Web-Enabled Collaborative Design Environment

Hyun Kim, Sang-Bong Yoo, and Hyun-Chan Lee

Recently, advanced information technologies have opened new possibilities for collaborative designs. In this paper, a web-enabled collaborative design environment is proposed, where the product data based on STandard for the Exchange of Product model data (STEP) is managed in a hierarchical database and the product metadata is used to efficiently search and utilize information scattered over the network. Several integrity constraints are depicted using EXPRESS to validate the combination of data from different sources. The knowledge represented as metadata and constraints on the interacting features differentiate this environment from previous ones. The collaborative conferencing system is also introduced to communicate and collaborate simultaneously among the related designers. As a result, the proposed environment allows the distributed designers to more efficiently obtain, exchange and communicate the design information throughout the design process.

Manuscript received January 18, 2000 ; revised July 12, 2000.

Hyun Kim is with the Concurrent Engineering Team, ETRI, Taejon, Korea. (phone: +82 42 860 5626, e-mail: hyunkim@etri.re.kr)

Sang-Bong Yoo is with the Department of Automation Engineering, Inha University, Inchon, Korea. (phone: +82 32 860 7386, e-mail: syoo@inha.ac.kr)

Hyun-Chan Lee is with the Department of Information and Industrial Engineering, Hongik University, Seoul, Korea. (phone: +82 2 320 1671, e-mail: hclee@wow.hongik.ac.kr)

I. INTRODUCTION

The product development environment will become increasingly global, network-centric and physically decentralized, which enables engineers to more effectively obtain, exchange and communicate a wide range of design information during product development. Therefore, the collaborative design has an important role in product development. Recently, advanced information technologies including Internet-related technology and distributed object technology have opened new possibilities for a collaborative design.

In order to support efficiently the collaborative design process under the advanced information infrastructure, various web-enabled tools should be integrated into the collaborative design environment. Though there are many issues associated with the collaborative design environment, we considered the following four major problems: 1) how to manage the product life-cycle information in a distributed environment; 2) how to search and utilize necessary information from vast amounts of data scattered over the network; 3) how to validate the combination of data from different sources; 4) how to support efficient communication for the cooperative design processes.

In order to effectively support the above-mentioned issues, we have employed the following approaches. Firstly, we used the STandard for the Exchange of Product model data (STEP) [1] standard to manage product information throughout the design process. Even though STEP is able to provide the integrated information model that pertains to product data, it does not accommodate the feature interaction essential for collaborative design. We proposed a scheme to represent the feature interaction by providing a mechanism that relates features to each other. Secondly, the product information based on STEP was managed in a hierarchical database, and the product metadata was used to efficiently search and utilize information scattered over the network. The metadata is managed by knowledge base that enables designers to search product data by general characteristics. Thirdly, we also

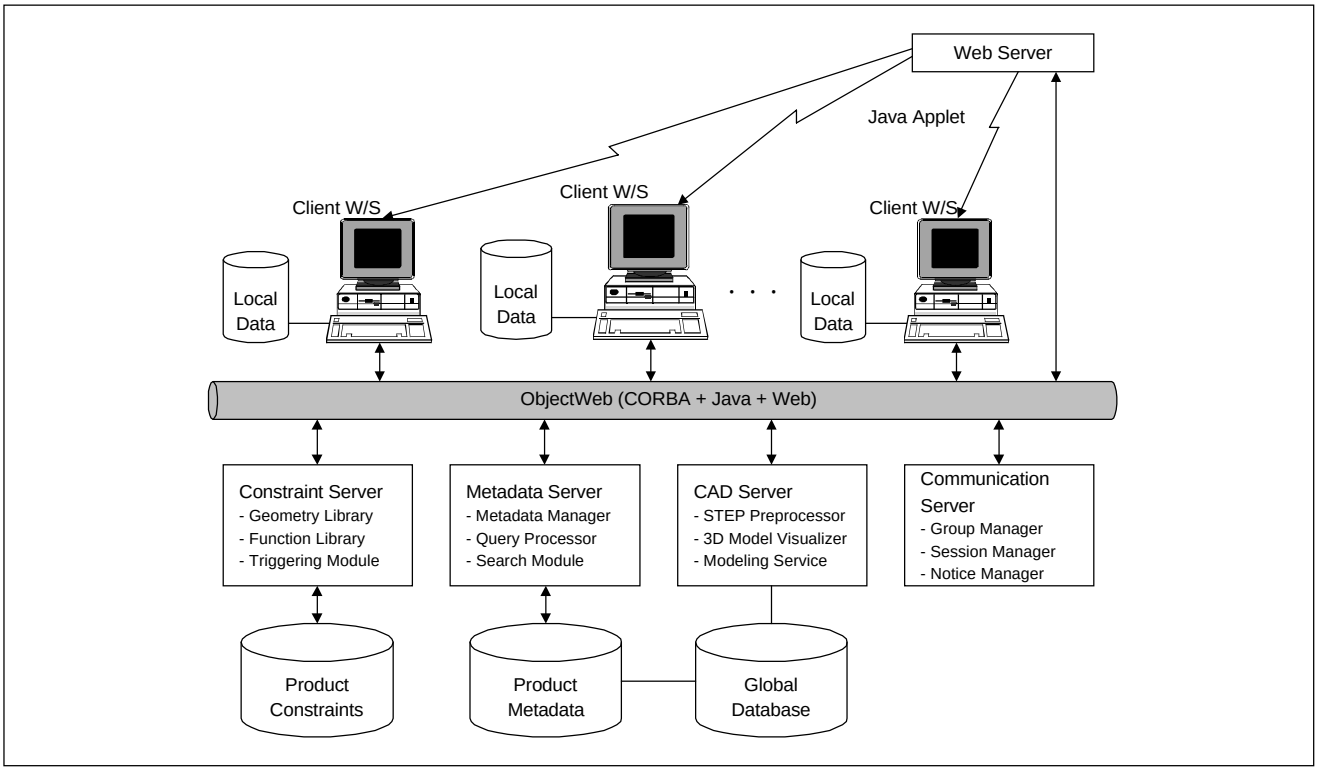

Fig. 1. The architecture of a collaborative design environment.

proposed the efficient validation approach of integrity constraints to validate the combination of data from different sources. Several integrity constraints were depicted and validated using EXPRESS [2], which is an object-oriented information modeling language developed for STEP. Fourthly, we proposed the collaborative conferencing system using STEP so that design participants can communicate and collaborate simultaneously with one another. This collaborative conferencing system enables multiple designers to share the same design object (such as 3D geometric model) and make a conversation using text-chatting and mark-up functionalities.

Figure 1 shows the system architecture for a web-enabled collaborative design environment. The Web-based client/server architecture is a three-tier structure including multiple clients, application servers, and databases. The communication between clients and application servers is done via a standard communication protocol-CORBA provided by the Object Management Group (OMG) for interoperability among the distributed objects. The basic functionalities of the three tiers are as follows:

1) Databases: This tier runs on high-performance workstations and is composed of the global database, the product constraints and the metadata.
   - Global database: This serves as the repository for information to be shared among the designers throughout the design life cycle. Each client designates a local database in which the working data and models are stored. Once a design process is complete, the working data are then moved to a global database and flagged as released data.
   - Product Constraints: The constraints are defined in the product model in EXPRESS. Constraints are either local, specific to a separate part, or global, relating to two or more parts.
   - Product Metadata: Product data developed by each designer are kept on a local machine. Only the metadata for the product are registered into the knowledge base to avoid exchanging all of the product data. The product metadata include the product name, designer's name, description, material, location, and other information, and are interfaced and managed by the Metadata Server.
2) Application servers: This tier consists of a Web server and several CORBA objects, including a Metadata server, a Constraint Server, a CAD server and a Communication server.
   - Metadata Server: It helps clients to access any installed global databases, and also acts as a client to the global database. Based on the stored metadata, it provides various search facilities and utilities such as registering and viewing searched data. Users can search the metadata to retrieve information about interesting product data.
   - Constraint Server: The integrity constraints defined in EXPRESS are validated by the Constraint Server, which manages the integrity of the product data. If two parts from different sources are combined into an assembly, and con-

straints are defined for this combination, the constraint validation module is triggered.

- CAD Server: It supports the visualization and manipulation of geometric models. It pre-processes STEP-based CAD data, and sends the simplified model to clients so they can view and manipulate it on the Web.
- Communication Server: It supports the ability of the design participants to communicate and collaborate simultaneously with one another. It also manages the design participants and communication sessions, and handles event messages from the session participants.

3) Clients: This tier provides a cross-platform end-user interface to the system. Though multiple clients have their own local database, they can share product models in the global database through the Web browser such as Netscape Navigator or Microsoft Internet Explorer. The metadata are encoded in XML and the geometric models are visualized by using the Java3D graphic library.

Using this environment multiple designers are able to share design information, check constraints and communicate simultaneously. The organization of this paper is as follows: Section II reviews related works. Section III explains the information models for interacting features, which are important because many design and analysis activities (e.g., part assembly and process planning) could depend on it. Section IV discusses the metadata of the product data and their applications for Web interfaces. Section V presents the methodology of the integrity validation for new designs and assemblies. Section VI discusses the collaborative design work through CAD conferencing and case examples. Section VII concludes the paper and discusses some future work.

II. RELATED WORK

Active studies have been performed recently in order to implement the collaborative design environment. They can be classified into five categories: 1) CAD conferencing, 2) work process modeling and management, 3) product data sharing, 4) agent-based knowledge sharing and 5) conflict management.

1. CAD Conferencing

It supports synchronous collaborative works by exchanging geometric models using a teleconferencing system under the networked environment. These researches focus on application sharing, co-authoring, visualizing three-dimensional geometry and desktop conferencing [3]–[5].

2. Work Process Modeling and Management

It supports asynchronous collaboration among work groups by modeling and managing the design workflow to handle a complex design procedure efficiently. Kim *et al.* [6] proposed the framework for process-centric collaborative design. Lavana *et al.* [7] introduced a directed hypergraph model for executable design workflows. Wallace *et al.* [8] introduced the object-based modeling of design problems in a distributed collaborative environment.

3. Product Data Sharing

For collaborative design, it is important to exchange and share the product data efficiently through the product life-cycle. STEP [9] and other related works have been performed on data exchange between different computer-aided systems. Hardwick *et al*. [10] proposed an information infrastructure architecture that enhances collaboration in a virtual manufacturing enterprise using STEP. Jasnoch *et al*. [11] used STEP-based models and a commercial object-oriented database system to share product data and support collaboration.

4. Agent-Based Knowledge Sharing

Research on agent-based knowledge sharing focuses on enabling collaboration among software agents, usually using a predefined language such as Agent Communication Language (ACL). In the SHARE project [12], Knowledge Query Manipulation Language (KQML) was used to help the design teams share their understanding of the design process. Case *et al*. [13] used Virtual Workspace Language (VWL) for communication and collaboration among designers.

5. Conflict Management

Conflict management is needed to coordinate information for collaborative design. The coordination strategies that avoid conflicts between the design participants have been proposed at the data level or at the process level [14], [15]. Klein proposed a hierarchy of conflict types and specific strategies for resolving those conflicts [16]. Gupta et al. proposed the constraint-based coordination strategies to allow for conflicts and support their resolution [17].

Validation of the integrity constraints via EXPRESS for the STEP data (stored in file systems or DBMS systems) has been carried out by a few researchers. Because integrity validation processes are often time-consuming and degrade the performance of the overall system, most of the previous studies concentrated on performance optimization. Mueller *et al*. [18] used multiprocessing, Yoo and Cha [19] proposed data dependency analysis, and Alt [20] used view materialization for optimization.

There have also been many studies that combine two or more of the technologies classified above [1], [10], [21]. Because designers interact with each other for several different purposes

in the process of collaborative design, a single technology alone is insufficient for collaborative design environments.

The environment proposed in this paper supports CAD conferencing, product data sharing, and conflict management. Even though we do not use agent technologies, such as ACL or KQML, the knowledge represented as metadata and constraints on the interacting features differentiate this environment from previous ones. The other main feature of this paper is that all the interfaces are designed for the Web such that user interfaces and other modules are built as agents that can be downloaded from web pages. The metadata are modeled in Resource Description Format (RDF) [22] and are encoded in XML. Java3D API is used to visualize and manipulate geometric models on the Web. Employing these new W3C standards enables the proposed approaches to be applied for many related applications in the future.

III. FEATURE INTERACTION

Feature modeling has become an essential ingredient in a collaborative design because features are the basic elements in part assembly, design analysis, and process planning. In this reason, features are well recognized in international standard activities. International Organization for Standardization (ISO) already provided an international standard on features [23]. ISO 10303 is an international standard for STEP, where Application Protocol (AP) 224 is "Application protocol: Mechanical product definition for process planning using machining features." The AP defines various machining features, but they are mostly single features.

In the process planning stage, single features provide information on how to manufacture the part. For a part with a simple shape, the information provided by a single feature is sufficient for process planning. For a part with a more complex shape, however, handling of interaction among the features is required. If the interaction is ignored, process planning of the parts becomes inefficient. Unfortunately, STEP AP 224 does not provide the schemes for feature interaction. This problem is also applied to other cases like part assembly.

In this paper, a scheme to represent the feature interaction by providing a mechanism that relates features to each other is proposed. The modeling mechanism of feature interaction is based on the entity explicit_geometric_constraint proposed by Parametrics group, which is included in the STEP Working Group (WG) 12 [24]. By applying the explicit geometric constraints on the features, we can model how the features are related to each other.

The basic features are defined in STEP AP 224 [23] as either single or repeated, where the repeated features are a replica of the same features in a systematic fashion. However, no scheme for the feature interaction is provided in the AP. In this paper, the basic feature definition of the AP is adopted as follows:

```
ENTITY feature_definition
ABSTRACT SUPERTYPE OF (ONE OF (boss, marking,
     flat_face, outer_bound, outside_profile,
     protrusion, pocket, removal_volume, round
     _hole, rounded_end, revolved_profile, slot,
     spherical_cap, step, thread, turned_knurl,
     externally_defined_feature_definition));
SUBTYPE OF (characterized_object);
END_ENTITY;
```

The Parametrics group of STEP has worked on the representation of the parametric design, such that support of the parametric design is achieved by adopting a variational design process or history-based design process. If the history-based design process is used, feature interaction can be modeled very easily. However, the history-based design process is not yet well defined and each commercial CAD system has its own scheme. Therefore, the Parametrics group adopted the variational design process through explicit geometric constraints. To represent the feature interaction, explicit geometric constraints on the related features are applied. The initially proposed scheme for the entity explicit_geometric_constraint by the Parametrics group is given in the draft of STEP Part 108 [24] as follows:

```
ENTITY explicit_geometric_constraint
ABSTRACT SUPERTYPE OF (ONE OF(parallel_constraint,
   point_distance_constraint, radius_constraint,
   curve_length_constraint, angle_constraint,
   direction_constraint, perpendicular_constraint,
   incidence_ constraint, coaxial_constraint,
   tangent_constraint,  symmetry_constraint,
   offset_constraint));
SUBTYPE OF (predefined_constraint);
END_ENTITY;
```

To describe the feature interaction simply, we restricted the interaction to a relationship between two features. If interactions exist among more than two features, the interaction can be defined repeatedly by pair-wise relationships. To accommodate the interaction in process planning and assembly when there are two features interacting, the relationship between the two features is represented using the explicit_geometric_constraint. This interaction is defined in three types: combined feature, intersecting feature, and mating feature. The following entity definition is for the interacting_feature.

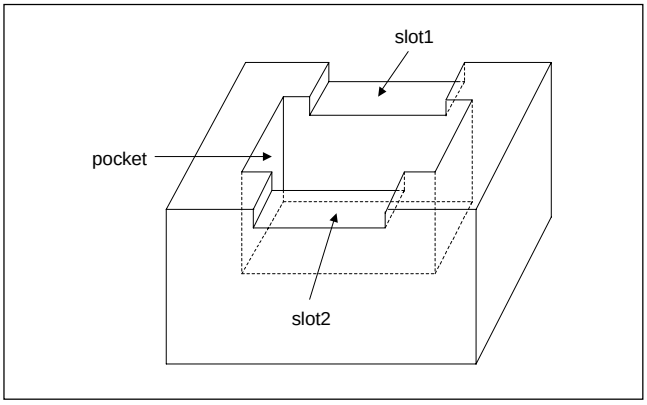

Fig. 2. Examples of a combined_feature and an intersecting_feature.

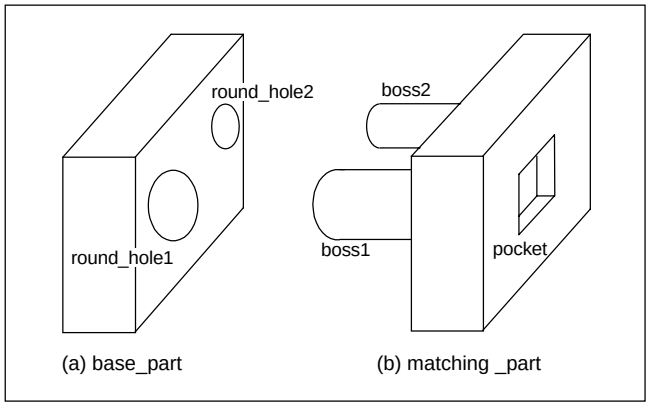

Fig. 3. An example of the mating_feature.

```
ENTITY interacting_feature
ABSTRACT SUPERTYPE OF (ONE OF(combined_feature,
    intersecting_feature, mating_feature));
END_ENTITY;
```

Other interacting features exist beyond the three proposed in this study, but their definitions are based on the applications used. For the purposes of this study, we have suggested a scheme for representing only fundamental interacting features. Definitions are provided below.

1. Combined Features

If one feature is divided by another one into two separate features, they must be recombined for the process planning. Otherwise, they must be treated as separate features, which increases the set-up and process time in the manufacturing processes. One example of a combined feature is given in Fig. 2, where slot1 and slot2 are separated by a pocket and should be combined into a single feature (slot). A combined_feature enables the designer to assign constraints for combining two features into one. The EXPRESS description of the combined_feature is as follows:

```
ENTITY combined_feature
SUBTYPE OF (interacting_feature);
    id           : identifier;
    name         : label;
    description : text;
    relating_features : SET[2:2] of feature_
                        definition;
    constraints :  SET[1:?]  of  explicit_
                   geometric_constraint;
END_ENTITY;
```

The constraints for the combined_feature are used to represent the coplanar attributes of the two faces from each feature. The most frequently used constraints include the coaxial_ constraint, parallel_constraint, or point_distance_constraint.

2. Intersecting Features

The combining of two features into one involves the intersection of the two features, the processing sequence of which should be determined very carefully. Therefore, the intersecting conditions are modeled using the geometric constraints. One example of the intersecting_feature is given in Fig. 2, where the pocket and combined_feature (slot1, slot2) are intersecting. The EXPRESS description of the intersecting_feature is as follows:

```
ENTITY intersecting_feature
SUBTYPE OF (interacting_feature);
    id           : identifier;
    name         : label;
    description : text;
    relating_features : SET[2:2] of feature_
                        definition;
    constraints :  SET[1:?]  of  explicit_
                   geometric_constraint;
END_ENTITY;
```

The perpendicular_constraint and angle_constraint are frequently used constraints for representing the intersection angle, and point_distance_constraint is used for representing the distance from the boundary edge to the intersection point.

3. Mating Features

When two parts are assembled, the assembling conditions between the two features from the separated parts should be defined. The mating_feature serves this purpose. An example of the mating_feature is given in Fig. 3, where (hole1, boss1) and (hole2, boss2) are mating_features. The EXPRESS description of the mating_feature is as follows:

```
ENTITY mating_feature
SUBTYPE OF (interacting_feature);
    id          : identifier;
    name        : label;
    description : text;
    relating_features : SET[2:2] of feature_
                        definition;
    constraints :  SET[1:?]  of  explicit_
                   geometric_constraint;
END_ENTITY;
```

Frequently used constraints for the mating_feature are the parallel_constraint, perpendicular_constraint, coaxial_constraint, and direction_constraint.

The feature interaction has been summarized in EXPRESS-G (Fig. 4), which includes the interacting_feature, feature_definition, and explicit_geometric_constraint. In this Fig., rectangles, solid lines, and thick solid lines represent the entity types, attributes, and subtypes, respectively. Combining the feature definitions and explicit geometric constraints allows us to model the interacting features.

IV. SHARING DESIGN INFORMATION THROUGH KNOWLEDGE BASE

In the proposed collaborative design environment, the metadata for the products are managed by the knowledge base. When a designer needs related design data, he/she views and searches the knowledge base. The knowledge base enables the user to perform a content-search for the entire product data, where the user can search the product data by not only product ID, registration No., or other predefined key words, but also general characteristics such as product name, description, material, features and so on. In this section, the structure of the metadata and the operations of the knowledge base are explained.

1. Metadata of Product Data

The metadata are the major part of the knowledge base, but there is no standardized content of the metadata because it is application specific. The first step of building a knowledge base is to determine the content of the metadata. In this paper, we assume that the product data are represented in STEP (especially in Application Protocol 203[25]—configuration-controlled design). The metadata are classified into the following six categories:

- Design: Information about STEP physical files. It includes the header information of the files and the information provided by the users when they register the files.
- Registry: Information about the registrant. It includes ID, name, email address, and registry date.
- Part: Information about the parts included in STEP files. Because

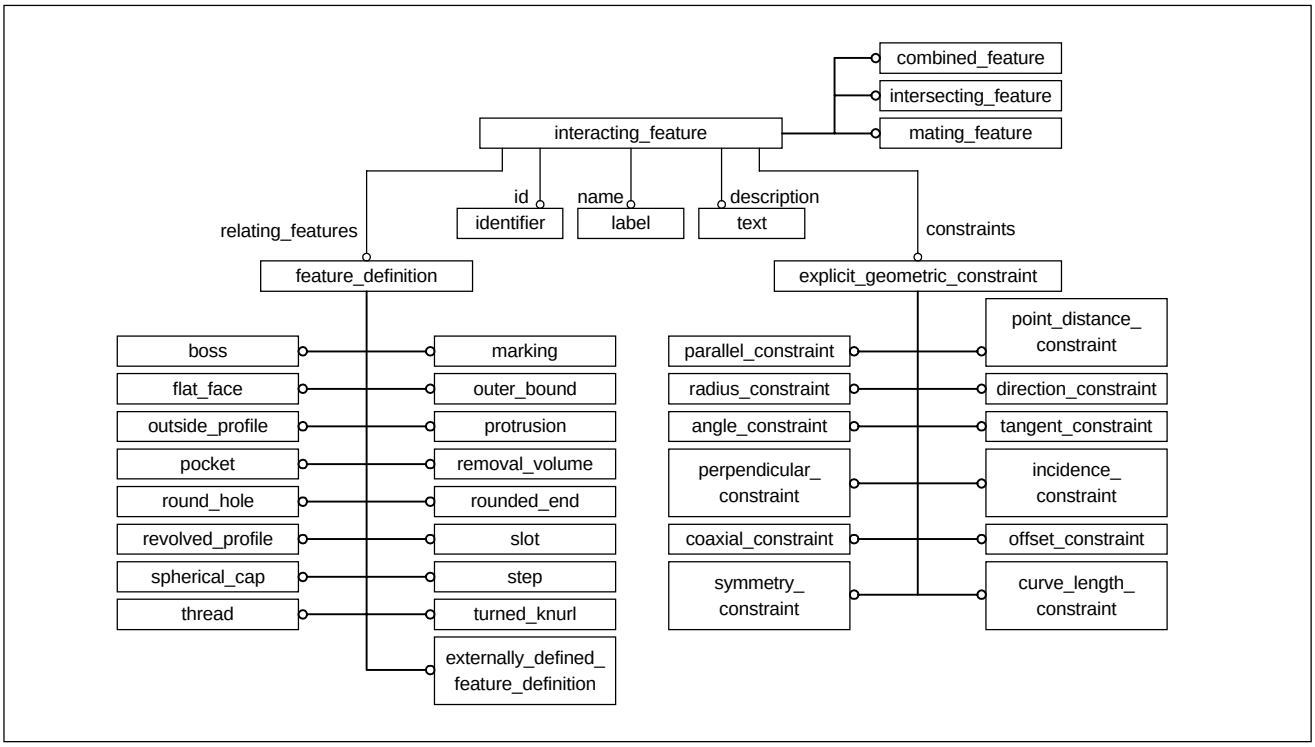

Fig. 4. EXPRESS-G for the interacting_feature.

Table 1. Categories and elements of the metadata.

Category	Element Name	Description	Category	Element Name	Description
Design	Filename	File name	Part	partID	Part ID
	FileDesc	File description		partName	Part Name
	Preprocessor	Preprocessor that generate the file		partDesc	Description
	SchemaName	Schema name		level	Level of part structure
	url	URL location		quantity	Quantity
	FileSize	Size of the file		associatedDoc	Related document
Registry	RegistrarID	Registrar ID		associatedPerson	Related person
	RegistrarName	Name of Registrar		containedBy	Part that contain this part
	registrarEmail	Email address		contains	Parts that this part contains
	RegistryDate	Registry date		approvalInfo	Approval Information
Document	DocID	Document ID			
	DocName	Document Name	Approval	approvalStatus	Approval status for the part
	DocDesc	Description			
	DocType	Document Type		approvedBy	Person who approves
Person	PersonID	Person ID			
	PersonName	Person Name		approvalType	Type of the approval
	Employer	Employer			
	PersonRole	Role		approvalDate	Approval Date

STEP 203 defines the product structure, several parts can be included in a STEP file.

- Document: Information about documents such as ID, name, description, and type.
- Person: Information about related personnel such as ID, name, employer, and role.
- Approval: Information about (the) approval for parts such as status, approver, role, and date.

A complete list of the metadata is summarized in Table 1, where the category names are used in the "Category Search." The metadata defined in Table 1 were modeled by RDF and extracted from STEP files. The RDF model of the metadata is not discussed in this paper due to the space limitations.

2. Implementation of Knowledge Base System

The knowledge base system in the metadata server manages the metadata as explained in Section IV-1. The structure of the knowledge base system is depicted in Fig. 5. The user interface module is implemented as JAVA agents and it communicates with the knowledge base system through CORBA ORB. The ORB interface enables more dynamic user interfaces on heterogeneous JAVA virtual machines. Because conventional CGI operations should be coupled with HTML forms, they are less flexible and require tedious updates for any changes in the user interfaces.

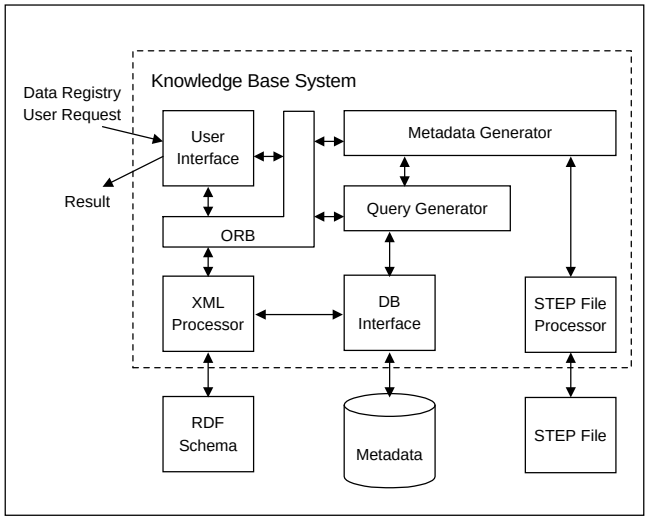

Fig. 5. Structure of Knowledge Base System.

The metadata are stored and managed by a relational database system and can be stored as an XML file. A database system is used because it provides fast index structures and other data management amenities such as concurrency control, crash recover, and query processing. As the size of the metadata increases, the performance of the XML file processing degrades proportionally. XML files are also used for returning the results to the users.

The metadata are generated as the users register design data or documents (see Fig. 6 where the location of a STEP is typed).

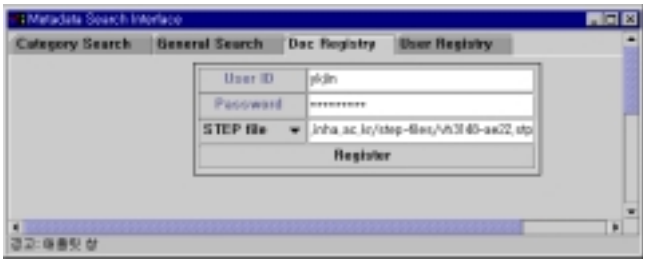

Fig. 6. User interface for data registration.

As explained in the previous subsection, major parts of the metadata are extracted from STEP files automatically. Because the metadata schema is designed based on STEP AP 203, it is not difficult to extract metadata from the given STEP files. The metadata generator in Fig. 5 interacts with the STEP file processor to extract the metadata and DB interface for storage. We used the SCL 3.1 from NIST [29] for the STEP file processor and the SQL Server from Microsoft for the relational database system.

A data search can be performed as a general search, category search, or XQL pattern. For a general search, a keyword(s) is used to find all the metadata items that contain it. The category search is based on the metadata schema as defined in Table 1. An example of the user interface of the knowledge base is depicted in Fig. 10 in Section VI.

V. INTEGRITY VALIDATION FOR PRODUCT DATA

In this section, we discuss the validation method of the integrity constraints defined for the product data introduced in Section III. In this collaborative design environment, registered parts designed by other people can be easily searched and reused. Because it is common practice to fail to notice some design flaws in those imported parts, validating the integrity constraints is very important for collaborative designs.

1. Integrity Constraints in EXPRESS

EXPRESS is very powerful for defining integrity constraints, which can include general procedures. In EXPRESS, there are four types of integrity constraints: uniqueness constraints, local constraints, existence constraints, and global rules. Beyond these basic types, constraints can also be defined by Boolean expressions or as actual entities. Each type of integrity constraint is discussed briefly in this subsection.

A. Uniqueness Constraints

UNIQUE clauses declare the uniqueness constraints on a single attribute or a set of multiple attributes of an entity type. If a UNIQUE is defined on a single attribute, no two or more entity instances can have the same value for that attribute. A UNIQUE definition on a set of multiple attributes restricts the same combination of values for the named attributes. Validation of the uniqueness constraints can be done in a local database. If the instances for the same entity are generated by other application systems, it should also be validated in the global database.

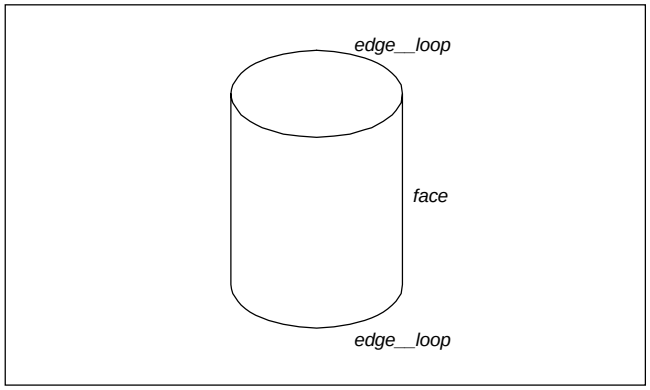

Fig. 7. The conceptual diagram of the entity *round_hole*.

As an example, the entity *part_management_information* contains general management information, such as identifier, name, description, and the quantities included in an assembly. The following entity definition has a UNIQUE clause, where *ur1* is a label and *id* declares the attribute to be unique, such that all the instances for the entity *part_management_information* can not have the same identifier.

```
ENTITY part_management_information;
   id   : identifier;
   name : label;
   description : text;
   quantities_included_in_superitem: INTEGER;
UNIQUE
   ur1  : id;
END_ENTITY;
```

B. Local Constraints

WHERE clauses define the local constraints that apply to each instance of an entity type. The local constraints specify the valid values of an attribute or a combination of multiple attributes. Because this constraint applies to a single instance, every local constraint can be validated in local databases.

The WHERE clause in the following example specifies that the number of form feature elements of a *round_hole* should be three (two *edge_loop*s and one *face)* as shown in Fig. 7.

```
ENTITY round_hole;
SUBTYPE OF (feature_definition);
WHERE
    wr1 : SIZEOF (SELF.ff_elements) = 3;
```

```
    wr2 : SIZEOF (QUERY(temp <* SELF.ff_
          elements | TYPEOF
          (temp) = ['face'])) = 1;
END_ENTITY;
```

C. Existence Constraints

The question here is whether the existence of an instance of one entity is dependent upon the existence of an instance of another related entity. If another entity has established a relationship with the current entity by way of an explicit attribute, an inverse attribute may be used to describe that relationship in the context of the current entity. Validation of the existence constraints depends on whether or not the related two entities are defined in the same local schema. If the two entities are defined in the same local schema, this constraint can be validated in the database. Otherwise, it should be validated in the global database.

The following example includes two entities, *bolt* and *nut.* The entity *bolt* has *fastener* as an attribute, which specifies the matching nut. On the other hand, the entity *nut* has an attribute *fasten,* which specifies the matching bolt. In order to properly maintain the integrity of the database, both members of the matching pair should exist in the database.

```
ENTITY bolt;
SUBTYPE OF (part);
    size: bolt_size_type;
    type: bolt_type;
    fastener: nut;
END_ENTITY;
ENTITY nut;
SUBTYPE OF (part);
    size: nut_size_type;
    type: nut_type;
INVERSE
    fasten: bolt FOR fastener;
END_ENTITY;
```

D. Global Rule

The global rules can be defined by EXPRESS RULEs, which specify the constraints among a set of instances of an entity type or of multiple entity types. A RULE consists of executable statements and a WHERE clause determines the validity of the data based on results of the executables. A RULE declaration includes the list of entities that are referenced by the rule. The validation of a RULE depends on where the referenced entities are declared. If they are all declared in one local schema, it can be validated in the corresponding local database. Otherwise, it should be validated in the global database.

As an example, Fig. 3 shows two parts, a *base_part* and a *matching_part*. In order to be successfully assembled, the holes in the *base_part* and the bosses in the *matching_part* should fit each other. The *rule1* in this example specifies that the minimum size of the *round_hole1* in the *base_part* should be greater than the maximum size of the *boss1* in the *matching_part*. This constraint is specified for three entities, *base_part, matching_part,* and *assembly.* This constraint is validated for the pair of parts (*base_part* and *matching_part*), which are defined to be assembled together by the entity *assembly.*

```
RULE rule1 FOR (base_part, matching_part,
assembly);
LOCAL
    part1: ARRAY OF base_part;
    part2: ARRAY OF matching_part;
    h1, h2: ARRAY OF round_hole;
    b1, b2: ARRAY OF boss;
END_LOCAL;
part1 = QUERY (temp <* assembly | TYPEOF
                (temp.component_parts) =
                ['base_part']);
part2 = QUERY (temp <* assembly | TYPEOF
(temp.component_parts) = ['matching_part']);
IF (EXISTS(part1)) THEN
   h1 = QUERY (temp <* part1.part_features |
   temp.ff_id = ['hole1']);
   h2 = QUERY (temp <* part1.part_features |
   temp.ff_id = ['hole2']);
END_IF;
IF (EXISTS(part2)) THEN
   b1 = QUERY (temp <* part2.part_features |
   temp.ff_id = ['boss1']);
   b2 = QUERY (temp <* part2.part_features |
   temp.ff_id = ['boss2']);
END_IF;
WHERE
wr1: IF (EXISTS(part1) AND EXISTS(part2))
THEN
   ((h1.ff_rep.profile.diameter –
    h1.ff_tolerance.range.lower_bound) >
   (b1.ff_rep.profile.diameter +
    b1.ff_tolerance.range.upper_bound))
  AND
     ((h2.ff_rep.profile.diameter –
      h2.ff_tolerance.range.lower_bound) >
     (b2.ff_rep.profile.diameter +
      b2.ff_tolerance.range.upper_bound))
   END_IF;
END_RULE;
```

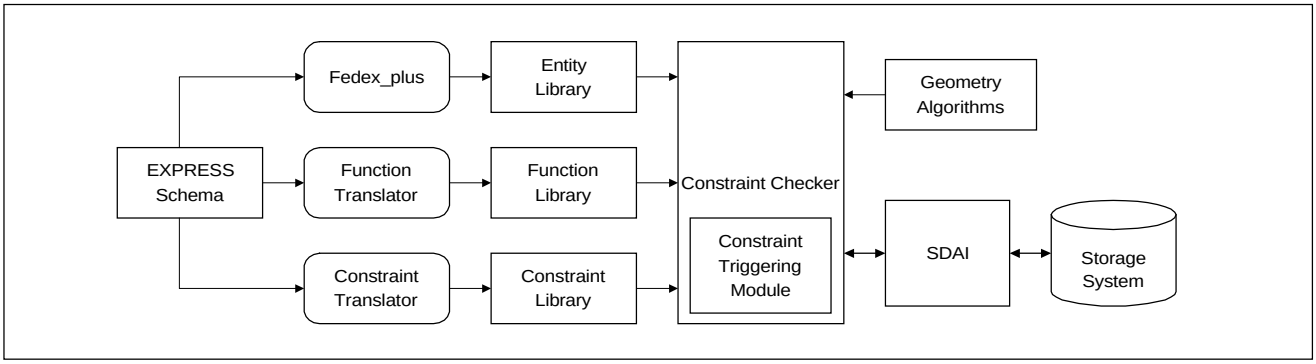

Fig. 8. The relationship among software modules for integrity validation.

E. Constraints Defined by Boolean Expressions

The characteristics of attribute values or the relationship among attribute values can be expressed as a Boolean expression. These Boolean expressions need to be translated into executable code and evaluated at run-time.

One example of a Boolean expression is the entity *free_form_constraint* that is defined in STEP part 108 as follows:

```
ENTITY free_form_constraint;
       SUBTYPE OF (explicit_constraint)
       constrained_elements: SET[1:?] OF
       value_assignable_expression;
       reference_element: OPTIONAL SET[1:?]
       OF value_assignable_expression;
       constraining_expression: boolean_
       expression;
END_ENTITY;
```

An instance of the Entity *free_form_constraint* in STEP file could be as follows:

```
#52 = FREE_FORM_CONSTRAINT ('ffc1', ('x','y'),
     ('z'), 'x + y < z.radius + 3')
```

In this example, the expression 'x + y < z.radius + 3' should be evaluated true when the three element x, y and z are instantiated.

F. Constraints Defined by Entities

The characteristics of some constraints are intrinsic to the definitions of the entities, e.g., parallel, perpendicular, incidence, coaxial, and symmetry. In these cases, some geometric algorithms to check each constraint are required. The following entity specifies that the members of a set of two or more lines or planes are mutually parallel.

```
ENTITY parallel_constraint;
       SUBTYPE OF (explicit_geometric_cons-
       traint);
       constrained_elements: SET[1:?] OF
       linear_geometry_element;
       reference_element: OPTIONAL linear_
       geometry_element;
       sense_check: BOOLEAN;
       side_check: BOOLEAN;
END_ENTITY;
```

2. Validation of Integrity Constraints

As discussed in the previous subsection, there are six types of integrity constraints in EXPRESS. Because each type of constraint has a different meaning and usage, the overall validation system should be carefully designed in order to maintain the consistency and efficiency of the overall system.

In order to validate the integrity constraints defined in an EXPRESS schema, the constraints should be translated into the appropriate forms. Figure 8 shows the relationship among the software modules related to validating the integrity constraints. Firstly, all the functions included in the constraints should be translated into a programming language (C++ is used in the prototype). The translated functions are then compiled and built into the function library. Secondly, the six types of integrity constraints also need to be translated into a programming language to build the constraint library. Thirdly, the entity structure and constraints defined in the schema are analyzed and compiled into the constraint-triggering module. The constraint checker, which is linked to the libraries and the constraint-triggering module, accesses the product data in the storage system and validates them.

In EXPRESS, there are built-in functions and user defined functions. For built-in functions like *sin( ), cos( ),* and *sizeof( )*, either C++ library functions or user defined C++ functions are used. User defined EXPRESS functions are translated into C++ functions. Because the basic structure of EXPRESS is similar to that of C++, and the keywords of EXPRESS have corresponding keywords in C++, the function translation is re-

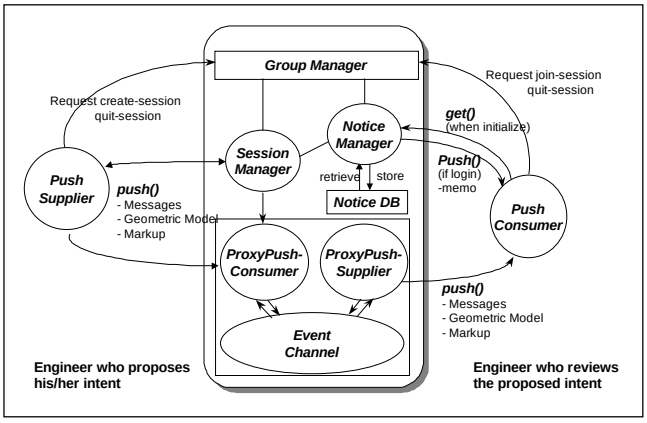

Fig. 9. Structure for the communication server.

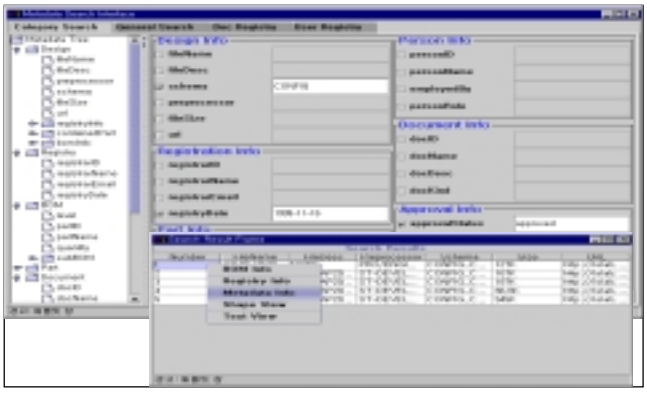

Fig. 10. User interface for a product data search.

latively simple. One difference between the two languages is the nested function, where EXPRESS allows functions to be defined within another function, but C++ does not. In order to handle this difference, nested EXPRESS functions are translated into separate functions in C++, such that each integrity constraint is translated into an executable module in C++. These executable modules access the product data in the storage system using Standard Data Access Interface (SDAI). An example of integrity validation between two parts to be assembled is shown in Fig. 11 in Section VI.

VI. COLLABORATIVE CONFERENCING

A collaborative design environment requires both synchronous collaboration as well as asynchronous collaboration (as discussed in previous sections). When design conflicts occur, or when the designers need to exchange their design intents, they should be able to share a geometric model in a same view and make a conversation in a synchronous fashion.

This collaborative conferencing is supported by the Communication Server in Fig. 1. The communication server (structure shown in Fig. 9) was implemented using CORBA Event Service, which is a facility defined by CORBA Services specification.

The communication server consists of a group manger, a session manager, a notice manager and an event channel. The group manager creates/deletes communication sessions, and allows participants to take part in the sessions. The session manager supports synchronous communication with the text-based chatting and the markup. The notice manager supports non-simultaneous communication via the memo or e-mail through the process of notification and acceptance. The event channel manages the transport of events between multiple suppliers and multiple consumers. As part of the CORBA Event Service, there are push suppliers that produce events intended for consumers, and push consumers that process events provided by suppliers. The push suppliers push events into the event channel, which then pushes those events into the push consumers. In the collaborative design environment, a push supplier is an engineer who proposes his/her own engineering intent, and a push consumer is an engineer who reviews the proposed engineering intent. The data transferred by the push suppliers and the push consumers are text-messages, markup objects and geometric models.

A shared geometric model can be visualized over the network using the CAD server (Fig. 1). STEP was used to represent a product model and to eliminate the data translation for communication. To this end, we compiled the AP203 schema and generated ROSE files and entity classes using the EXPRESS compiler of the ST-Developer [26]. The ROSE library was used to implement the procedure for retrieving data from physical files, and constructing objects of classes transformed from entities. Although STEP data makes it possible to represent a solid model completely, there is too much information for simply visualizing on the network. Therefore, the STEP data are converted to the simplified B-Rep model, in which topology data are removed to minimize data. CAD server sends the converted model to clients through ORB. Note that the complete solid model is managed only on the server-side, not on the client side. Each client can change his/her own model, but the complete model in the CAD server is not modified without model change event via communication server. Transformations such as translation, rotation and scaling are also handled in client side in order to reduce the load on the CAD server.

The following steps may be an example of the procedure of collaborative works in this system. A user carries out collaborative works on a Web browser by downloading a Java applet. He/she can load the existing model from the global database by searching the metadata in the Metadata Server. In the left window of Fig. 10, we can see the overall metadata view. When the user performs category search, one or more metadata categories are selected, metadata element is selected, and keywords are typed in. The user can select a particular design item from the search results and perform further operations. BOM information, registry

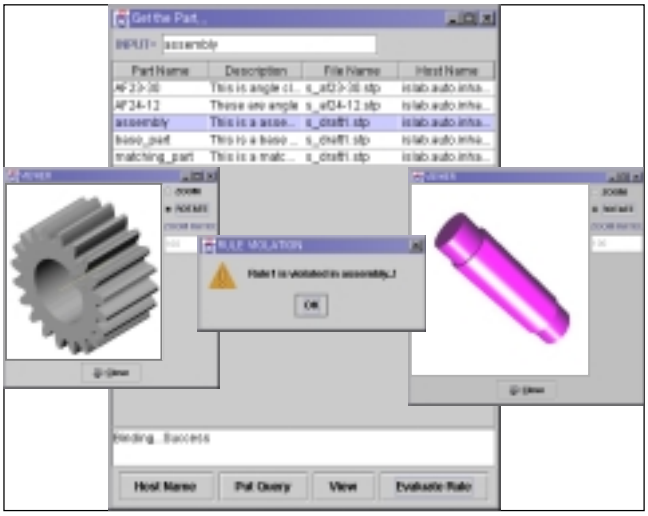

Fig. 11. An example of interactive rule validation.

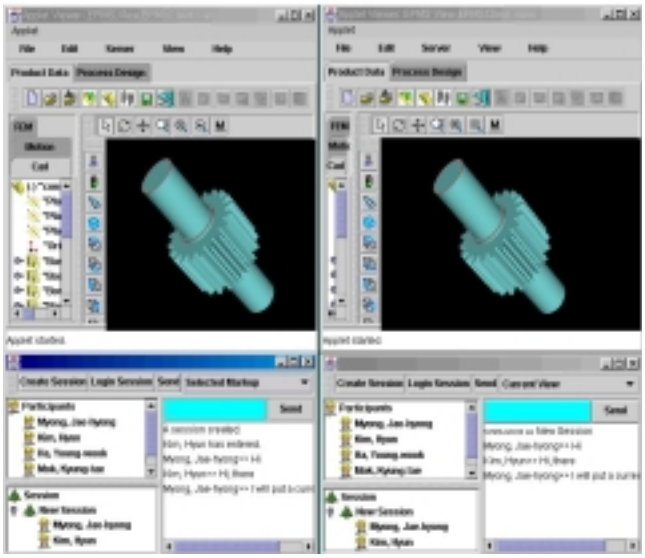

Fig. 12. Client-side browser for collaborative works.

information, metadata information, shape viewing, and text viewing are available for each selected item.

When the user tries to assemble with several parts selected out of the metadata server, the constraint checking is carried out by rule validation. The user interface in Fig. 11 shows that an instance of an assembly includes a gear_part and a shaft_part. The user can view them by pressing the view button, which initiates the transmission of the STEP data for the gear_part and a shaft_part to the CAD server. The CAD Server converts them to the simplified model and sends them to the user. In this example, a warning message is issued because the dimensions of a hole in the gear_part are not within the tolerance range defined by the rule. More specially, the minimum size of the *round_hole* in the *gear_part* should be greater than the maximum size of the *cylinder* in the *shaft_part*.

At this point in the example, the user (a gear designer) must communicate with a shaft designer, so he/she creates a communication session where they can discuss their opinions and solve any problems in a synchronous fashion. At this time, they share the same geometric model and make a conversation using text-chatting and markup functionality. Figure 12 shows a client-side browser to share the geometric model and communicate with the related engineers.

Once the designers find a solution, the models can be saved to the global database so that others can share it. At that time, the constraint server also checks the integrity constraints. If the integrity is validated, some data are added to metadata via the metadata server, and the STEP file is saved to the global database.

VII. CONCLUSION

In this paper, we have presented the design and implementation of a collaborative environment based on standard product data (STEP). One design objective of this environment was to provide web interfaces to users. Because several different activities are involved in collaborative designs, the ubiquitous web browsers are used for user interfaces. The other objective of this research was to separate knowledge from design data. The interacting features, constraints, and metadata are defined and managed by a global knowledge base, whereas the design data are managed by each designer in local databases. This approach distributes traffic over the networks and users can access the most updated product data directly from the local databases. We also introduced the collaborative conferencing system that multiple designers can share the same design object and make a conversation using text-chatting and markup functionalities.

Current implementation of the proposed environment does not support work process modeling, which is applicable to fixed and complex design processes. Even though it is not difficult to include conventional workflow modules to this environment, a new paradigm to deal with dynamic workflow should be devised. As the technologies and procedures involved in a collaborative design change often, workflow engines should adapt to new situations. Another possible extension of this work is to support agent-based knowledge sharing which could be a viable approach to applying agent technologies to handle dynamic changes in the design process.

ACKNOWLEDGMENT

The authors wish to thank the anonymous referees and the editors of ETRI Journal, whose constructive comments greatly improved this paper. This paper was partially supported by Inha University in 1999, and partially supported by Brain Korea 21.

REFERENCES

[1] ISO, *Industrial Automation Systems and Integration Product Data Representation and Exchange Part 1: Overview and Fundamental Principles*, Geneva, 1994.

[2] ISO, *Industrial Automation Systems and Integration Product Data Representation and Exchange Part 11: Description Methods: The EXPRESS Language Reference Manual*, Geneva, 1994.

[3] Hussein, K., Feniosky, Pena-Mora, Sriram, R.D., "Cairo: A System for Facilitating Communication in a Distributed Collaborative Engineering," *Proceedings of the Fourth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises, IEEE Computer Society*, 1995.

[4] Schulman M.A., "Collaborative Communication in 3D," *Proceedings of Concurrent Engineering-A Global Perspective*, 1995, pp. 185–190.

[5] Yung-Chou Kao, Grier C.I. Lin, "Development of a collaborative CAD/CAM system," *Robotics and Computer-Integrated Manufacturing*, Vol. 14, 1998, pp. 55–68.

[6] Hyun Kim, Jae Yeol Lee, and Sung-Bae Han, "Process-centric distributed collaborative design based on the Web," *Proceedings of ASME Computers in Engineering Conference*, DETC99/CIE-9081, Las Vegas, Nevada, 1999.

[7] H. Lavana, A. Khetawat, F. Brglez, K. Kozminski, "Executable Workflows: A Paradigm for Collaborative Design on the Internet," *Proceedings of the 34th Design Automation Conference*, Anaheim, CA, 1997.

[8] N. Senin, N. Borland and D. R. Wallace, "Distributed Modeling of Product Design Problems in a Collaborative Design Environment," *CIRP International Design Seminar Proceedings: Multimedia Technologies for Collaborative Design and Manufacturing*, LA, CA, 1997.

[9] J. Owen, 1993, STEP-An Introduction, Winchester.

[10] M. Hardwick and D. Spooner, "An Information Infrastructure for a Virtual Manufacturing Enterprise," *Proceedings of Concurrent Engineering: A Global Perspective*, McLean, VA, 1995, pp. 417–429.

[11] Jasnoch, U., Alok, S.K., and Haas, S., "A Collaborative Environment Based on Distributed Object-Oriented Databases," *Proceedings of the Fourth Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, IEEE Computer Society, 1995.

[12] G. Toye, M.R. Cutkosky, J.M. Tenenbaum and J. Glicksman, "SHARE: A Methodology and Environment for Collaborative Product Development," *Proceedings of Second Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, Morgantown, West Virginia, 1993, pp. 33–47.

[13] M.P. Case and S.C.-Y. Lu, "The Discourse Model for Collaborative Engineering Design," *Computer-Aided Design*, Vol.28, No.5, 1996, pp. 333–345.

[14] Edwards, K.W., "Policies and Roles in Collaborative Applications," *Proceedings of CSCW '96*, 1996.

[15] M. Kamath & K. Ramamritham, "Correctness Issues in Workflow Management," *Distributed Systems Engineering (DSE) Journal : Special Issue on Workflow Management Systems*, Vol. 3, No. 4, 1996.

[16] Klein, M., "Supporting Conflict Resolution in Cooperative Design," *IEEE Transactions on Systems, Man and Cybernetics. Special Issue on Distributed Artificial Intelligence*, Vol. 21, No. 6, 1991.

[17] Gupta, L., Chionglo, J., and Fox, M., "A Constraint-based Model of Communication and Coordination in Concurrent Design Projects," *Proceedings of the 5th Workshop on Enabling Technologies: Infrastructure for Collaborative Enterprises*, 1996.

[18] W. Mueller, G. Lehrenfeld, and N. Wiechers, "Parallel Validation of STEP Files," *Proc. Third International EXPRESS User Group Conference,* Meeting Associates, Clifton Park, N.Y., 1993.

[19] Sang Bong Yoo and S. K. Cha, "Integrity Maintenance in a Heterogeneous Engineering Database Environment," *Journal on Data & Knowledge Engineering*, North-Holland, Vol. 21, No. 3, Feb. 1997, pp. 347–363.

[20] Jochen Alt, "Optimizing EXPRESS Rule Evaluation with File-grained Dynamic Materialization in CAD Applications," *Proc. Fourth International EXPRESS User Group Conference,* Meeting Associates, Clifton Park, N.Y., 1994.

[21] F. Schoenefeld and C. Bohm, "Using EXPRESS Database Technology for Accessing NCBI Genomic Data," in P. Wilson, editor, *Proc. Third International EXPRESS User Group Conference,* Meeting Associates, Clifton Park, N.Y., 1993.

[22] D. Brickley, R.V. Guha, "Resource Description Framework (RDF) Schema Specification," *W3C Proposed Recommendation*, March 1999.

[23] ISO/IS 10303-224, "Industrial Automation Systems and Integration–Product Data Representation and Exchange–Part 224: Application Protocol: Mechanical Product Definition for Process Planning Using Machining Features," *International Organization for Standardization*, 1999.

[24] Michael J. Pratt, "Product Data Representation and Exchange–Integrated Application Resource–ISO/WD 10303-108: Parameterization and Constraints for Explicit Geometric Product Models," ISO TC 184/SC4/WG12 N321, 1999.

[25] ISO, *Industrial Automation Systems and Integration Product Data Representation and Exchange Part 203: Application Protocol: Configuration Controlled Design*, Geneva, 1994.

[26] STEP Tools Inc, *ST-DEVELOPER Reference Manual*, 1997.

Hyun Kim received the B.S., M.S. and Ph.D. degrees in the department of mechanical design and manufacturing from Hanyang University, Seoul, Korea, in 1984, 1987 and 1997, respectively. He had worked for Systems Engineering Research Institute (SERI) from 1990 to 1998. He joined ETRI in 1998 and has worked for the software development related to engineering design such as CAD/CAM/CAE. Currently, he is a senior research scientist in Concurrent Engineering Team, ETRI. His research interests include Concurrent Engineering, Web-enabled CAD, and Virtual Engineering.

Sang Bong Yoo received a B.S. degree in Control and Instrumentation Engineering from Seoul National University, Seoul, Korea in 1982, and M.S. degree in Electrical and Computer Engineering from the University of Arizona in 1986, and a Ph.D. degree in Electrical and Computer Engineering from Purdue University, W. Lafayette, Indiana, U.S.A in 1990. Currently, he is an associate professor of department of Automation Engineering at Inha University, Inchon, Korea. Previously, he was a technical manager at Samsung Electronics Co., Seoul, Korea. His research interests include Engineering Databases, Distributed Systems, System Integration, and Knowledge Management. He is a member of the IEEE Computer Society.

Hyun Chan Lee received the B.S. degree from Seoul National University in 1978, the M.S. degree from Korea Advanced Institute of Science and Technology in 1980, and the Ph.D. degree in Industrial and Operations Engineering from the University of Michigan, Ann arbor in 1988. Before he joined the University of Michigan, he worked for the Pusan Steel Pipe Inc. for three years in the strategic planning department. In 1988, he joined the Korea Electronics and Telecommunications Research Institute as a head of design automation section. He is now an associate professor of Hongik University, Seoul, Korea, in the department of Information and Industrial Engineering. His research interests include CAD/CAM, surface modeling, computer graphics, computational geometry, engineering database, and product information management.