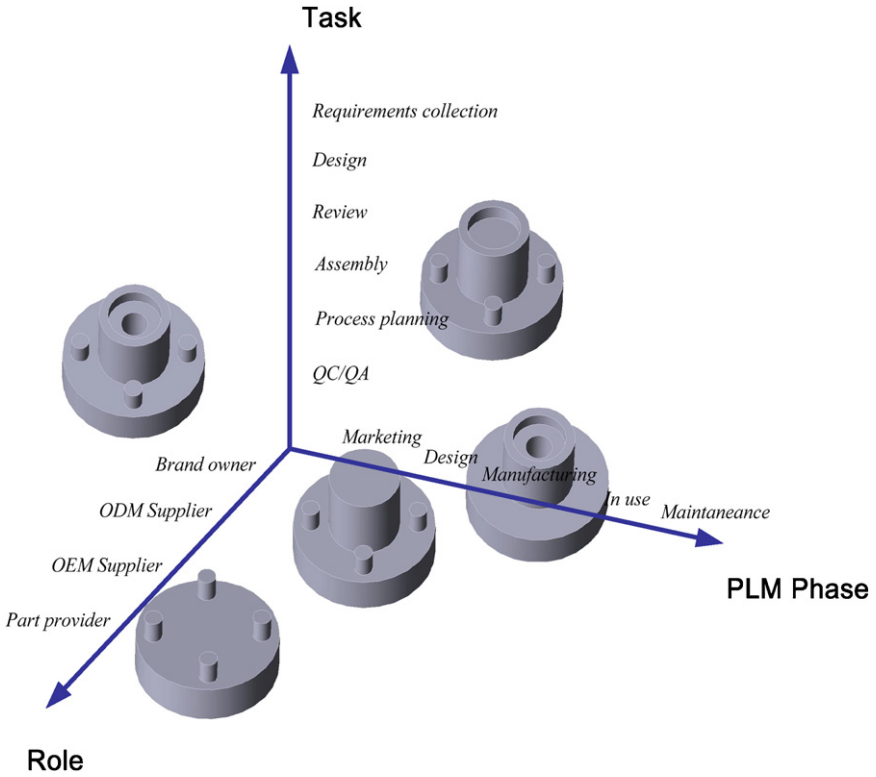

Fig. 1. Schematic for a 3D product model with LODs.

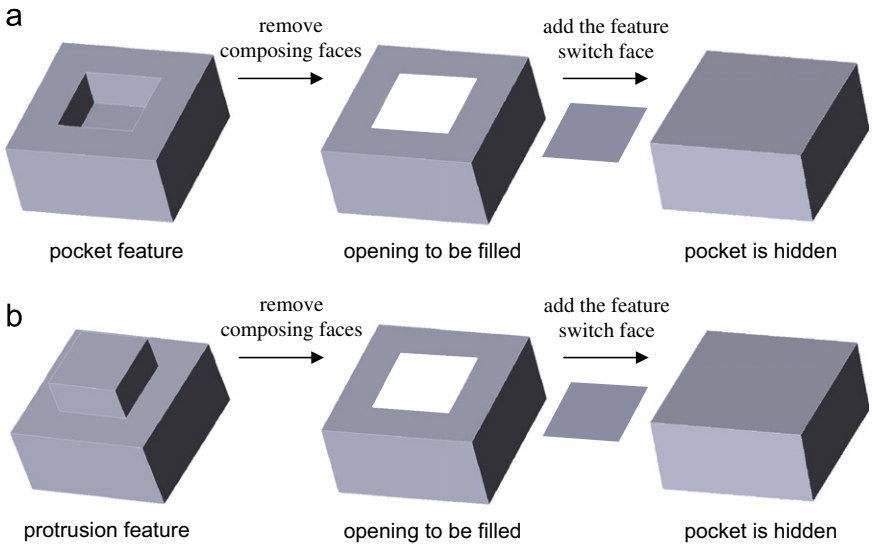

Fig. 2. Procedure of hiding features.

Table 1
Implications of the third digit in FaceID

Value	Definition
1	Face composing an additive feature
2	Face composing a subtractive feature
3	A transient face
4	A switch face of an additive feature
5	A switch face of a subtractive feature

purpose [26], briefly reviewed as follows. L_m represents all the faces of a B-Rep model at the construction step m. V_n represents a set of faces that composes the feature volume corresponding to the nth feature. $F_{n,b}$ stands for the bth face of V_n. $F_{(pq,rs)}$ indicates the Boolean difference between two face sets $F_{p,q}$ and $F_{r,s}$. $\Lambda_{m,n}$ is the collection of the (partial) overlapping face pairs in L_m and V_n. Moreover, $\oplus$ and Θ indicate Boolean intersection and difference operators, respectively. The switch face(s) is expressed as $\oplus_{m,n}$ with the above notations. The other faces required for the display of the model L_m with V_n hidden are expressed as $[(L_m \cup V_n) - \Lambda_{m,n}] \cup \Theta_{m,n}$.

A tree-like data structure is proposed along with a special coding scheme for representing the hierarchical relationships between features, the modeling step, the composing faces, and the switch face(s) of each feature. Each node in the tree structure corresponds to a 2D planar face (or portion of a face). Different node types are characterized with a code, referred to as *FaceID*, which consists of three single digits. The first two digits are only for explanation purpose. They specify the node level in the tree and the index at a given level, respectively. Note that the level zero starts from the top. The third digit characterizes the property of the mesh data contained in the node, as shown in Table 1. A transient face denotes a composing face existing at some previous step, but not in the current model.

Suppose the first modeling step constructs a simple block as shown in Fig. 3, which contains six faces with FaceID 001, 011, 021, 031, 041, and 051, respectively. After a step feature V_1 has been created, the nodes 011, 021, and 031 are decomposed into two child nodes with their IDs as 013, 023, and 033, respectively, as shown in Fig. 4. In addition, the faces changed by the feature creation are updated with the addition of three new nodes (101, 111, and 121) along with the corresponding switch faces (104, 114, and 124). The remaining nodes (132, 142, and 152) represent the faces that compose the effective volume of V_1. Displaying the step feature requires the mesh data represented in the nodes 001, 041, 051 (unchanged faces), 101, 111, 121 (portions of the faces changed by the step creation), and 132, 142, 152 (the faces composing the feature volume). In contrast, to hide the step we have to replace 132, 142, 152 with 104, 114, 124. The creation of a design feature changes the hierarchy of the tree and possibly the FaceIDs as well as the type of some nodes. New nodes may be created and their connection to the other nodes have to be determined. Only the leaf and oval nodes in the data structure correspond to the actual faces making up the 3D model. The root and diamond-shape nodes simply connect the other nodes and maintain the hierarchy after each feature creation.

The data structure described above serves as a kernel for the generation of geometric models at different LODs. The creation of each form feature changes the entire hierarchical structure and possibly the FaceIDs as well as the type of some related nodes. New nodes may be created and their relations to the existing nodes need be updated instantly. The following algorithm describes a systematic approach for updating the LOD tree.

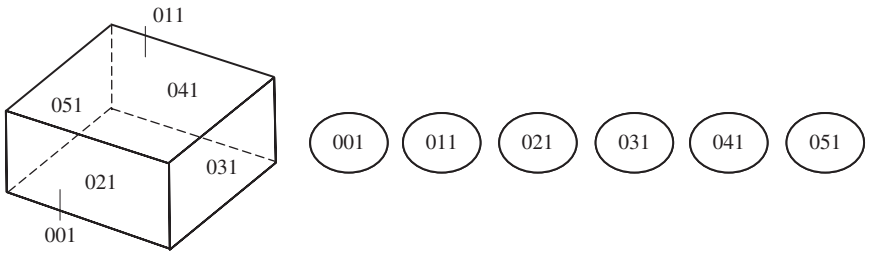

Fig. 3. Nodes corresponding to the block feature.

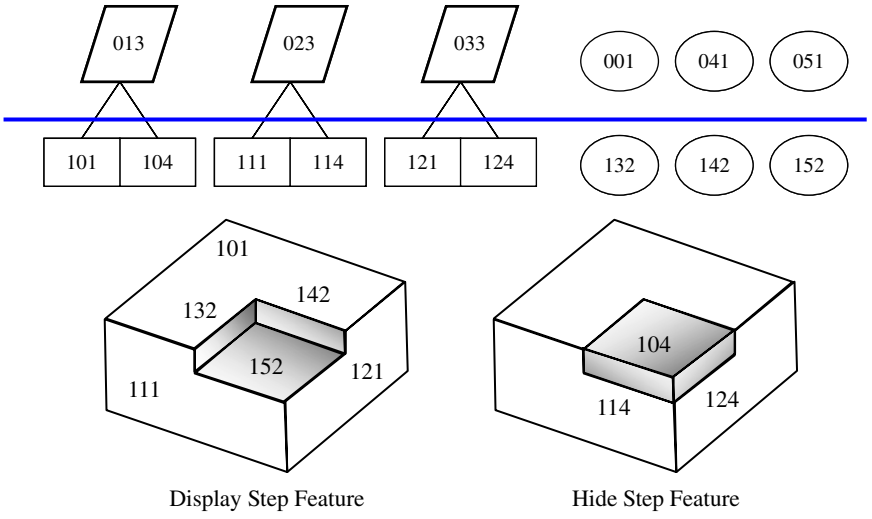

Fig. 4. Nodes updated after the creation of a step feature.

Algorithm L_i represents the model at modeling step i (L_0 is the initial model).
There are n form features (V_1–V_n) to be created in sequence
*get*_FaceID($F_{m,n}$) returns the FaceID of the node $F_{m,n}$
*set*_FaceID($F_{m,n}$) sets the FaceID of the node $F_{m,n}$
Let $i = 0$, $L_{\text{tool}} = L_i$, if_overlap = false;
For each V_j ($j = 1$ to n)
  For each element $F_{\text{tool},a}$ of L_{tool}
    If $i =$ $= 0$
      set_faceID($F_{\text{tool},a}$)
    EndIf
    For each element $F_{j,b}$ of V_j
      set_faceID($F_{j,b}$)
      If $F_{\text{tool},a}$ and $F_{j,b}$ overlap
        if_overlap = true
        Set the third digit of *get*_FaceID($F_{\text{tool},a}$) as 3
      EndIf
    EndFor
  EndFor
  Update_Intersection ($\oplus(\Lambda_{\text{tool},j})$)
  Update_Difference ($\Theta(\Lambda_{\text{tool},j})$)
  $i = i+j$
  $L_{\text{tool}} = L_i$
EndFor

Function set_faceID($F_{m,n}$)
  FaceID.first_digit = m
  FaceID.second_digit = n
  If form feature with positive effective volume
    FaceID.third_digit = 4
  Else form feature with negative effective volume
    FaceID.first_digit = 5
  EndIf
EndFunction

Function Update_Intersection($\oplus(\Lambda_{m,n})$)
  If form feature with positive effective volume
    Set the third digit of *get*_FaceID(each element of $\oplus(\Lambda_{m,n})$) as 4
  Else belong to a form feature with negative volume
    Set the third digit of *get*_FaceID(each element of $\oplus(\Lambda_{m,n})$) as 5
  EndIf
EndFunction

Function Update_Difference ($\Theta(\Lambda_{m,n})$)
  For each element of $\Theta(\Lambda_{m,n})$
    FaceID.first_digit = get_FaceID(feature switch node).first_digit
    FaceID.second_digit = get_FaceID(feature switch node).second_digit
    FaceID.third_digit = get_FaceID(parent node).third_digit
  EndFor
EndFunction

The multi-LOD technique divides a complete mesh-based model into smaller data fractions with explicit design implication. Correct combination of these "design fractions" produces simplified variants of a 3D CAD model. Thus design information can be protected, both semantically and visually, by means of integration of feature-based model and 2D mesh. This approach also contributes to 3D CAD streaming, which is crucial in real-time data transmission and visualization. The user first receives data fractions corresponding to a simplified 3D model without detailed features. Some or all of the features (their mesh data to be specific) can be transmitted later in the correct sequence, depending on the actual needs or network bandwidth. Such a progressive transmission with incremental refinement in design means that it has a good potential in many engineering applications such as collaborative design, virtual reality, and e-commerce.

2.3. XML-XSLT approach

The data representation described in the previous section characterizes the LOD information in a product model during its construction process. This section presents an XML/XSLT-based approach for the generation of all the available LOD models. The reason for adopting this approach is to follow the philosophy of separating data from presentation. There exists only one set of product data represented in an XML document object model (DOM) object. The display/hide of the design features at a LOD is explicitly specified in an XSLT style sheet. The geometric model at the LOD is obtained from parsing the XML object with the corresponding XSLT. In other words, the LOD information is extracted from the complete model by filtering out hidden features expressed in the XSLT. This design pattern, shown in Fig. 5, provides an elegant way for rendering different LOD models. In addition, the information about "who should see what" can be separately captured as business intelligence in text-based XSLT. Changing the geometric model at a LOD is achieved by simply replacing the old XSLT with a new one. Such a change frequently occurs in CPD, in which the accessibility of a participant to product information may evolve as the process proceeds. In addition, implementation of distributed software such as web-based collaboration platform becomes simplified due to the flexibility offered by this design.

Fig. 6 shows the structure of the XML object proposed by this work. It characterizes the design features, the hierarchy among the features in terms of their construction sequence, the mesh data, and the parameters denoting each feature. The LOD models are stored under the branch *LODs_Geometry:ID* is equivalent to the FaceID specified in Table 1; the numerations of *Dependency Type* is listed in Table 2. Although this paper is focused on visualization with multiple LODs, the similar idea is applicable to engineering attributes or other textual information by adding necessary parameters into the structure.

3. Multi-agent collaborative design system

A software agent is a piece of program that must possess any two of the three behaviors: autonomy, cooperation, and learning [27]. Autonomy refers to the fact that an agent can operate on its own without human interventions. Cooperation indicates a social ability of an agent to interact with other agents or human via some communication

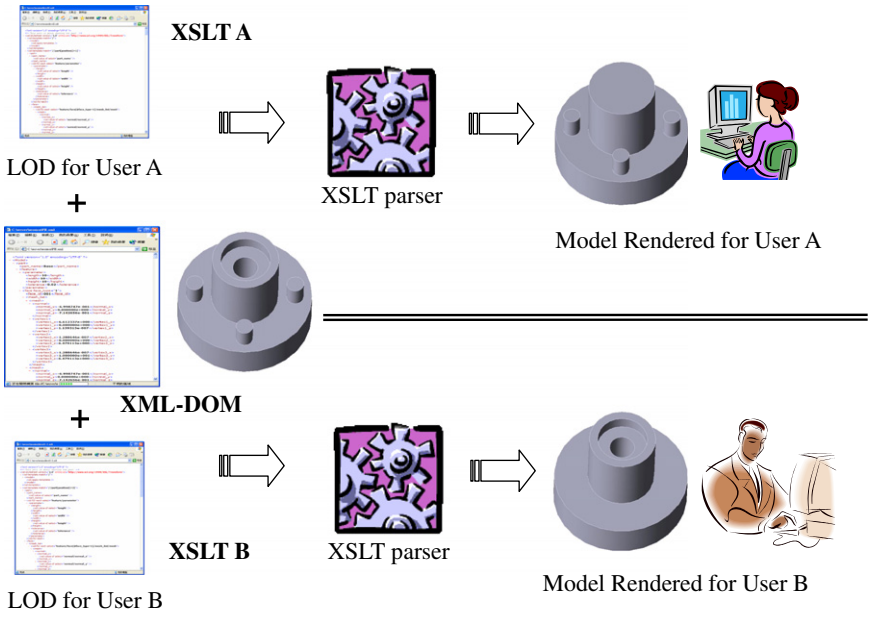

Fig. 5. Generation of a product model at different LODs with XML/XSLT.

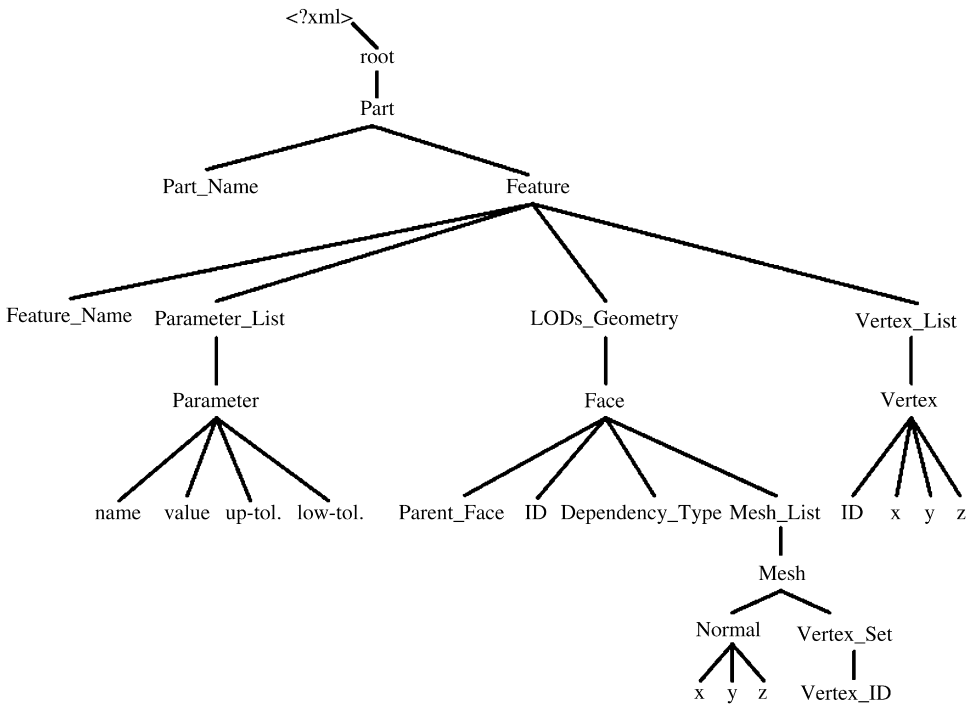

Fig. 6. XML schema of the product model with all LODs.

Table 2
Numerations of dependency type in the XML schema

Name	Meaning
Root node	No parent, one or more children
Leave node	One parent, no child
Independent node	No parent, no child
Transient node	One parent, one or more children

language. An intelligent agent has to learn as it reacts to the external environment. Agents are considered particularly appropriate or distributed collaboration [15]. A MAS facilitates interaction and negotiation by agents that represent various participants in CPD. These agents contain different business intelligences and domain knowledge that enable execution of individual development tasks. This is the main reason for the implementation of the LOD concept using the agent technologies.

3.1. Development environment

This research implements a networked synchronous 3D assembly system based on the concept of collaborative design with geometric models at different LODs. The software development environment is JADE 3.2 (Java Agent Development Environment) [28] due to the inter-operability of JAVA and its open resources. JADE follows the Agent Management References Model (AMRF) specification established by Foundation for Intelligent Physical Agents (FIPA) [29] shown in Fig. 7. AMRF provides a standard infrastructure with well-defined func-

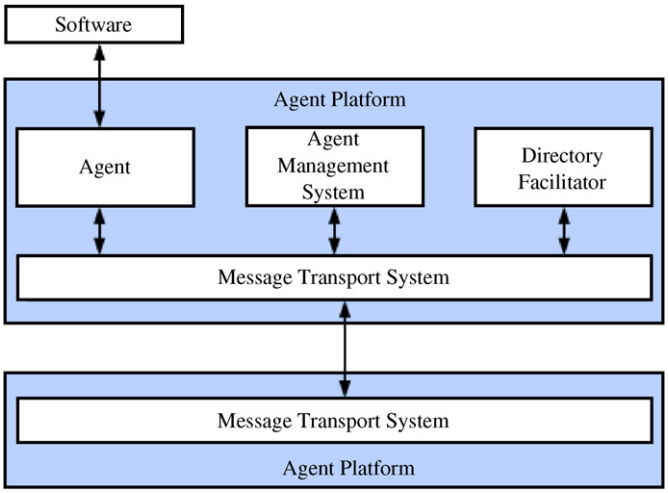

Fig. 7. Schematic of the Agent Management References Model [29].

tions with which multiple agents can be readily created, registered, located, communicated, and de-registered. Note that this standard does not specify how an agent-based system is implemented in terms of software structure. This work follows a client–server design pattern for the system implementation. Another common approach is based on peer-to-peer (P2P) techniques. The prototyping system contains major software components as follows:

- *Agent platform* (AP): it is a software container at which a group of different agents is deployed.
- *Agent*: it is the basic component within the agent platform. Each agent must be registered with descriptions designated by AMRF to provide specific services.

- *Agent management system* (AMS): it is a registry directory that assigns each registered agent a unique and valid identifier (AID, Agent Identifier) so that it can be correctly located and invoked.
- *Directory facilitator* (DF): it is a registry directory that provides yellow-book services. Every agent registers its services against DF so that other agents can query these services through DF.
- *Message transport system* (MTS): it coordinates communication among agents within the same AP or across platforms.

3.2. MAS architecture

This MAS adopts a client–server software architecture shown in Fig. 8. Such an architecture design simplifies information management during the collaboration involving multiple users. Every authorized user has to finish a logging process offered by the server before collaboration starts. Collaboration is carried out within a group of pre-defined users. The server and each client, respectively, correspond to an agent platform (container) that contains a group of different agents. Information exchange between agents follows a common message standard—agent communication language (ACL) [29] in the current specification of FIPA99. The agents on the server side include:

- *Main Server Agent*: the server consists of three software modules: client management, PDM, and LODs management. *Main Server Agent* is responsible for creating three child agents that perform these tasks in the beginning of collaboration.
- *Client Management Agent*: it collects username, password, authority, collaboration role, and accessibility of LODs for each collaborator; then it conducts security check against the backend database.
- *PDM Agent*: it receives product models from the collaborators who own them; then it checks the model validity at its receipt. Finally, it distributes the received models among the users involved.
- *LODs Management Agent*: this agent is responsible for the generation of the geometric model with respect to the LOD requested by and assessable to some user. This is accomplished by rendering the XML data with the corresponding XSLT, as explained later.

On the other hand, a client contains the following agents:

- *Main Client Agent*: this agent provides GUIs for the end user to interact with the system. Besides, it is responsible for creating child agents that, respectively, perform the tasks such as negotiation of design parameters, data upload, and data download.
- *Design Agent*: this agent is in charge of design negotiation, whose tasks include selection of negotiation strategy, execution of negotiation rules, and communication with other agents involved during negotiation.
- *Information Provider Agent*: it uploads the product model to the server when the user it represents decides to share the information with others.
- *Data management Agent*: this agent communicates with *LODs Management Agent* on the server and establishes a connection for receiving product model in XML over the Internet.
- *User Status Agent*: it indicates the status of a human user during the collaboration process and creates a *Domain Agent* based on the design knowledge owned by the user.
- *Domain Agent*: this agent offers design knowledge and negotiation intelligence of a client during the collaboration process. It is also responsible for ensuring fulfillment of the design constraints imposed by the client.

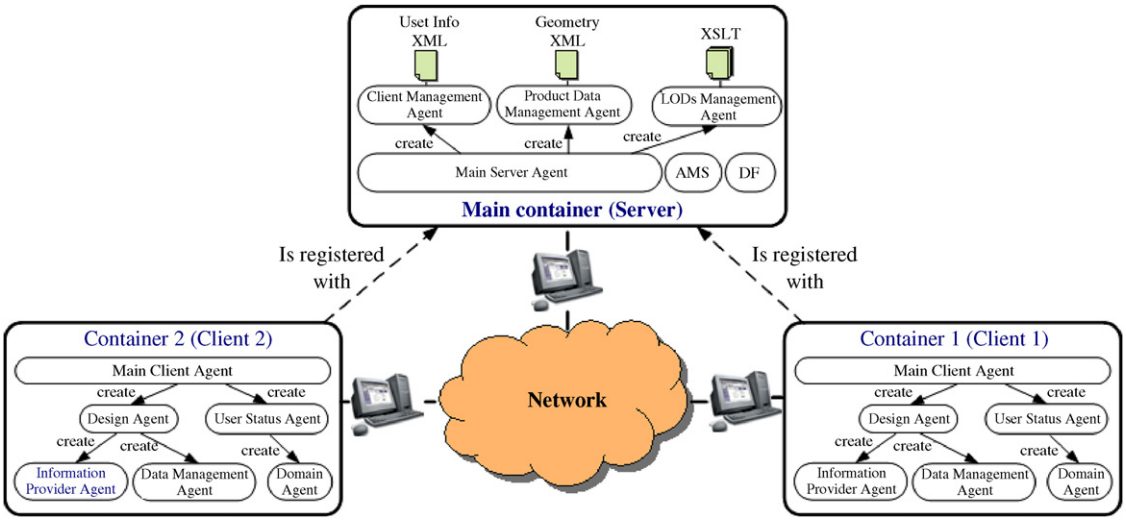

Fig. 8. Software architecture of multi-agent collaborative design system.

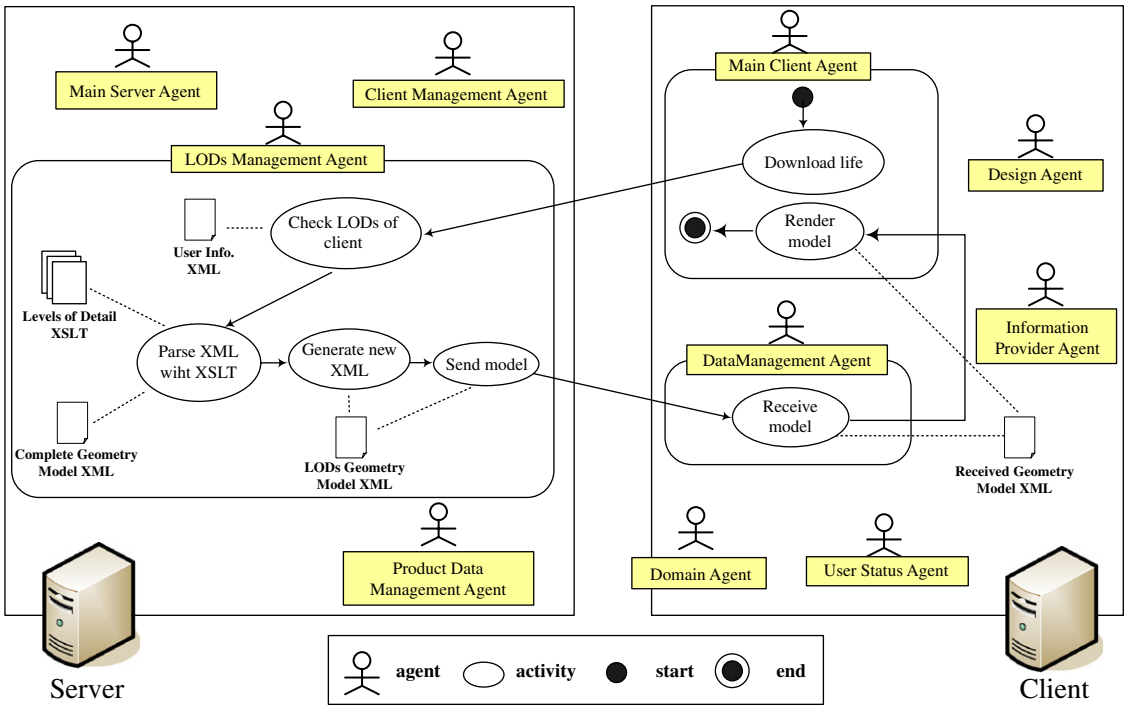

Fig. 9. A typical process of data exchange in the system.

Fig. 9 describes a typical process of data exchange in the prototyping system, which consists of five steps:

1. The actual user asks for data download on its *Main Client Agent*, which then makes the request to *LODs Management Agent* on the server.
2. *LODs Management Agent* retrieves the LOD authorized to the user and the corresponding XSLT sheet.
3. *LODs Management Agent* parses the XML object residing on the server that represents the complete product model with the retrieved style sheet. The model corresponding to the LOD is thus generated.
4. *Data Management Agent* on the client side is informed of the transfer of the requested data. It then receives the model in XML.
5. *Main Client Agent* renders the LOD model with the GUIs.

3.3. Generation of LOD models

The LOD models required in design collaboration need to be generated prior to the collaboration process. Our prototyping system adopts CATIA V5 as the major CAD tool. CATIA V5 is a feature-based design system in which a 3D model automatically carries the feature information from the construction procedure of the model. This, as a result, simplifies the system implementation by avoiding feature recognition. A special module is built into CATIA using component application architecture (CAA) [30] V5 C++, which creates all the LOD models represented in the XML schema shown in Fig. 6. For other non-feature based CAD tools, feature recognition can be applied to produce the LOD data from a B-Rep model. Note that the assumptions made in Section 2 must be satisfied in this case.

4. Collaboration scenario

4.1. Collaborators and individual tasks

A scenario of 3D mold design realized using the prototyping system is used to demonstrate the LOD concept. The mold assembly consists of three major sub-assemblies: a piston, a mold body, and a mold base. The piston consists of a connecting rod and a cap is assembled by bolts. The connector and the piston are joined together with a pin. The mold base holds the other components and attaches the system to a molding machine. There are five collaborators participating in this collaborative design process. As the system owner, $C1$ leads the collaboration process and determines individual authority. Suppose $C1$ decides to outsource the piston and connector as a subsystem to supplier $D1$. Due to its limited production capacity, $D1$ chooses to subcontract the connector to its next-tier supplier $D2$ and focuses on the development of the piston. Two other part providers are also involved in this project. $S1$ is a fixture manufacturer who is responsible for designing the mold base. $S2$ provides standard parts such as pins and screws. $S2$ is invited into the collaboration because the system owner specifies the company as a consign supplier for $C1$. Table 3 summarizes all these

Table 3
Collaborators and their individual tasks during the collaboration process

Collaborator	Tasks
Product owner $C1$	Lead the entire product development Design and manufacture of the mold body
First-tier supplier $D1$	ODM supplier of $C1$ Design and manufacture of the piston
Second-tier supplier $D2$	ODM Supplier for $D1$ Design and manufacture of the connector
Fixture supplier $S1$	Design the mold base
Standard part provider $S2$	Supply all standard parts

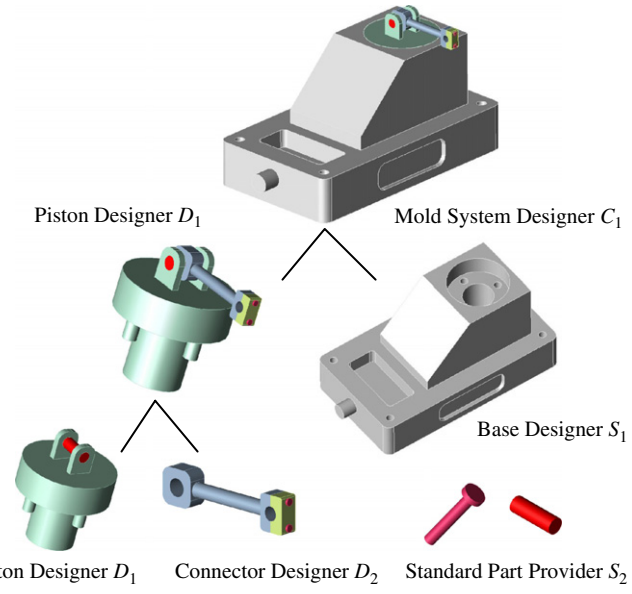

Fig. 10. The mold assembly structure and the owner of each component.

participants and their individual roles during collaboration. Fig. 10 shows the assembly structure and the owner for each sub-assembly.

4.2. Collaborative 3D assembly

The collaboration process is focused on the design review of the mold system. The leading company $C1$ has acquired the CAD model of all the parts prior to the collaboration process and uploaded them to the server. In addition, it needs to specify available LODs and their accessibility to other participants. The server then generates an XML file that can provide all possible LOD models. The next step is to send corresponding XSLT sheets over to each *Client Management Agent*. After the logging authorization, each client agent contacts *LODs Management Agent* and downloads the model assigned by $C1$. Synchronous collaborative assembly is ready to perform once every one has obtained the data corresponding to correct LOD.

The product owner ($C1$ in this case) is the only one who can lead assembly process. The participator of $C1$ selects the parts to be assembled and determines the assembly methods among them via GUIs provided by the client program. *Main Client Agent* thus receives the assembly commands and then dispatches the corresponding tasks to *Design Agent*. This agent program will invoke a non-agent utility on the client side to perform the actual assembly operation by a series of 3D transformations. In addition, it will relay the assembly commands to the *Design Agents* of other collaborators so that their product models can be updated and synchronized after the assembly. Fig. 11 illustrates the interactions among the agents involved in the process.

This system currently allows two different functions for the users: view and assembly. A collaborator normally is allowed to visualize the part(s) it owns and some other parts interfacing with them, but the latter may not contain all the details. For instance, $D2$ owns only the connector, so it should not access to the mold body or base. Similarly, the base manufacturer $S1$ is limited to the product data related to fixturing of the base. It is not necessary for the company to know any detail about the part being molded. That is, the cavity of the mold body ought to be hidden from $S1$. In addition, $C1$ may grant $S2$ the permission to view some standard parts that concern the piston design, but may be not including the mold base. Fig. 12 illustrates the overall assembly process during collaboration. There are two major activities in this scenario: the assembly of the piston/base and the assembly of the piston/connector. The vertical and horizontal axes indicate the different phases during the collaboration and individual roles, respectively. Figs. 13–15 show the screen shots for the product owner, first-tier supplier, second-tier supplier, and fixture supplier at different stages during the process.

5. Conclusion and future work

This research has developed a MAS that realizes the concept of LODs for collaborative 3D design. Product model is categorized into multiple levels by hiding different design features. Participants are allowed to visualize the geometric models at different LODs, depending on the individual roles and their actual needs in the collaboration process. A software framework is proposed for implementing an Internet-based synchronous 3D assembly system using the multi-agent technologies. The focus is on the issues of function design of each agent, interactions among different agents, the client–server IT structure, and rendering of LODs via an XML/XSLT approach. This approach enables the separation of data from business intelligence in real-time applications. In addition, a scenario of online 3D mold assembly with multiple users is used to demonstrate the effectiveness of the LOD concept in real industrial setting. This concept provides an operational mechanism

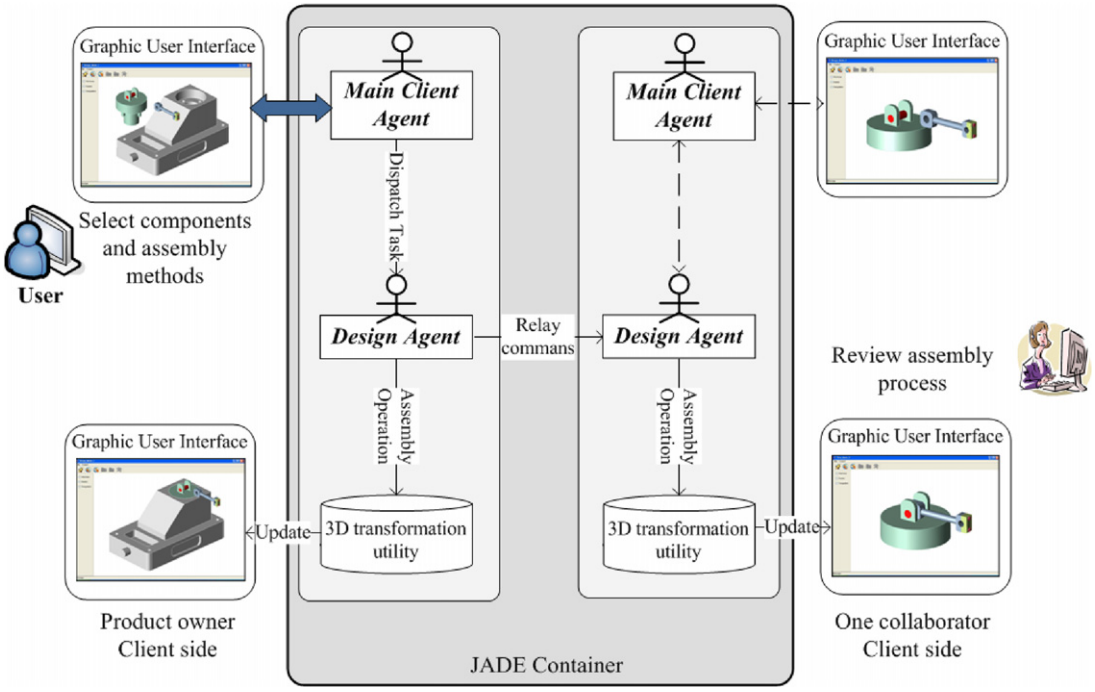

Fig. 11. Collaborative assembly between the product owner and another collaborator.

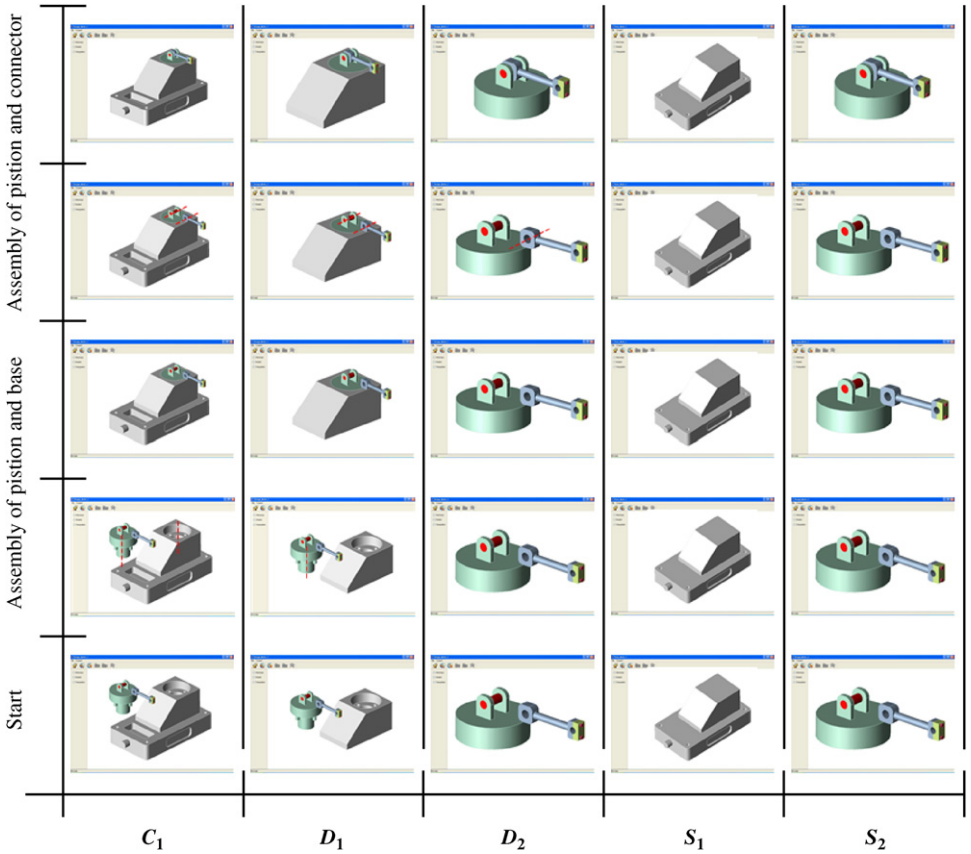

Fig. 12. Collaborative mold assembly based on the LOD scheme.

for enhancing security of information sharing in distributed product development. It improves the efficiency of design collaboration by reducing the product data transferred over the network. The result also verifies that the agent-based approach facilitates automation of complex engineering activities.

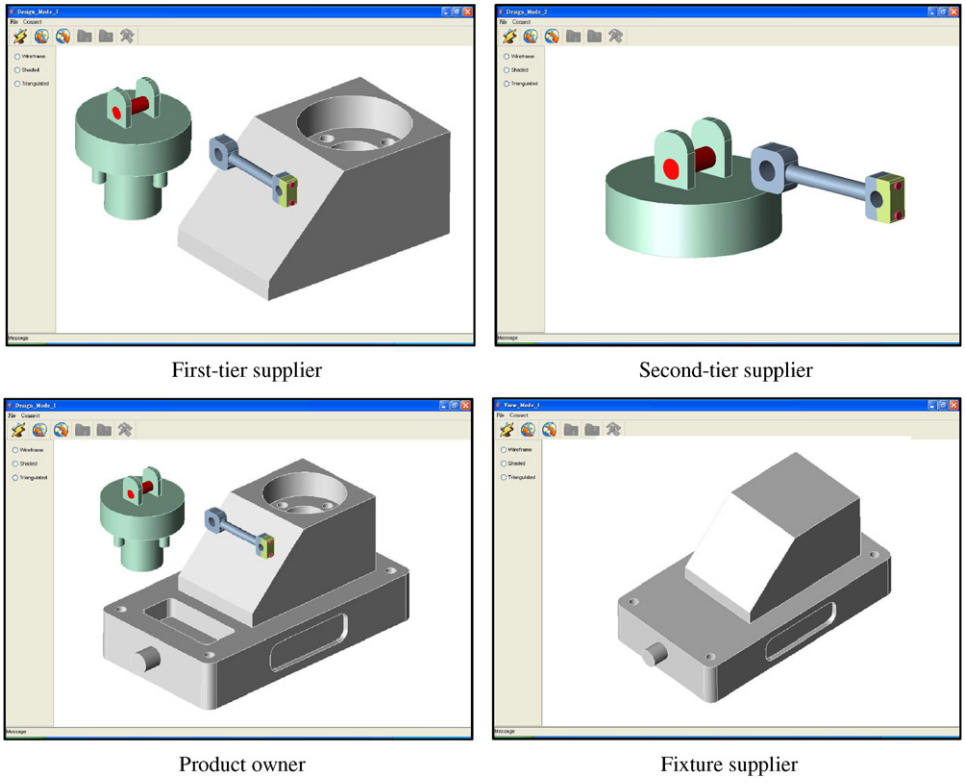

Fig. 13. Screen shots for different participants before the collaboration.

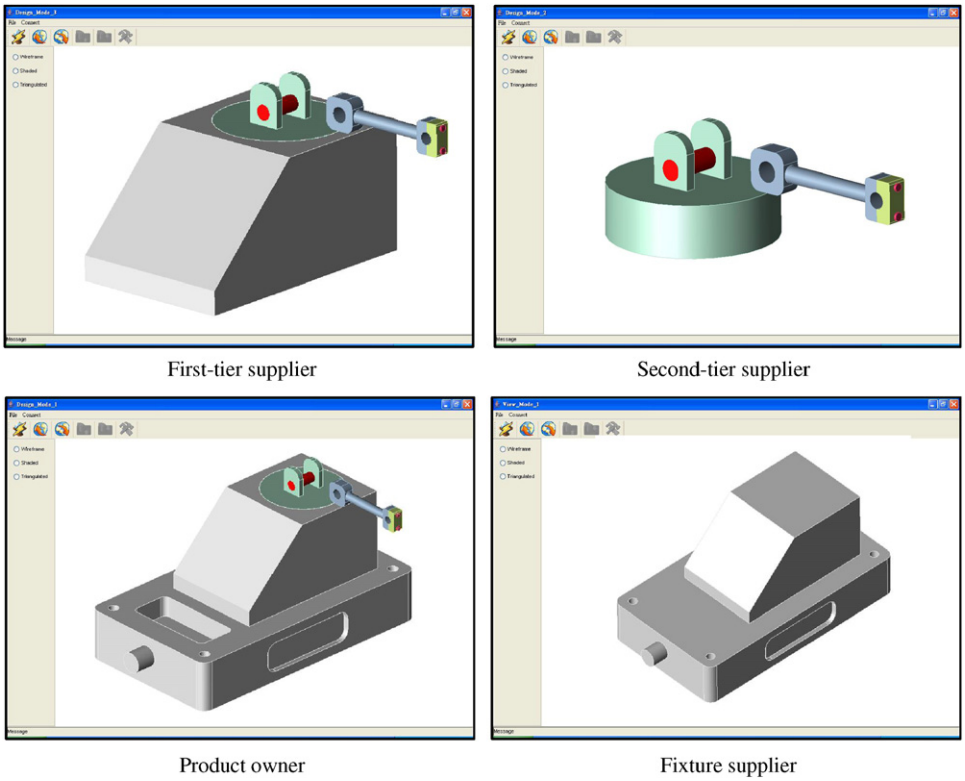

Fig. 14. Screen shots for different participants after the first assembly.

A number of topics deserve further investigation based on this work. One critical issue is to extend the modeling capability of the current scheme so that it can handle more complex feature shapes. Next, multileveled data inevitably induces information imbalance among users and the resulting conflicts in collaboration. A conflict resolving

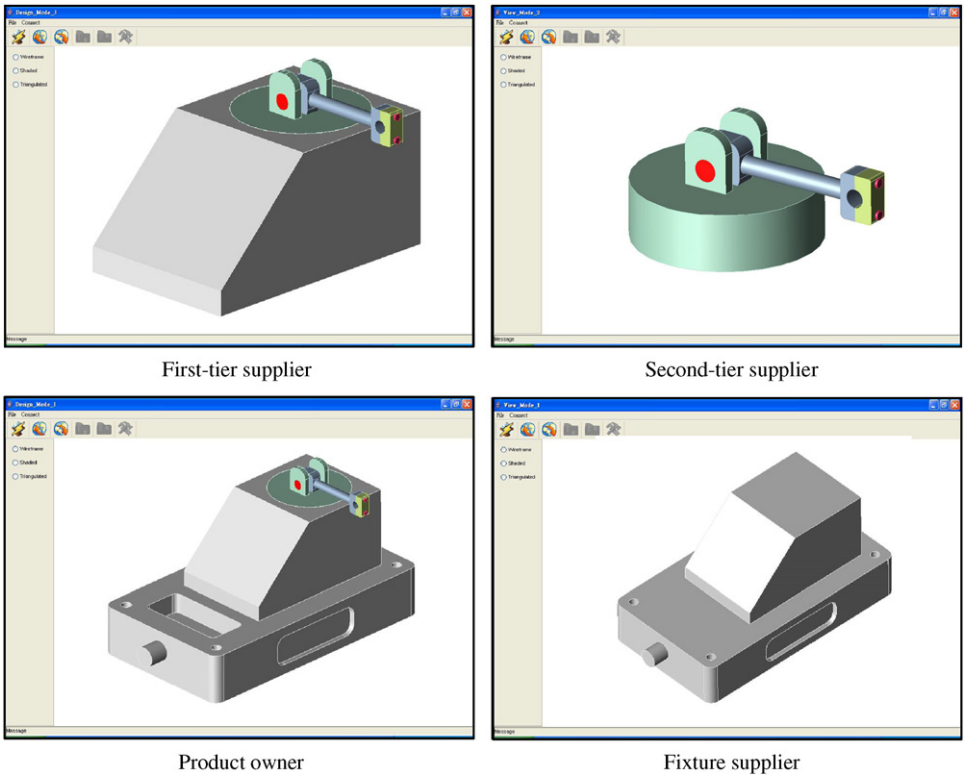

Fig. 15. Screen shots for different participants after the second assembly.

mechanism is required to overcome this potential problem. As proposed in the previous research [31], three approaches can be used to facilitate this mechanism: rule-based, knowledge-based, and case-based reasoning. Considering the characteristics of intelligent agents, we suggest adoption of the first approach, i.e. to impose rules-based induction capability in the agent programs. Alternatively, a real person supervising the collaboration process can be informed when a conflict occurs, and thus reconcile this situation manually. In addition, online modification of design features based on mesh model is a powerful technique in 3D design review, one of the critical activities in engineering collaboration. This technique fits well within the agent-based framework, in which a design change is automatically initiated by some agent program while the actual model modification can be performed by a client-side utility. The LOD tree provides an effective representation for implementing it. Finally, the prototyping system serves as a fine platform for realization of 3D CAD streaming enabled by the LOD scheme. A CAD model can be divided into a series of smaller data packets by LOD (or feature). A token indicating the LOD information is attached to each packet so that the system can pick up the right location for continuing transfer when the data stream is interrupted in an unexpected manner. The model can also be updated incrementally for rendering at the destination side upon receiving the data of each LOD. It does not need to wait until the transfer of the whole model is finished. We are currently working on this technique.

Acknowledgments

This work was supported by the National Science Council of Taiwan under grant number NSC 94-2213-E-007-064. The authors would like to thank Mr. Mu-Chi Sung at National Chia Tung University for his assistance in the system implementation.

References

[1] Chu CH, Chang CJ, Cheng HC. Empirical studies on inter-organizational collaborative product development in Asia Pacific region. ASME J Comput Inf Sci Eng 2006;6(2):179–87.
[2] Hoppe H. Progressive meshes. In: Proceedings of the 23rd annual conference on computer graphics and interactive techniques; 1996. p. 99–108.
[3] Rossignac J. Edgebreaker: connectivity compression for triangle meshes. IEEE Trans Visualization Comput Graphics 1999;5(1):47–61.
[4] Frey PJ, George PL. Mesh generation. Oxford: Hermes Science Publishing; 2000.
[5] Pebay PP, Thompson D. Communication-free streaming mesh refinement. ASME J Comput Inf Sci Eng 2005;5(4):309–16.
[6] Bellenger E, Coorevits P. Adaptive mesh refinement for the control of cost and quality in finite element analysis. Finite Elements Anal Des 2005;41(15):1413–40.
[7] Fuxin F, Edlund S. Categorization of geometry users. Concurrent Eng: Res Appl 2001;9(1):15–22.
[8] Fuxin F. Configurable product views based on geometry user requirements. Comput Aided Des Appl 2004;1:377–86.
[9] Cera CD, Kim T, Han JH, Regli WC. Role-based viewing envelopes for information protection in collaborative modeling. Comput Aided Des 2004;36:873–86.

[10] Cera CD, Braude I, Kim T, Han JH, Regli WC. Hierarchical role-based viewing for multilevel information security in collaborative CAD. ASME J Comput Inf Sci Eng 2006;6(1):2–10.
[11] Qiu Z, Fuh JYH, Wong YS. Secure CAD model retrieval and data consistency: issues on role-based visualization. Comput Aided Des Appl 2006;3(1–4):139–44.
[12] Chao KM, Norman P, Anane R, James A. An agent-based approach to engineering design. Comput Ind 2002;48:17–27.
[13] Hadeli, Valckenaers P, Kollingbau M, Van Brussel H. Multi-agent coordination and control using stigmergy. Comput Ind 2004;53:75–96.
[14] Shen W, Barthes JP. An experimental multi-agent environment for engineering design. Int J Coop Inf Syst 1996;5(2/3):131–51.
[15] Zhao G, Deng J, Shen W. CLOVER: an agent-based approach to systems interoperability in cooperative design systems. Comput Ind 2001;45:261–76.
[16] Anumba CF, Ugwu OO, Newnham L, Thorpe A. A multi-agent system for distributed collaborative design. Logist Inf Manage 2001;14:355–66.
[17] Tang D. An agent-based collaborative design system to facilitate active die-maker involvement in stamping part design. Comput Ind 2004;54:253–71.
[18] Park H, Cutkosky M, Conru A, Lee SH. An agent-based approach to concurrent cable harness design. AI EDAM 1994;8(1):45–62.
[19] Sun J, Zhang YF, Nee AYC. A distributed multi-agent environment for product design and manufacturing planning. Int J Prod Res 2001;39(4):625–45.
[20] Shen W. Distributed manufacturing scheduling using intelligent agents. IEEE Intell Syst 2002;17(1):88–94.
[21] Dornfeld DA, Wright PK, Wang FC, Sheng PS, Stori JA, Sundararajan V, et al. Multi-agent process planning for a networked machining service. Trans NAMRI/SME 1999;27:191–6.
[22] Song Y, Lee K. Incremental transmission of B-Rep models through the network. Comput Aided Des Appl 2004;1:523–30.
[23] Lee SH. Feature-based multiresolution modeling of solids. ACM Trans Graphics 2005;24(4):1417–41.
[24] Li WD, Lu WF, Fuh JYH, Wong YS. Collaborative computer-aided design: research and development status. Comput Aided Des 2005;37:931–40.
[25] Chen XL, Fuh JYH, Wong YS, Lu YQ, Li WD, Qiu ZM. An adaptable model for distributed collaborative design. Comput Aided Des Appl 2005;1:47–55.
[26] Chu CH, Wu PH, Hsu YC. Collaborative 3D product development with multiple levels of detail in visualization of design features. Comput Aided Des Appl 2006;3(6):789–802.
[27] Nwana HS. Software agents: an overview. Knowledge Eng Rev 1996;11(3):205–44.
[28] 〈http://jade.tilab.com/〉.
[29] 〈http://www.fipa.org/〉.
[30] 〈http://www.caav5.com/〉.
[31] Li X, Zhou X, Ruan X. Conflict management in closely coupled collaborative design system. Int J Comput Integrated Manuf 2002;14(4):345–52.