Computers in Industry 60 (2009) 381–391

Contents lists available at ScienceDirect

Computers in Industry

journal homepage: www.elsevier.com/locate/compind

Collaborative feature-based design via operations with a fine-grain product database

Y.-S. Ma [a,*], S.-H. Tang [b], C.K. Au [c], J.-Y. Chen [d]

[a] Dept. of Mechanical Engineering, 4-9 Mechanical Engineering Building, University of Alberta, Edmonton, AB T6G 2GB, Canada
[b] School of Mechanical and Production Engineering, GuangDong University of Technology, China
[c] School of MAE, Nanyang Technological University, Singapore
[d] ST Microelectronics Pte. Ltd., Singapore

ARTICLE INFO

Article history:
Available online 24 March 2009

Keywords:
Collaborative engineering
Feature-based engineering
Fine-grain product database

ABSTRACT

This paper reports a collaborative product design framework and a prototype system that supports multiple CAD systems. The key contribution is an 'operation'-based, multi-application oriented, and near real-time collaboration mechanism which can significantly reduce collaboration communication load over the network. The mechanism is discussed and demonstrated with examples. To support the proposed multi-application collaboration system, a fine-grain feature-oriented product database is used. This research is a continued effort based on a shared common product modeling scheme, which covers fundamental issues of generic feature, feature level interoperability, engineering intent and operation definitions.

1. Introduction

1.1. Literature review

Due to the competition pressure and rapid change of market, shortening time-to-market has become the critical objective for many companies. As a result, concurrent and collaborative engineering (CCE) become an industry trend. In a CCE environment, it is common for engineering tasks to be carried out by a group of engineers who may be distributed in terms of both time and space. Furthermore, different engineering partners use different applications; hence a product model generated from one application system has to be used directly by other ones. Currently, information sharing among multiple applications becomes the bottleneck for CCE [1].

There are two strategies for information sharing in computer readable form, using proprietary direct translators or a public domain neutral intermediate format. Compared with direct translators, the neutral intermediate format strategy is suitable for information sharing among a large number of applications. In order to enable such interoperability, a commonly acceptable, comprehensive and well-defined information model is crucial. A lot of efforts have been made to model product entire lifecycle for the implementation of CCE. The existing standards for data exchange include IGES, VDAFS, SET, STEP, etc. For example, STEP is the most popular one and regulates mainly geometric information of a product although it was intended to cover the entire lifecycle of a product, i.e. from design to analysis, manufacture, quality control testing, inspection, and product support functions [2].

Under the STEP framework, Szykman and Sriram presented a heterogeneous system integration method and contributed to CCE on database information sharing [3]. Zhang [4] also presented a web-based data exchange framework to provide STEP data translation services for virtual enterprises. However, during data translation, some useful geometric and non-geometric information such as features and their associated semantics are usually stripped off. Therefore, this approach does not support complete information sharing.

Currently, most of the CAx systems are feature-based. Features are found very useful to encapsulate engineering intent in computer systems. Therefore, representing feature-level information uniformly is required so that engineering meaning is fully shared among CAx applications. Many researchers proposed to use design geometry information as the input to derive downstream application features by feature conversion [5–9]. A multi-view feature modeling approach that was supported by feature links was proposed [8,9]. An "associative feature" definition was developed in [10,11] for establishing the built-in object-oriented mechanisms among related geometric entities. Such features are application-specific and multi-facet features while self-validation methods were defined for keeping feature validation and consistency. Compared with the one-way feature conversion approach, such a multi-way feature association approach is more promising to support multi-view product modeling because of the

* Corresponding author. Tel.: +1 780 492 4443; fax: +1 780 492 2200.
E-mail address: yongsheng.ma@ualberta.ca (Y.-S. Ma).

0166-3615/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.compind.2009.02.013

object technology's scalability and the possible implementation for lifecycle servicing methods. Recently, Chen et al. proposed a unified feature modeling scheme [12] related to inter-application association management for high-level feature information sharing among different CAx applications. The unified feature model is essentially a generic semantic feature model for different CAx applications covering relations among geometric and non-geometric entities. This unified feature model can be further extended to knowledge-based models by incorporating reasoning techniques [13].

To support collaborative design [14] based on feature modeling, web-based systems were developed in a semantic modeling approach [15]. In some of them, enhanced multi-view feature models [9,16] were adopted which can maintain feature semantics among CAx applications. Mechanisms for feature model validity and feature conversion are described. Li et al. [17] presented a system to support feature-based design in distributed collaborative environment. The 'face-based' data structure in the client side can greatly reduce communication load between the server and client. A modeler-supported server provides feature modeling and feature validation functions in the server side. These efforts have illustrated that features can be used as the intermediate medium to express design and engineering modeling details with reduced transferring load across the network. Unfortunately, the interoperability issue was not addressed because different semantic schemes are used. For example, in [17], Open CASCADE and Java development toolkit were used.

Another more generic approach is to make use the modern database technology. A database allows the handling of a large volume of data and is generic for reading, writing, updating and deleting operations. The database management system (DBMS) can ensure the security and transparency for the users of CAD data. Therefore, compared with the traditional file-based approach, databases are appropriate tools for information sharing among multiple applications.

Kim and Han [18] described an interface (OpenDIS) between the geometric modeling kernel and the DBMS for the implementation of CAD system that uses the STEP database as the native storage. A prototype CAD system was implemented using the Open CASCADE geometric modeling kernel and ObjectStore. The STEP methodology was used for the database schema. However, currently, STEP files are very large in size and cannot fully support information for different CAx applications, particularly for feature-based applications; so directly using STEP format files as the network-based collaborative engineering medium is not satisfactory.

Other than research efforts, there also exist commercial efforts which can support CCE to some extent, such as CAD web portals developed by CAD/CAM-E Inc. [19], OpenDXM by ProSTEP [20]. All these web portals are in fact operated by translator providers. During data translation, useful information such as features is not maintained. PDM/PLM solutions, such as Unigraphics' TeamCenter, Pro-E's Windchill can be basically categorized as a kind of file-based data management environments and cannot provide flexible information sharing with finer-grain levels of granularity. There is one commercial system, OneSpace by Cocreate [21] currently offering some collaborative modeling capabilities. However, its modeling facilities are severely constrained by the modeler at the server SolidDesigner, and by the model format into which it converts all shared models [14]. Although a lot of research and development work have been done to enable CCE, the following problems still exist:

(1) *Duplicated data and conflicts* [22]. Product and process information is often stored in file format, which means duplicated data and potential conflicts. In addition, files are not flexible enough to support the multi-view functions required by CCE applications. Furthermore, it is difficult to extract and manage useful information from distributed files. For example, multiple end-users or applications cannot synchronize definitions and modifications easily for the data stored in individual files.
(2) *Information loss*. As for geometrical information, although many existing systems claim to support CAD-neutral (e.g. STEP-based) data exchange, they lack feature level interoperability so far. Useful information such as features is often stripped off during data exporting and importing. Therefore complete information sharing has not been achieved.

Therefore, the research work presented in this paper, that is to achieve feature level information sharing among multiple applications with a feature-oriented product database support, can be justified. This paper has seven sections. After this introduction, Section 2 gives a description of a comprehensive and neutral product model definition of an entire product model and related integration interfaces. Section 3 introduces the proposed method to synchronize multiple application systems via feature-based operations. Section 4 describes interactions between the engineering applications and the product database while Section 5 briefly introduces the whole system architecture. Section 6 demonstrates the prototyped system with cases. Section 7 draws conclusions.

2. A comprehensive and neutral product model

2.1. Information integration infrastructure

To achieve information integration, a four-layer infrastructure was proposed in [23] consisting of application, information, representation and physical model layers. In the application layer, different feature-based functional application interfaces and procedures, e.g. design, tooling, manufacturing, are represented as sub-modules; they need to be developed according to application-specific requirements.

Next, the information layer contains four components again. (1) The first component is a meta-product model, named as entire product model (EPM). The EPM component describes information dependencies across applications, and contains the domain classification ontology and metadata. It also contains assembly-part models, product geometry and topology, and the related attributes. (2) The second component in this layer is the 'unified feature' [12] which provides a generic modeling framework for different application feature modeling by giving a set of generic feature templates. Although different applications define features in different ways, their features can be fully represented with a set of common types of data entities in the categories of geometry, topology, dimensions, tolerances, constraints and parameters, etc. In addition, different application features refer to the same master product geometry. (3) The third is the application feature component. Detailed feature objects are organized by different sub-application models, or specific application views for the product development processes. (4) The last but not least, is the EXPRESS specification component which offers the formal definitions of information modeling structures, graphical interfacing specifications for the EPM, the unified feature model and the application sub-models [24]. This component is useful for future application integration reference and protocol implementation.

Then, in the representation layer, for implementation, the EXPRESS-defined information models need to be mapped to database schema for data storage, programming data structures like 'workform' format, and neutral content format for data communication.

Finally, in the physical layer, a network-oriented database provides a repository of information entities, e.g. geometrical entity data, feature properties and others objects' data. In the database, all kinds of information are stored as data elements across database tables. Such an open yet neutral data structure makes the information sharing natural, flexible and with different levels of granularity.

2.2. Generic feature model

From the application point of view, it is essential that each feature type has a well-defined meaning or well defined semantics. However, to address feature-level interoperability, feature definition in an 'open' and neutral format is important. Here 'open' format means the generic definition of features allow different instances of user-defined feature types to be incorporated into the working collaboration framework either automatically or interactively with minimum mapping effort. The generic feature class introduced in [23] offers the structured description of all common properties and methods of application feature types. Such properties include feature shape representation with parameters, constraint types, reference mechanisms and validity methods. This generic feature model also provides a template for application-specific feature definitions.

2.3. Co-existence of different application features in geometry representation

The unified feature model allows different applications to define different features, even though there could be physically overlapping in space or volume [25]. They are associated to the same master product model via a cellular model of geometry representation. A cellular model represents a part as a connected set of volumetric quasi-disjoint cells [26]. By cellular decomposition of space, cells are never volumetrically overlapped. As each cell lies either entirely inside or outside a shaped cell, a cell can be used to construct multiple features. A feature shape can be represented explicitly as one cell or a set of related cells in the part. Explicitly maintained feature shapes in the product model make feature association and reference persistent in application implementation.

3. A method to synchronize multiple application systems via feature-based *operations*

For developing a collaborative engineering platform, one of the major issues is engineering data sharing/exchange. For example, the sharing/exchange process in a collaborative CAx platform should be dynamic, consistent and able to support various CAx systems. In this interoperability issue, the key challenge is modeling the interactions among functions of different systems. In short, a synchronous and dynamic functional interoperability is needed for a collaborative CAx platform. However, owing to the large sizes of 3D geometry files, data transmission among different CAx applications through the Internet is very time-consuming and unreliable. Although incremental data file transfer is an alternative way to reduce the network-load where only the modifications instead of the whole CAD model are transferred incrementally during collaboration, but even the first-time downloading is still a very time-consuming task. To the best knowledge of the authors, there is no reported research that can meet the feature-level interoperability requirements. In this research, a new method, based on feature operations, is proposed to address the issue.

3.1. A premise

The parametric feature technology is one of the core technologies in the current CAD/CAM systems. Engineering patterns are defined as templates that are associated with dimensions, constraints, attributes and are geometrically represented by a set of entities and controlling parameters. In the past three decades, parametric feature modeling has been proved to be effective to support many kinds of design strategies (e.g. top-down or bottom-up, etc.) and approaches (e.g. initial design, resembling design and variant design, etc.). In addition, it can capture the engineering information and maintain flexible models.

The proposed method is an extension of the parametric feature modeling technology, but there is a premise. A set of commonly used neutral features can be achieved in equivalent forms with different application systems by mapping, grouping, combination, or manipulating existing predefined feature types explicitly. Let us first discuss about the soundness of this premise. As reported in Chen et al. [27], to simplify and verify this premise, some basic form features were selected. By comparing some popular commercial CAD software tools such as Unigraphics, Pro/E, Catia and SolidWorks and the design form features defined in STEP [2], it can be concluded that their form features can be defined by using another commercially available feature scheme.

Here, sketch feature needs to be explained. Strictly speaking, a sketching module is a tool that provides convenient means to define and create a feature. A sketch is not a form feature in fact. A sketch includes the attached plane and a two-dimensional boundary. By sketching a contour of the desired feature, the designer defines the topology of the feature by its edges and vertices. Then by constraining the sketch, some information, such as position, orientation of the desired feature is defined. Thus the sketch is a special feature that contains information of the resulted form feature it creates. Sketching function has been employed by most current commercial CAD systems as a tool to create form feature. So 'sketch' is selected as a basic feature to support the proposed method for data sharing/exchange.

3.2. The concept of operation

An operation is defined as a set of associated commands (or methods), which are responsible for the functions and manipulations of a commonly acceptable or equivalent set of feature entities among multiple engineering application packages. This set of commands is usually feature-based and can be directly used to support the interfaces of the central databases with different feature-based systems. Hence, by identifying a generic functional processing object type named as operation, different feature level properties and processes can be standardized and manipulated with generic methods such that they can be commonly supported by object technology within individual engineering application systems. The implementation of operation class and its applications have been reported in [27]. In essence, the interpretation of such commands would be dynamically associated with the specific feature functions defined within the interfacing application system. Therefore, the standardization of operations that are supported by different systems is the critical condition if this approach is to be adopted in real implementation.

3.3. Data type definition for operations

Supporting operations is a basic requirement for the generic feature definition. It is also important for the manipulations of the associative feature model, especially for distributed collaboration system implementation over the web because the communication load between distributed client and database server can be reduced significantly in comparison to communication with lower level geometric data.

Operation definition is represented as a generic and concise object type in EXPRESS-G as shown in Fig. 1. An operation entity

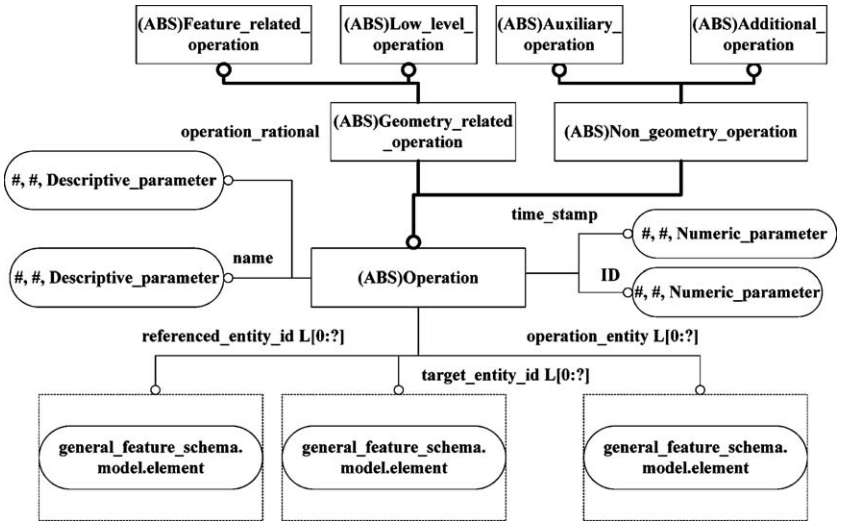

Fig. 1. Operation definition.

has its *name* and *ID*. An attribute named *time_stamp* is used to record time sequence during a collaboration session. An operation records the entities to be created or modified in an *operation_entity_list*. In the *referenced_entity_list*, entities that are related to a particular operation are recorded. For example, when an operation that re-constrains a feature with reference to an element of another feature is sent, the old and new *constrained_entities* and *referenced_entities* are recorded in the *referenced_entity_list* such that the application, which receives this *operation*, can easily match the corresponding entities in its application. Such matched entities in the receiving application are recorded in the *target_entity_list*, which is used for model update according to the operation. An *operation_rational* specifies what kind of action the operation will do to the operation entity, e.g. for feature-related operation, *operation_rational* attribute specifies the actions such as *add, delete* or *modify*. Operations can be flexible in record size according to different synchronization intervals and could be nested.

As categorized by [27], operations have two types, namely geometry- and non-geometry-related operations. The geometry-related operations can be further classified into feature-related and low-level operations according to the entities that they manipulate. The low-level operations are to create or to modify low-level entities, such as points, lines and faces. The feature-related operations (feature operation) include instantiating a feature or modifying a feature. Non-geometry related operation could be divided into ''auxiliary'' and ''additional'' operations. ''Auxiliary'' operations are mainly to facilitate the designer in geometric modeling but do not affect the geometric shapes, such as layer management and view manipulation. Other non-geometric operations can be classified into ''additional'' group, such as those related to file management.

Note that the time stamp in each operation has a set of time-based attributes to record the sequence of the confirmed decisions made upon the generation of each operation (e.g. when the user confirms by clicking 'save' or 'update database') after its contents are delivered, verified and executed. Operation records are also stored into the product model database with their references associated to the operation ID, target features, referenced entities, parameters, etc., together with their sequences. Therefore, generally, product model can be regenerated from time to time by running through the operations recorded, and if necessary, operations can be rolled back and forth if milestones or stop points are inserted so that more flexibility or variations for engineering activities can be supported. Clearly, within the product database, features are static in nature recording the existing model entities, relations, constraints while operations are sequential and time based. By cycling the associated operations of any feature, its construction history can be traced for verification or future auditing purpose.

3.4. Generic feature-based collaborative engineering with the operation concept

As illustrated in Fig. 2(a), a traditional CAD system consists of several layers, such as data repository file, runtime database, modeling kernel, application functions and CAD modular applications. The modeling kernel deals with the low-level geometric data and mainly takes charge of creating, maintaining and displaying the geometric model. It provides basic functions for the application layer to access the database and data repository (usually in the form of files). The modular application layer is an intermediate layer which provides functional interfaces and controls the sessions. It acts as a bridge that connects the user functions to the lower level generic application functions and further to the modeling kernel. This layer enables a series of User Interface (UI) functions for the designer to input the design data and design intent into the CAD system. According to the design intent, the designer chooses a group of operations, such as creating a part. The application layer extracts the information of the UI functions and calls one or more functions provided by lower level sub-systems, such as the generic application module and the modeling kernel. During this process, the design data and knowledge are converted and embedded into the lower level CAD data and stored in the repository via the modeling kernel. According to the input, the modeling kernel modifies the geometric model and updates the display. So far, majority of such repositories are based on part and assembly files instead of scalable neutral databases.

Now, a new collaborative engineering mechanism is suggested here by introducing operations into the information flows. Operations supported by a CAx system act as interfacing message carriers for the user to access the system data and functions as illustrated in Fig. 2(b). An engineer uses Graphical User Interfaces of the given CAx system and generates operations incrementally to design, plan, reprocess and evaluate his work on top of the feature-based product model [23]. Take a typical design process as an example. The designer converts the design concepts and rules into a group of incremental operations; and then through these

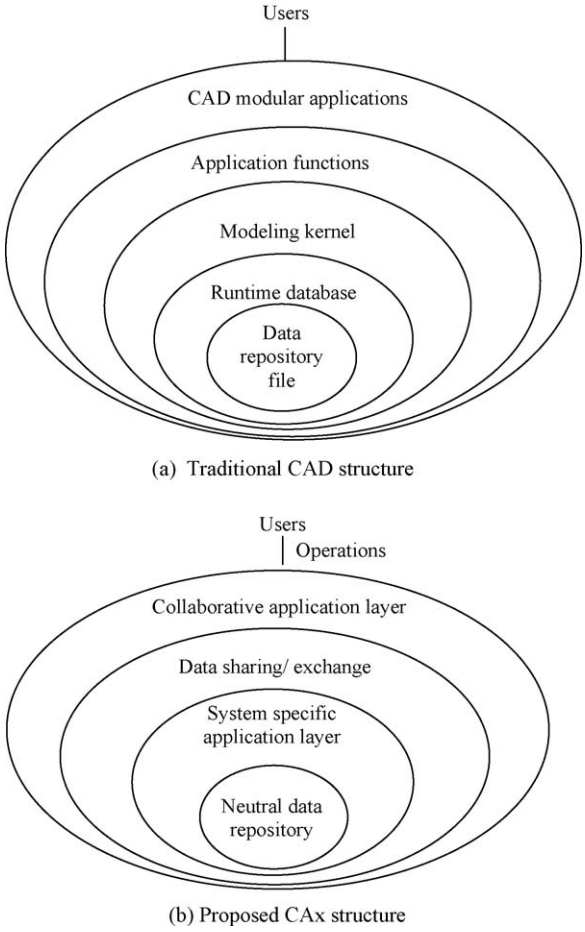

Fig. 2. Schematic comparison of traditional and proposed CAx structures.

operations, the design data and design intent can be created and consistently maintained by a Web-based product model database system without the tedious system-specific feature definitions. Such a 'plug and play' protocol is based on a common set of operations required for the functionality of the related engineering processes.

If all the CAx systems can interact with a set of standardized neutral operations, then the feature level interoperability can be expected. Fig. 3 shows the virtual mapping method between a system-specific feature command scheme and a standardized operations-based scheme. Note that the corresponding mapping tools between the specific CAx system and the neutral and generic feature system have to be developed and made available automatically upon legitimate requests across the network by the web service mechanism. In addition, such operations can generate and manipulate even higher level information than features, such as product meta-data and semantics embedded in engineering modeling processes. The communication language which consists of operations, reference entities and other associated parameter values can be defined as a high level 'feature markup language' (FML) and used for information transfer between the specific CAx client and the neutral server databases.

4. Interactions between engineering applications and the fine-grain product database

Once the standardized operations become the acceptable communication vehicle with an agreed vocabulary collectively to support the incremental changes sent to the product repository, the interaction methods between CAx systems and the database can be developed. Here, the 'fine-grain' product database needs some elaboration. In informatics, grain size is a concept of accessible object level that the end user or developer can make use of for the required inquiries and manipulations. In the context of collaborative system, the grain size of interoperability refers to the neutral and abstract level of information entities that can be accessed via common interfaces. As to the collaborative engineering platform, the informatics grain size indicates the level of flexibility of manipulation and information sharing in the collaboration sessions. In the current available commercial CAx systems, the grain size of interoperability is at the part file level and for geometric entities only attributing to some geometry exchange standards, such as IGES and STEP. A fine-grain product model was proposed in [23,25] because in the proposed system design, all the CAD entities, including geometrical and non-geometrical ones, are extracted and stored into a generic product database. Since the accessibility to all levels of entities, from points, edges, faces, to slides and features is expected to be possible, hence the proposed approach is 'fine-grain' for the product repository. Here, the 'fine-grain' entities refer to the neutrally accessible objects that can be manipulated.

Operations are dynamically created relating to the stages of the end user's activities. It is mapped into standard methods related to generic features as well as the sequence of executions. The results generated are converted into static features and other low level geometric and non-geometric entities. Using operations provides a dynamic collaboration mechanism for users to cooperate on a single product model via different CAx interfaces and rolled back and forth easily. Operations enable engineering activities to be 'saved' incrementally and allow such activities continued anywhere and anytime. This mechanism provides the means of capturing the user's design sequences and making the feature-based engineering activities valid in a very effective manner. Fig. 4 illustrates the interactions between the cooperating CAx and CAD systems and the network-based product model database. This interaction method is supportive to the generic feature representation schema described in [25], and the feature representation in database as reported in [23].

5. System architecture

As shown in Fig. 4, the proposed repository system adopts a client–server architecture. The functional servers include a web server, an application object server and a feature object server to provide different functionalities. In the following section, the roles of these functional servers are described briefly. For more detailed information, refer to [28].

5.1. Web server

The web server contains a collaboration manager, security manager and session manager. Collaboration manager provides two kinds of accesses for multiple users, namely, "HTTP" access through MDAI (multi-view data access interface) by adopting ASP (active server pages) and direct socket-based access. It can support data check-in and check-out by multiple users through the web; it also supports socket-based collaboration among multiple users. The security manager is used to check whether the user has the right to access the product model data and what kind of access right he may have. Users are classified into several groups. Each group has different access rights. Such management data are stored in the database. When a user signs in through collaboration manager by his user name and password, the security manager will check the user's information stored in the database. Then according to the role of the users, e.g. product designer, tooling designer, process planner, or CAE analyzer, selective views of the

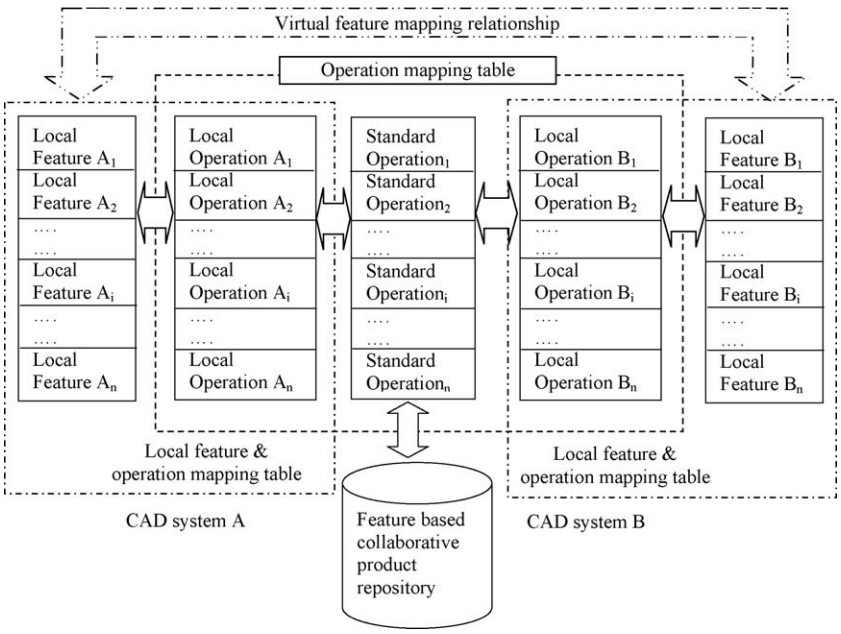

Fig. 3. Virtual feature mapping mechanisms.

product model data will be provided for the user [29]. The session manager is responsible for controlling concurrent access by multiple users of the same data, and controls concurrent access of product data.

5.2. Application object server

The application object server provides categorized information services (e.g. design or manufacturing applications) for different users such that users can interactively carry out feature-based operations via generic user interfaces (to be developed) with the support of the dedicated views by a product model manager. The product model manager is responsible for managing different application views as an integrated product information model and maintaining the information consistency in the product level.

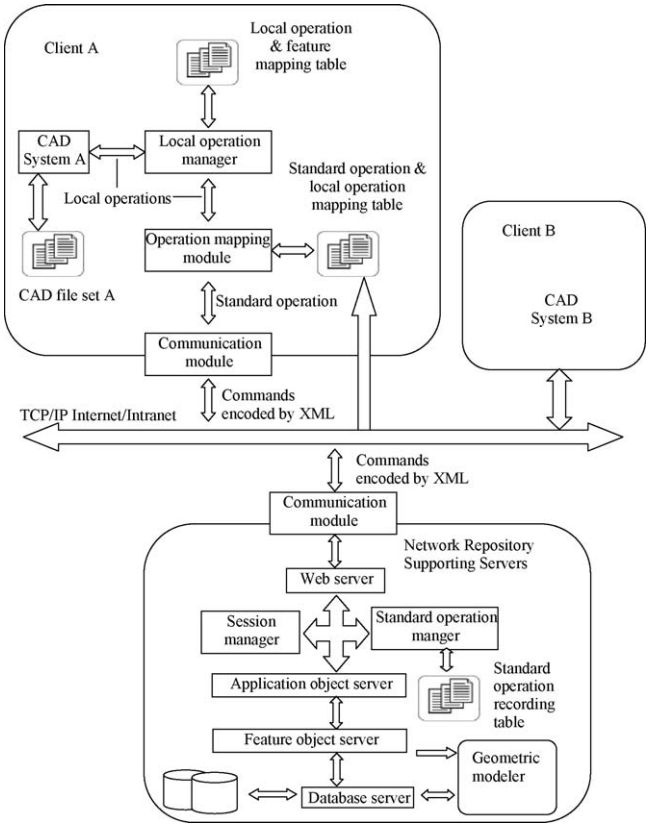

Fig. 4. System architecture.

5.3. Feature object server

The product information model includes different feature models with feature level constraints, inter-application constraints and the underlying solid models (via cellular models). The feature object server maintains different application feature libraries and therefore can provide feature object methods for application packages. To maintain the meaning of a feature during feature modeling operations, such as adding, deleting and modifying features, the feature manager calls the constraint solver and the geometrical modeler to create/update all feature instances based on those feature creation/manipulation functions and constraint solving functions defined in application-specific feature class. The constraint solver can check the validation of all constraints, which are part of the feature definition. The geometrical modeler here is used also to validate feature geometry.

5.4. Database server

The database server provides physical storage for all kinds of data including product model data, security management data and so on. Within the database, geometrical data and features for different applications are stored as data elements across tables. Database manager can provide data manipulation functions (e.g. save, restore and validate functions) with the help of geometrical modeler. These functions are used to organize information for different application views according to users' requirements.

5.5. Geometrical modeler

In the proposed system, a solid modeler has been tightly integrated into the repository server with interfaces to the feature object server and database server. It provides all the geometry-related services such as creating and modifying geometrical entities [25].

5.6. Information flow

In the proposed system, when a user wants to check-in or check-out product information, he can access through ASP. Then, the security manager, by connecting to the database manager, will check the user information stored in database, and decide whether the user has an access right, what kind of access right he has, and what kind of information (application-specific view) he wants. Finally, he can check-in or check-out product information. If multiple users want to collaboratively design a product, the participants have to register first with the collaboration manager. After security checking, a new session is established according to the user's profile. Similarly, multiple users can collaboratively work on the product through the application object server where the product manager plays the role of the information broker. During modeling collaboration, feature-level operations are communicated in most cases so the communication load among distributed clients and server is minimized. The application object server, through product model manager, consistently manages product information in the server side. After each feature operation, such as insertion, modification and deletion, the geometrical modeler will be called to process and validate the feature geometry. The feature manager will also call constraints solver to check all the involved constraints (intra-application constraints and inter-application constraints) to validate the feature model. Finally, the finished product model can be stored into the database by the database manager.

6. Prototype system implementation

To prove the practical effectiveness of this operation-based collaborative engineering mechanism, preliminary prototyping efforts have been carried out. The first question to be answered is whether there is a set of mutual-equivalent features between two different CAx systems?

6.1. Feature-based modeling with operations

To address the above question, a client server testing environment has been set up. The database server is connected to two client workstations. Two different client CAD systems are installed respectively, one is UGS NX v.2 and the other is SolidWorks 2004. A set of equivalent features have been identified and implemented into the operation data structures with the required mutual agreeable parameter structures. Fig. 5 shows a partial feature mapping table where equivalent features are paired between UGS NX and SolidWorks. Fig. 6 illustrates the detailed operation steps involved to construct a real industrial model. These operations have been implemented, and the modeling steps can be collaboratively executed or generated by any client who has the editing control [27]. The switching of editing control is also tested.

6.2. Feature-oriented database

In the prototyped system, the feature-oriented product database has been established on the basis of database schemas described [23]. For the database server, ORACLE 9i, an object-relational database solution has been adopted [30]. Product information includes geometrical entities, features (with parameters and constraints) and other information. High-level feature information connects with low-level geometrical data with the help of intermediate reference layer (feature labels). The use of database makes information sharing among multiple applications possible with great flexibility and granularity. Eventually, the proposed repository system is aimed to support highly intelligent reasoning for engineering decision making [31]. Fig. 7 shows a portion of the geometrical data (faces) stored in the database,

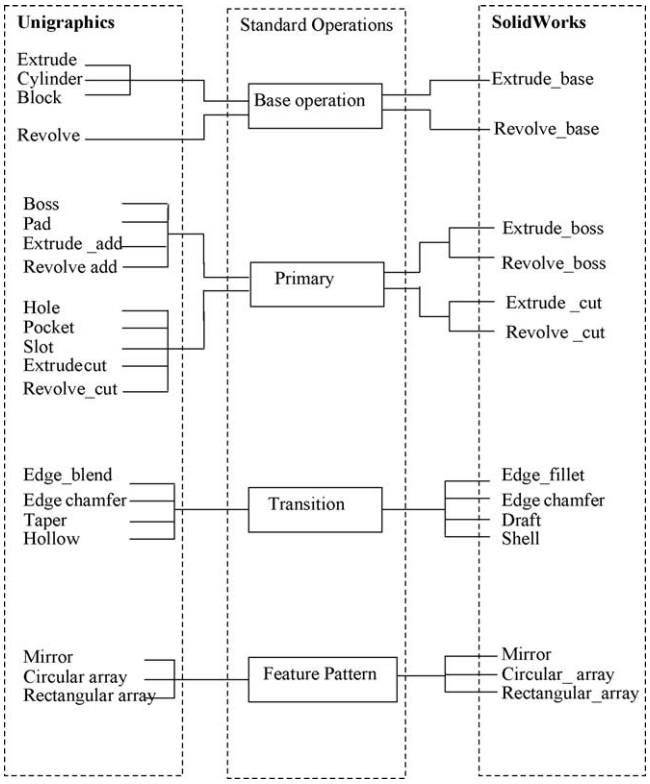

Fig. 5. Partial operation mapping table between unigraphics and solid works.

which belongs to a part *fix_jaw*. Fig. 8 lists all the design features of part *fix_jaw*. In the prototyped system, a database manager (DB manager) has been implemented to be responsible for reorganizing data elements stored in different database tables for different applications, such that the proposed system is multi-view supported. Generic *save*() and *restore*() functions have been developed [25] in order to manage feature information, geometrical data and operations. DB manager is connected with database server via OCCI (Oracle C++ call interface). OCCI is an API that provides C++ application access to data in an Oracle database. OCCI enables C++ programmers to utilize the full range of Oracle database operations, including SQL statement processing and object manipulation [30].

6.3. Web-based multi-client collaboration

In the prototype system, collaboration among two clients participating in a product design session has been tested. Both the clients are first registered in the session manager after security checking, then the collaborative design can be started by specifying product related domain information. Two design views and one manufacturing view participated in the design of *fix_jaw*, a part which belongs to a clamping vise product. As shown in Fig. 8, in the design view, the part *fix_jaw* constains seven features which are an *extrusion*, a *wedge_cut*, a *slot*, two *counter_bore_holes* and two *simple_holes*.

During collaboration process, only one system is allowed to edit by managing the process lock attribute. In the edit view, after each modeling operation, it will send feature operation instead of entire file to the server side. Note, in real application, the interval of two continuous operations can be flexibly determined in a batch of commands manner. Fig. 9 is a manually captured screen with an interactive viewer window showing the existing part loaded in the application and the sending operation control buttons from the edit view after adding two *simple_hole* features. In real situation,

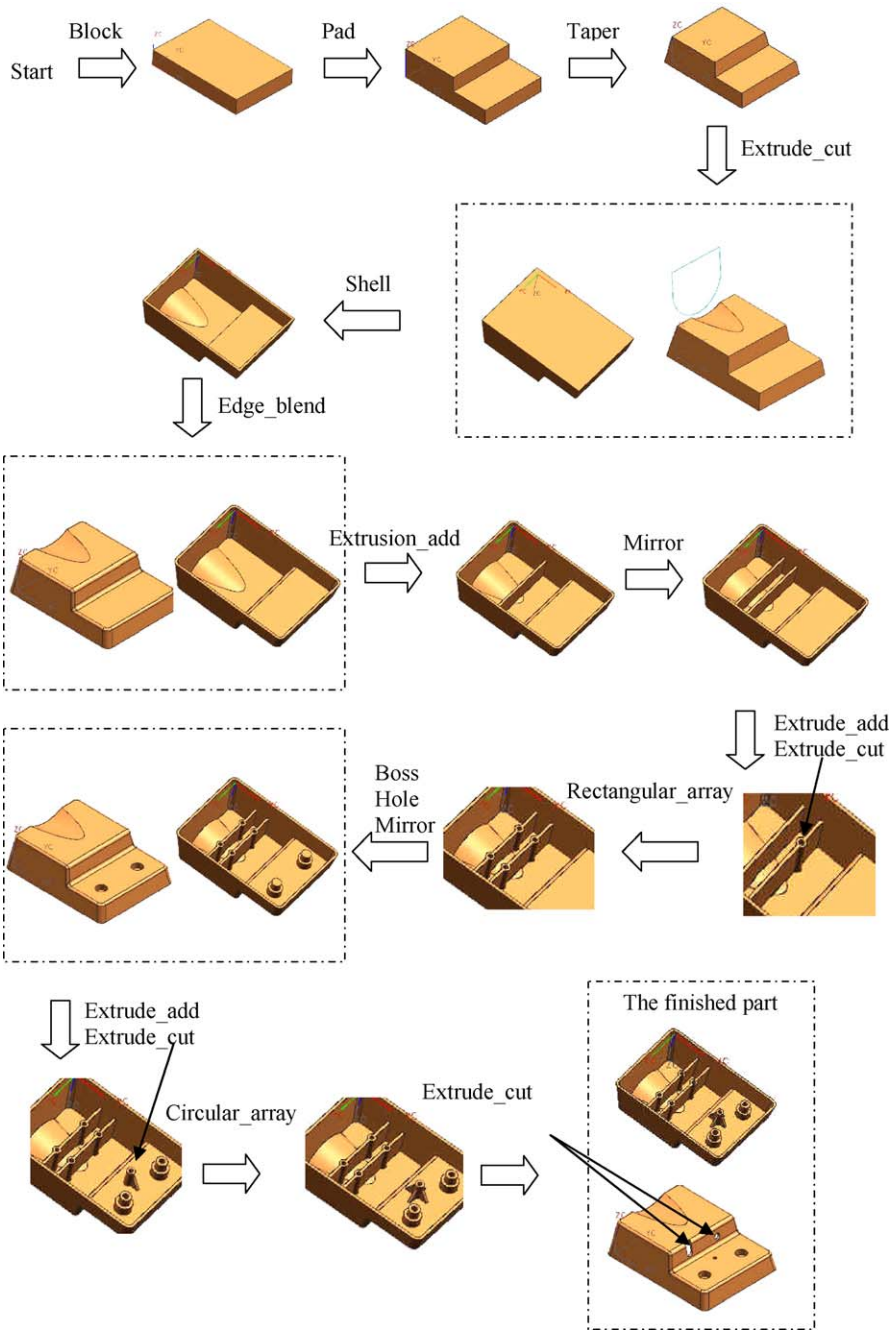

Fig. 6. An example modeling process by operations.

the sending process is automated transparently. In this case, the size of the feature operation file is only 4.07 kB, while the size of the whole part file is 28.4 kB. In real applications, the local part files can be huge depending on the complexity of geometry. Clearly, by sending feature operations instead of the complete CAD files or STEP files, the communication load between server and distributed clients can be reduced significantly.

Theoretically, upon receiving a feature operation, product manager in the server side re-evaluate the feature model using a constraint solver (partially developed) and keep a consistent runtime product model in the server side. According to the domain of the participants, different server actions will be executed. After the current set of feature operations being sent through network to the server side; then, the server side will validate the feature model. For those who have the same application domain with the current edit client, after the validation, the newly updated feature models will be sent to them and the corresponding views updated.

In the *fix_jaw* design example, after receiving the feature operation, the server side creates the two *simple_holes* according to their constraints and parameters (for details, please refer to [25]); then update the design view; finally, this view can evaluate and update the feature model. The result after model evaluation is the same as shown in Fig. 9.

For a client workstation which has different applications, the product manager first call the current application packages to re-evaluate the view. This kind of model evaluation contains two actions. If only '*modify feature operation*' is being processed, the corresponding application view will try to automatically propagate the changes by solving inter-application constraints and local constraints of the corresponding view. If automatic model re-

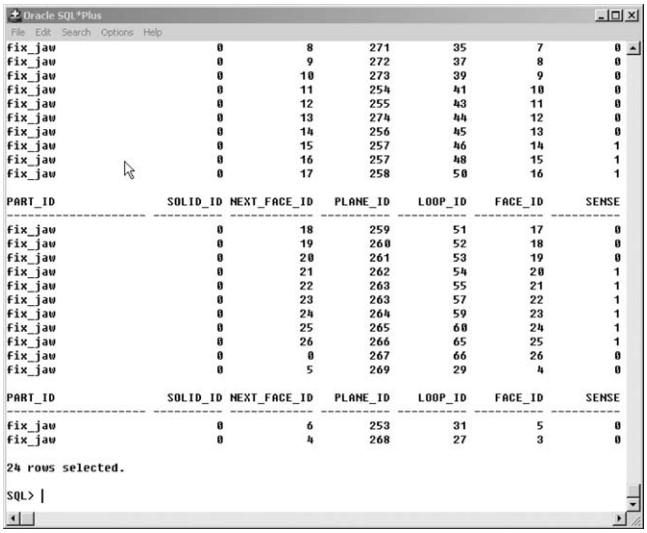

Fig. 7. Partial geometric data stored in database.

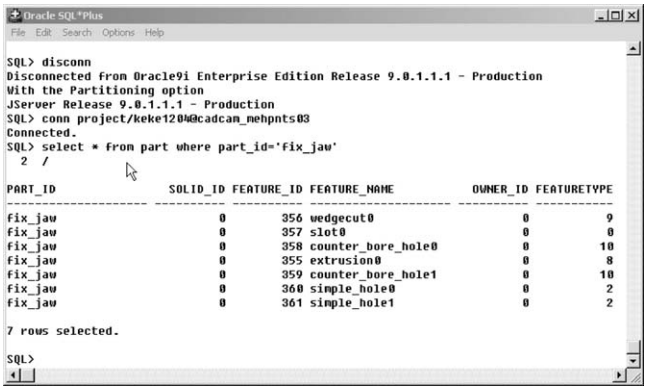

Fig. 8. The feature list of *fix_jaw* part.

evaluation is not successful due to constraint conflicts (or the view does not exist), the product manager will require the client to open the both views in the client side by downloading the entire file from the server side. Fig. 10 shows the result with the loaded design model together with the manufacturing view, where two *simple_hole* features are added in the design view. The interactive

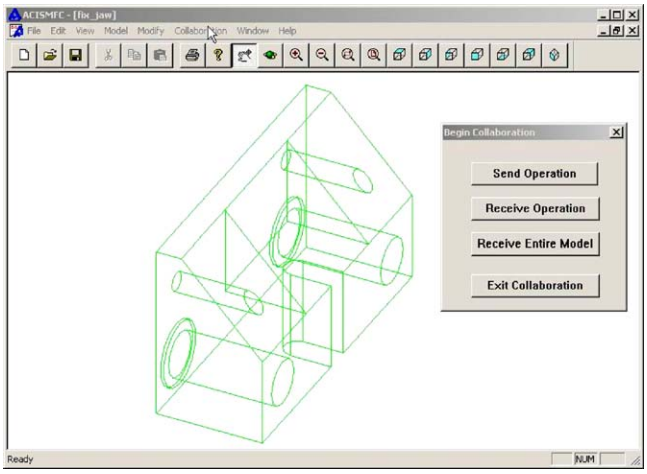

Fig. 9. *Send* feature operation from the edit view.

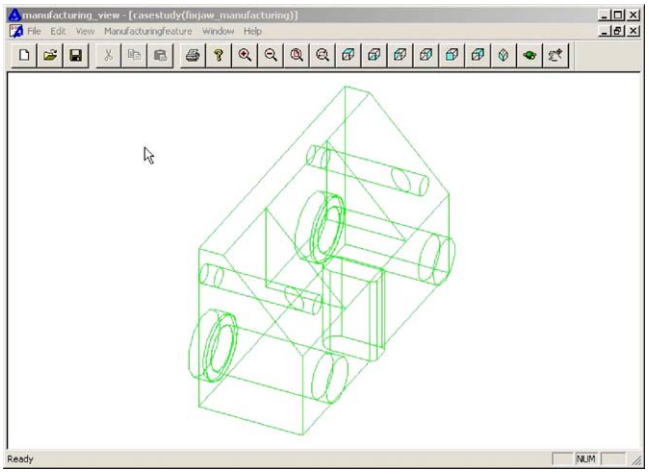

Fig. 10. The manufacturing features generated for the example.

resolving process is carried out as follows: firstly, create the stock automatically or manually by drawing on the basis of design feature model; then, interactively identify manufacturing features by predefined feature shape and choosing the machining tool as well as cutting tool; finally, a manufacturing feature model is generated. In such a way, multiple application collaboration can be realized. Note that manufacturing feature model is shown in Fig. 10.

Due to the space limitation of this paper, the actual operation records generated in the prototyped system are not presented. Although UG NX2 and SolidWorks 2004 are taken as examples, we have been convinced in the investigation that equivalent common features can be identified with other feature-based CAD systems. In addition, based on our observation, if such collaborative interoperability can be realized between two different CAD systems, it could also be applicable to different application systems such as CAD, CAM and CAPP systems.

6.4. Web-based access

The prototype system also supports online communication between distributed clients and database server through the web. For this purpose, all the clients should have the access to the check-in and check-out data services through the MDAI controlled by a collaboration manager. In the prototyped the system, the

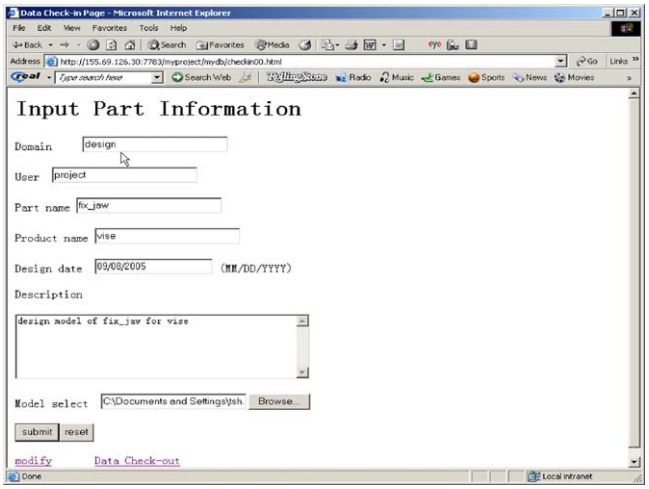

Fig. 11. Data check-in through MDAI part.

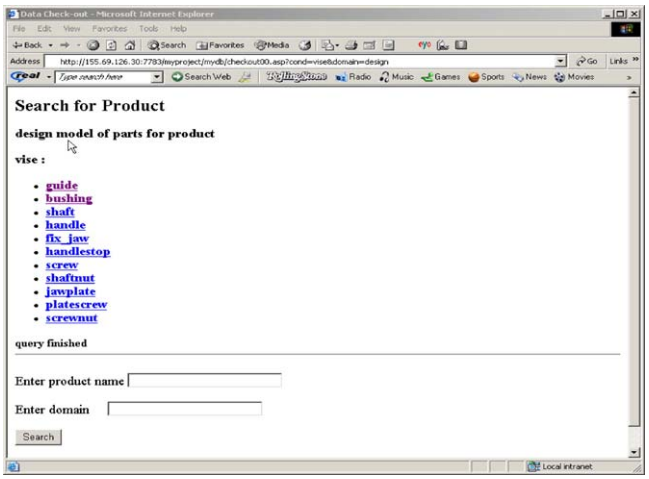

Fig. 12. Data check-out through MDAI.

MDAI has been established adopting the ASP (active server page) technology. Apache has been used as the "HTTP" server to provide data communication service through the web. A security manager in the web server side is responsible for preventing unauthorized access.

Once the user has been successfully logged in after the security checking, the user can check in the product information. This procedure is illustrated in Fig. 11. By specifying product related information, the 3D model can be uploaded to the server side. Then, on the server side, product manager will validate the model by different application view packages and finally check the information into the database.

If the user wants to check out the product information from the database, MDAI also provides database *search* and data *check out* functions. By specifying the search items, e.g. the product name, the *search* function will search in the database, after which the matching product with all its containing parts will be checked out. Finally, the user can download the file to the client side for further processing. This procedure is illustrated in Fig. 12.

7. Conclusions

In this paper, a new communication data structure, *operation*, has been defined to minimize the network information transfer load in collaborative engineering and to support nearly real time collaboration among multiple applications by incrementally transfer model modifications. The scenarios studied are based on the web-based, database-driven, and feature-oriented system architecture. The novelty of this research is the new mechanism via operations to address feature-level interoperability among multiple applications. The client–server architecture can provide shared access for multiple users. The proposed four-layer information model can integrate different applications with an associative fine-grain product repository, and allow the manipulation of application-specific information with sub-models. The generic feature model provides an integrated and associated feature modeling framework for different application feature models. With the proposed operation-based synchronization and scalable database support, product engineering processes can be organized for multiple applications with flexibility and finer-level of granularity. A geometrical modeler has been incorporated into the system to provide lower level geometrical modeling service, with which feature geometric modeling can be realized. Through some preliminary case studies, it can be concluded that the proposed operation synchronization mechanism could be very useful for a prospective global collaborative engineering framework with the aid of a fine-grain feature-oriented product database.

References

[1] Y. Li, Y. Lu, W. Liao, Z. Lin, Representation and share of part feature information in web-based parts library, Expert Systems with Applications 31 (4) (2006) 697–704.
[2] Industrial Automation Systems and Integration—Product Data Representation and Exchange—Part 1: Overview and Fundamental Principles, ISO 10303-1:1994 (E), ISO, Geneva, 1994.
[3] S. Szykman, R.D. Sriram, Design and implementation of the web-enabled nist design repository, ACM Transactions on Internet Technology 6 (1) (2006) 85–116.
[4] Y.P. Zhang, An Internet based STEP data exchange framework for virtual enterprises, Computers in Industry 41 (2000) 51–63.
[5] K. Rouibah, S. Ould-Ali, Dynamic data sharing and security in a collaborative product definition management system, Robotics and Computer-Integrated Manufacturing 23 (2) (2007) 217–233.
[6] C.H. Wang, S.Y. Chou, Entities' representation modes and their communication effects in collaborative design for SMEs, International Journal of Manufacturing Technology 37 (5–6) (2008) 455–470.
[7] W.D. Li, S.K. Ong, A.Y.C. Nee, Recognizing manufacturing features from a design-by-feature model, Computer-Aided Design 34 (2002) 849–868.
[8] A. Noort, G.F.M. Hoek, W.F. Bronsvoort, Integrating part and assembly modeling, Computer-Aided Design 34 (2002) 899–912.
[9] W.F. Bronsvoort, A. Noort, Multiple-view feature modeling for integral product development, Computer-Aided Design 36 (10) (2004) 929–946.
[10] Y.-S. Ma, T. Tong, Associative feature modeling for concurrent engineering integration, Computers in Industry 51 (2003) 51–71.
[11] Y.-S. Ma, G.A. Britton, S.B. Tor, L.Y. Jin, Associative assembly design features: concept, implementation and application, International Journal of Advanced Manufacturing Technology 32 (5–6) (2007) 434–444.
[12] G. Chen, Y.-S. Ma, G. Thimm, S.-H. Tang, Unified feature modeling scheme for the integration of CAD and CAx, Computer-Aided Design & Applications 1 (1–4) (2004) 595–602.
[13] G. Chen, Y.-S. Ma, G. Thimm, S.-H. Tang, Knowledge-based reasoning in a unified feature modeling scheme, Computer-Aided Design & Applications 2 (1–4) (2005) 173–182.
[14] Y. Shen, S.K. Ong, A.Y.C. Nee, Product information visualization and augmentation in collaborative design, Computer-Aided Design 40 (9) (2008) 963–974.
[15] R. Bidarra, W.F. Bronsvoort, Semantic feature modeling, Computer-Aided Design 32 (2000) 201–225.
[16] W.F. Bronsvoort, R. Bidarra, A. Noort, Semantic and multiple-view feature modeling: towards more meaningful product modeling, in: F. Kimura (Ed.), Geometric Modeling—Theoretical and Computational Basis towards Advanced CAD Applications, Kluwer Academic Publishers, Dordrecht, 2001, pp. 69–84.
[17] W.D. Li, S.K. Ong, J.Y.H. Fuh, Y.S. Wong, Y.Q. Lu, A.Y.C. Nee, Feature-based design in a distributed and collaborative environment, Computer-Aided Design 36 (2004) 775–797.
[18] J. Kim, S. Han, Encapsulation of geometric functions for ship structural CAD using a STEP database as native storage, Computer-Aided Design 35 (2003) 1161–1170.
[19] CAD/CAM-E website. http://www.cadcam-e.com/.
[20] OPENDXM overview. http://www.haitec.de/HAISite/haisite.nsf/ContentByKey/EMXH55LFC8-EN-P.
[21] G. Chryssolouris, D. Mavrikios, M. Pappas, A. Web, Virtual Reality Based Paradigm for Collaborative Management and Verification of Design Knowledge, Methods and Tools for Effective Knowledge Life-Cycle-Management, Springer, Berlin, 2008, pp. 91–105.
[22] C.J. Date, H. Darwen, Foundation for Future Database Systems: The Third Manifesto, Addison-Wesley, 2000.
[23] Y.-S. Ma, S.-H. Tang, G. Chen, A fine-grain and feature-oriented product database for collaborative engineering, in: W.D. Li, S.K. Ong, A.Y.C. Nee, C.A. McMahon (Eds.), Collaborative Product Design & Manufacturing Methodologies and Applications, Springer, England, 2007, pp. 109–136.
[24] ISO, Industrial Automation Systems and Integration—Product Data Representation and Exchange—Part 11: Description Methods: The EXPRESS Language Reference Manual, ISO 10303-11:1994 (E), Geneva, 1994.
[25] G. Chen, Y.-S. Ma, G. Thimm, S.-H. Tang, Associations in a unified feature modeling scheme, ASME Transactions: Journal of Computing & Information Science in Engineering 6 (2) (2006) 114–126.
[26] G. Chen, Y.-S. Ma, G. Thimm, Change propagation algorithm in a unified feature modeling scheme, Computers in Industry 59 (2–3) (2008) 110–118.
[27] J.-Y. Chen, Y.-S. Ma, C.L. Wang, C.K. Au, Collaborative design environment with multiple CAD systems, Computer-Aided Design & Applications 2 (1–4) (2005) 367–376.
[28] S.-H. Tang, Y.-S. Ma, G. Chen, A web-based collaborative feature modeling system framework, in: Proceedings of the 34th International MATADOR Conference, 2004, pp. 31–36.
[29] W. Shen, Q. Hao, S. Wang, Y. Li, H. Ghenniwa, An agent-based service-oriented integration architecture for collaborative intelligent manufacturing, Robotics and Computer-Integrated Manufacturing 23 (3) (2007) 315–325.
[30] ORACLE online documentation. Available at: http://www.oracle.com.
[31] X.F. Zha, R.D. Sriram, M.G. Fernandez, F. Mistree, Knowledge-intensive collaborative decision support for design processes: a hybrid decision support model and agent, Computers in Industry 59 (9) (2008) 905–922.

Dr. Y.-S. Ma has been an associate professor in the Dept. of Mechanical Engineering, University of Alberta, Canada since September 1, 2007. His main research areas include CADCAM, product lifecycle management, feature-based product and process modeling. Dr. Ma received his BEng degree from Tsing Hua University, Beijing (1986) and obtained both MSc and PhD degrees from UMIST, Manchester University, UK in 1990 and 1994 respectively. Before joining U of A, he had been an associate professor in the school of Mechanical and Aerospace Engineering, Nanyang Technological University (NTU), Singapore from 2000 for 7 years. He started his career as a lecturer at Ngee Ann Polytechnic, Singapore from 1993 to 1996. Between 1996 and 2000, he worked as a Senior Research Fellow and a Group Manager at Singapore Institute of Manufacturing Technology (SIMTech).

Dr. S.-H. Tang obtained his PhD degree from Nanyang Technological University (NTU), Singapore in 2007. He is currently an associate professor in School of Mechanical and Production Engineering, GuangDong University of Technology. His research interest includes web-based collaborative engineering, STEP standard applications, and mechanical design.

Dr. C.K. Au has been an assistant professor since 2001 at School of Aerospace and Mechanical Engineering, Nanyang Technological University (NTU), Singapore. His research interests are Geometric Modeling and Physically Based Modeling. Dr. Au received his B.Sc. (1985) and M.Sc. (1992) from University of Hong Kong and Ph.D. (1998) from Hong Kong University of Science and Technology all in Mechanical Engineering.

Mr. J.-Y. Chen graduated from the MEng programme of Nanyang Technological University (NTU), Singapore in 2006. Currently, he is a modeling and design engineer of STMicroelectronics Pte. Ltd. His research is concentrated in the field of CAD/CAM and tooling design.