An Infrastructure for Inter-Organizational Collaborative Product Development

Dragan S. Domazet
Gintic, Singapore
ddomazet@gintic.gov.sg

Miao Chun Yan
Gintic, Singapore
cymiao@gintic.gov.sg

Chee Fook Yew Calvin
Gintic, Singapore
cfychee@gintic.gov.sg

Hinny P.H. Kong
School of EEE, NTU, Singapore
ephkong@ntu.edu.sg

Angela Goh
School of Applied Science, NTU, Singapore
asesgoh@ntu.edu.sg

Abstract

Inter-organizational collaborative product development relies heavily on Internet-based technology. As heterogeneous computer environments are typically used in such cases, problems related with data sharing and work coordination at the global level are basic obstacles for widespread implementation of the concept of virtual engineering teams in global product development. In this paper, a software infrastructure is proposed to solve the addressed problems. An event-driven software component framework called the STEP Object Management Framework (SOMF) enables the sharing of common STEP-based product model data, manages collaborative and distributed workflows and provides interfaces to OMG compliant Product Data Management (PDM) systems and workflow management systems. The component framework is based on open standards (CORBA, STEP, Java) and permits a flexible configuration to meet the specific requirements of organizations involved in joint product development projects.

1. Introduction

Inter-organizational collaboration on joint product development requires Internet-based electronic business to business communication. Two or more organizations may form a temporary alliance (known as a virtual enterprise) to jointly develop, produce and market products. By integrating their resources, skill and knowledge, they offer better and cheaper products on the global market. Inter-organizational collaboration in product development offers an attractive opportunity, but is not easy to realize. Typically, collaborating organizations are geographically dispersed and use different computer technology (e.g., CAD/CAM systems, PDM systems). Special problems appear when they are developing complex mechanical products. For example, complex 3D shapes of parts need to be transferred between different systems without any error.

This paper analyzes typical problems related to engineering virtual teams that collaborate on a joint product development project. More specifically, problems of data sharing and work coordination are addressed (Section 2), limitations of the current technology are identified (Section 3), and a possible solution, such as an infrastructure for engineering collaboration, is proposed (Section 4). This solution is based on the use of open standards, such as CORBA [1] and STEP [2-5], Java [6] and Internet technology, and object-oriented database technology [7, 8]. The functionality of a software component framework (still under development), the STEP Object Management Framework (SOMF), is described. It provides users with a shared, active STEP-based integrated product data server, supported by a component that manage collaborative STEP-data related workflows, and interfaces to OMG compliant PDM [9] and workflow management systems [10]. SOMF also provides Java applets that allow authorized Internet users to use SOMF services that provide access and retrieval of STEP databases. Section 5 describes the current SOMF implementation and the technology used. Section 6 highlights benefits to be achieved with the proposed solution. Concluding remarks are given in Section 7.

2. Problem Formulation and Requirements Specification

Two or more groups of engineers collaborate to develop a product. They work in different locations using different systems (e.g., CAD/CAE/CAM systems) and computer platforms linked via the Internet (Fig. 1). They

also need to search and retrieve 3D models of components from suppliers' part libraries. They need to share a common product definition in order to:

- view parts designed by other team members,
- design assigned components or sub-assemblies under constraints specified,
- analyze, discuss and modify design solutions, interactively and/or asynchronously,
- propagate design modifications as soon as possible, and
- review and verify design solution in different phases of a development process.

In order to avoid inconsistent data arising from database updates, it is necessary to coordinate group activities of group members by providing a collaborative workflow management with the following features:

- Run-time modifications of workflow definitions: as collaborative engineering processes are not very structured, and are dynamic in nature.
- Flexible exception handling is necessary as unpredictable situations occur very often in real-life collaborative engineering processes and need to be successfully resolved.
- Activities (human- or program-related) should be driven by different kinds of events generated in a collaborative engineering environment, as it provides a more reliable and responsive working environment.

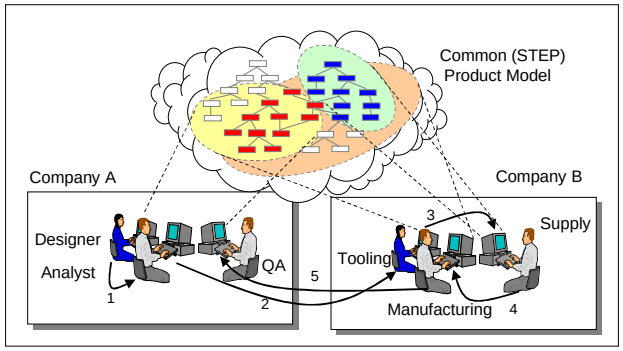

Fig. 1 Inter-organizational group collaboration

Computer systems needed to support virtual teams working on a collaborative product development must satisfy the above requirements. They can be summarized and formulated as follows:

A. Shared Product Data Repository:

A1. Support of *fine grain data sharing* with an appropriate transaction control and flexible data granularity access and locking.

A2. Use of a *neutral or standard product data model* that can be mapped in different application-specific data format, used by end-users with minimal or no loss of information.

A3. Use of *active objects* when needed [11], i.e. objects that generate specific events when certain operations are invoked.

A4. Users of the shared repository may be anywhere on Internet and should access needed data any time (if authorized, of course).

B. Collaborative Workflow Management:

B1. In addition to managing well-structured production workflows (with fixed sequence of activities), there must also be support for *ad-hoc*, *adaptive*, *flexible* and *dynamic workflows* that are modifiable during run-time. Users should be able to specify new activities or modify executing workflows.

B2. There is a need to support *transactional workflows* in order to provide reliable process control in cases of network, equipment or software failures.

B3. *Workflow exceptions* must be catered for when a process reaches a state not supported by the process model.

B4. Besides reacting to user responses according to predefined workflow plans, the Workflow Management should be *event-driven*. In other words, activities can be activated or deactivated according to product data states (database events generated by active data objects), operations of linked applications (application events) or other events from the environment (external or time events).

B5. Workflow management should be *distributed* in order to support networked users who may also specify new workflows or modify existing ones, and to improve the performance of Internet wide workflow management.

3. Current Technology

There are two basic kinds of computer systems that are currently used to support collaborative engineering groups:

a) Groupware tools, such as group editors for sharing text documents, sketches or drawings. They also provide textual, audio and visual communication channels.

b) Product Data Management Systems (PDM Systems) that contain different functional modules that find and deliver needed data or document files from data and document repositories, manage typical workflows, incorporates design review and authorization features, support product viewing and mark-up, propagate notifications, etc.

Despite the fact that most of the products from these two categories are very sophisticated IT products with many users in industry, they do not satisfy all of the requirements specified above.

Groupware tools currently do not provide group 3D CAD editors that can manage and modify 3D CAD data represented in a neutral data format that seamlessly

translates into other formats used by end-users' CAD systems.

Current *PDM systems* have the following deficiencies that limit their effectiveness for collaborative product development:

a) They basically manage and distribute data files or document files *and do not provide sharing of application data at a flexible, user-requested data granularity level.* End-users or applications must checkout all data stored in a file even if they need a small portion of it. This prevents other users or applications from modifying the same file which might be desirable in some cases. On the other hand, it is possible to share the data for read access, such as in design viewing. It also requires long data file transfer over the Internet, as the file contains most of unneeded data. The fixed data access granularity at the file level in many cases is not an optimal one and can slow down the engineering process.

b) Data files use application specific formats and may be modified only by their native applications. Only product viewing and markup is possible when product design originates from different CAD systems.

c) It is difficult to maintain data consistency and integrity of a product data model stored in many data files spread across different locations in different formats.

d) The solution proposed in this paper overcomes the above mentioned deficiencies of the current technology as it satisfies most of the specified requirements for collaborative product development by virtual engineering teams.

4. Proposed solution

We are developing a software framework, called the *STEP Object Management Framework* (SOMF) that may enhance product information sharing capabilities of PDM systems in heterogeneous environments, such as usually found in virtual enterprises (Fig.2). It can also provide direct support of product data sharing by heterogeneous CAD/CAM systems and can be integrated by OMG compliant [10] workflow management systems.

The proposed solution provides three specific features:

1. *Fine grain data sharing:* Any data object may be selectively accessed and used as controlled by an Object-Oriented Database Management System (OODBMS). We use ObjectStore [7].
2. *Active data objects:* Any data objects may be specified as active objects that may generate specific events when exposed to certain database operations. These events can be used to generate notifications, trigger other activities and processes
3. *Dynamic Distributed Workflow Management:* A running workflow may be modified in any time, by adding, deleting or modifying its activities by any authorized process participant.

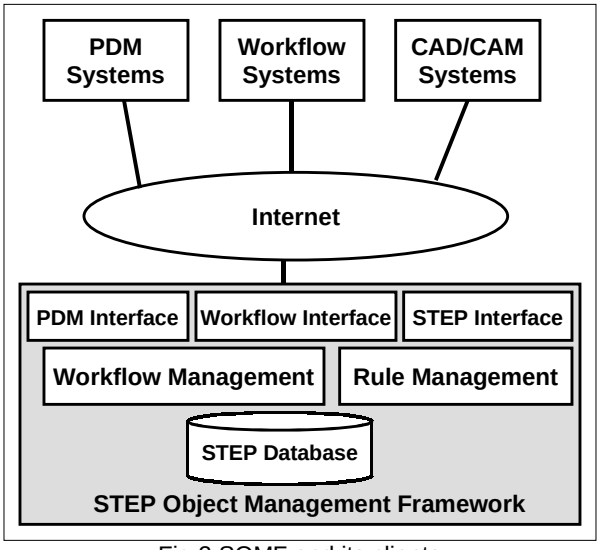

Fig.2 SOMF and its clients

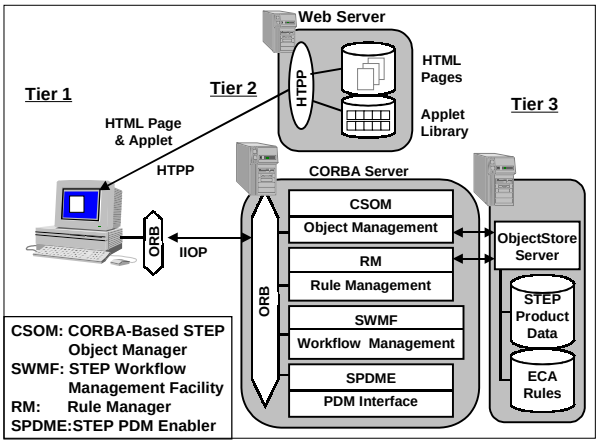

Fig.3 SOMF components

SOMF consists of the following components (Fig. 3):

- ***Common Object Bus:*** The proposed framework leverages the CORBA (the Common Object Request Broker Architecture) technology [1]. The CORBA serves as common software object bus for object communication in a heterogeneous environment.
- ***STEP Object Manager*** *(CSOM – CORBA-based STEP Object Manager)**:*** This module manages storage and retrieval of STEP product data.
- ***Repository of Shared Product Information*** *(ObjectStore Server):* An object oriented database acts as the repository which stores and manages all product information that need to be shared, and provides a unified data access to all collaborative members. Meta data and application data are stored as database objects in one or more databases. The

database schema is compliant with STEP standard ISO 10303-203 [3] in order to support different CAD/CAM/CAE/PDM users.

- ***Collaborative Workflow Manager*** *(SWMF-STEP Workflow Management Facility):* This component coordinates the work of members of virtual product development teams. It manages data-driven workflows that can allow modification of running workflows. SWMF is designed to interoperate with OMG compliant production workflow management systems [10].
- ***Rule Manager*** *(RM):* It implements rules and constraints used by other components, such as CSOM and SWMF. It also provides dynamic behaviors for active STEP data objects and workflow objects. *Event-Condition-Action (ECA) rules* [12-18] *are* used to implement workflow management and active database features.
- ***PDM Interface*** *(SPDME- STEP PDM Enablers):* It is a set of interfaces to allow CSOM to inter-operate with PDM systems that are compliant with OMG PDM Enablers specifications [9].

A set of examples is provided for the end users as the templates to write their own clients to SOMF server components. These examples provide sample Java applet solutions to check-in/check-out STEP data or to query STEP databases.

The next sub-sections briefly describe the functionality of these main SOMF components.

4.1. Product model data sharing

Sharing of product model data provided by an OODBMS offers needed data sharing features in collaborative engineering environments. Users may create sub-assemblies by using different CAD systems (Fig. 4).

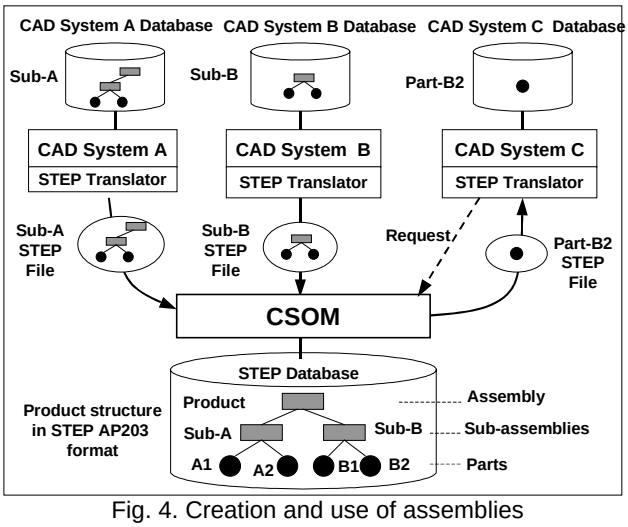

Fig. 4. Creation and use of assemblies

When these sub-assemblies are sent (using standard STEP data files [5]) to CSOM, it reads, converts data from STEP Part 21 data format into C++ objects and stores the submitted data as database objects, linking them with other previously stored product model data objects. The conversion to C++ objects is possible because CSOM uses C++ classes created previously according to STEP AP203 specification [3], after compiling STEP 203 entities specified in the EXPRESS language [4] to C++ classes and ObjectStore database schema [7]. This compilation of data type definition file (written in EXPRESS) into C++ classes and compilation of database schema (written in C++) is performed only initially, when the database is created. Later, it might be needed only occasionally, when the database schema needs to be modified (e.g. when STEP standard is modified, or when improvement of database performance is required). As standards (such as STEP AP 203) do not change frequently, the database schema migration, though possible, is not performed frequently.

The database integrates the entire product model data (geometric and non-geometric) and contains data regarding the product structure and configuration, geometric shape, material and design activities. As only data objects are stored (not data files), users can use different data granularity to access and check-out data, such as all STEP AP 203 unit of functionality (UoF) data sets, sub-assembly or part geometric and non-geometric data, regardless of the way the data were created. When a user needs a product data, he needs to specify the product ID number and the kinds of data needed (such as unit of functionality codes). When storing data, different events may be generated that trigger various ECA rules, such as

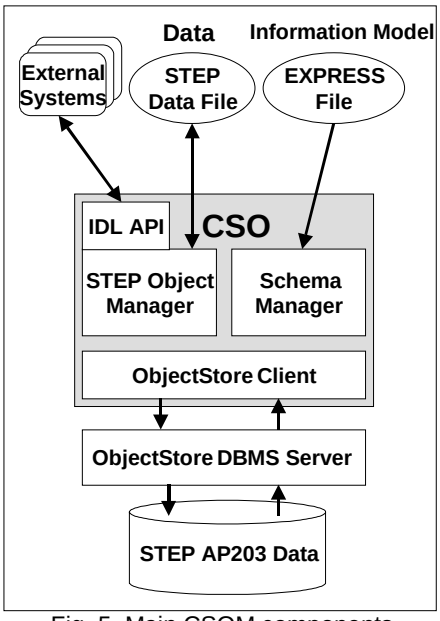

Fig. 5. Main CSOM components

automatic database updates, generation of notifications, etc.

The main benefits of data sharing supported by an OODBMS is data integrity and consistency automatically managed by the OODBMS. Any stored design change is immediately visible to all database users. Users can check out only the needed data using appropriate data granularity. This minimizes the number of objects being locked and allows other users to work with other data objects and minimize the size of data files needed to be transferred to remote users.

Figure 5 shows main CSOM components of SOMF. *Schema Manager* module reads information model file specified in EXPRESS language [4] and creates C++ classes of the database schema to be used for storing product data (this activity may be done periodically when information models need to be modified). The current version of CSOM uses STEP AP203 EXPRESS specification as the basis for its database schema [3]. *STEP Object Manager* module provides all functions related to database query, data access, data storage and retrieval, reading and writing STEP data files. A C++ class library provides appropriate methods for navigational data access. It provides efficient data access and retrieval along the navigational paths through database roots. *IDL API* is a CORBA interface of CSOM specified in OMG Interface Definition Language (IDL) [1]. Any CORBA-based client application may communicate with CSOM by invoking any methods of the CSOM object through its CORBA skeleton. In CORBA terms, CSOM is a server object implementation with an API interface specified in IDL. The current interface supports the following methods:

checkin() – to check-in the data specified in a STEP file (as a database input)

checkout() – to retrieve the requested data set and check-out the data by creating a STEP file to deliver the requested data (for example by specifying the product ID and needed UoFs).

query() – to access and get the data from any stored AP 203 entity (implementation object) using a navigational access path.

viewDB() – to get the product ID numbers of all stored products in a database.

dbfindPID() – to find data related to a product defined by its product ID number.

To illustrate how data may be retrieved, the following example is given. In order to read all BOM data of a product, an appropriate database root attached to a class (“next_assembly_usage_occurrence”) was created (named “NauoRoot”):

db->create_root(“NauoRoot”)->
set_value(next_assembly_usage_occurrence) (1)

The following operation is specified when BOM data need to be stored:

db->find_root(”NauoRoot”)->
set_value(next_assembly_usage_occurrence) (2)

Through “NauoRoot” root all objects in the next_assembly_usage_occurrence list can be accessed and extracted by another ObjectStore (OS) API:

db->find_root(“NauoRoot”)->get_value() (3)

If a given product needs to be extracted, its ID (PID) number is adjusted as a top level PID. To extract BOM data, three root objects (such as next_assembly_usage_occurrence) need to be used. Each of three OS lists holding BOM objects will be traversed by three OS cursors. An OS cursor keeps track of the iteration over an OS list to ensure every element in the list to be visited. For each list, a search path must be created to reach a part ID object from any one of objects in the list. Sometimes, this path may be very long and complex. Through this path, a part ID object will be retrieved. Then an evaluation will be performed to check whether an object in the list matches that one given by the user. If it is true, return this object; Otherwise, the next object in the list will be examined. Only requested objects will be moved and written into a STEP file and sent to the user. All objects extracted from these three lists are then moved and saved as a STEP file by calling Rose API [26]: ROSE.saveDesign(). This API translates objects from the Rose internal data format into the STEP Part 21 file format. C++ methods were developed for storing and retrieving STEP AP203 data at different levels of granularity (such as UoF level, STEP entity or attribute level).

CSOM is based on some modules of PIKS (Product Information and Knowledge Server) developed at Gintic [19. 20].

4.2. Collaborative workflow management

Most commercial workflow management systems are not very appropriate for dynamic engineering environments and do not support ad-hoc modifications of running workflows. Product development processes may be structured only at a higher level, such as design verification and release procedures. The most creative and important engineering activities, such as activities in conceptual design and in pre-release design process are difficult to model. On the other hand, design collaboration and management systems that are based only on an e-mail system and groupware tools do not provide the necessary level of process control and product data management. The workflow management module of SOMF (called STEP Workflow Management Facility – SWMF) is

designed to fill the gap between production workflow management system, and groupware tools or e-mail systems that are normally based on ad-hoc network communication. SWMF manages STEP data-related sub-workflows for virtual group collaborations. SWMF is able to communicate with any OMG compliant production workflow system used in the organization. The following are the basic functions of SWMF:

- *Interoperability with the master workflow management system:* Communication with commercial production workflow systems based on the OMG compliant interface [10]. SWMF executes STEP-data related sub-workflows (Fig. 6) only using most of the build-in functions of the master workflow management system (such as user administration, project set-up, etc.).

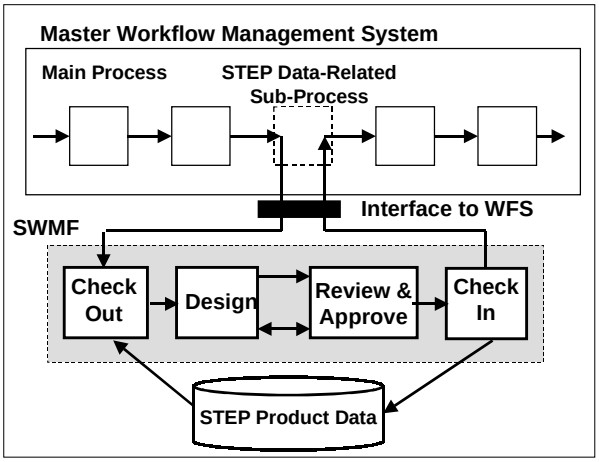

Fig. 6 SWMF managed sun-workflow

- *Support of dynamic workflows:* The specification of a sub-workflow (e.g. process description/model) may be modified during its execution.
- *Data-driven workflow management:* Activities are driven by events, generated not only by the process model, but also by active database objects (Fig. 7). In both cases, Event-Condition-Action (ECA) rules (defined directly by users or generated by the process modeling tool) are triggered to execute the specified activities when an appropriate event is detected and the specified conditions are met (such as database states). As ECA rules are first class CORBA objects, they may be created, modified, queried and destroyed during the execution of an affected workflow thus modifying its execution. Sets of ECA rules specify a workflow or part of it, and may be distributed according to users' preferences and performance requirements. Users may create ECA rules by using a specially developed applet (provided by SOMF).

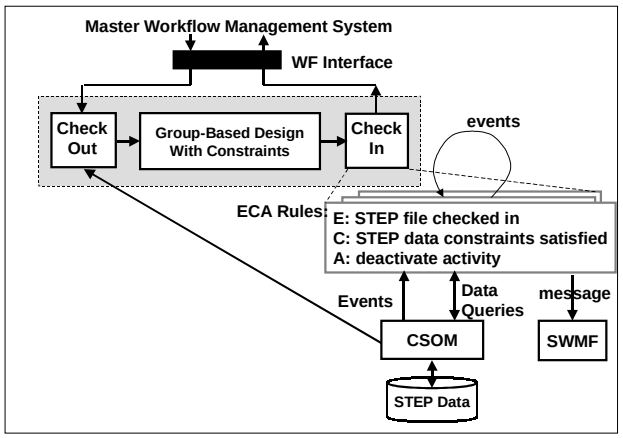

Fig. 7 Active data-driven workflow management

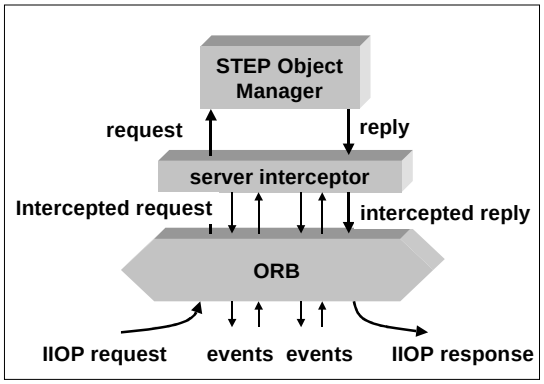

Fig. 8 Events generation by ORB interceptors

Figure 8 shows how the database events are generated. The interceptor of the ORB is used to raise events when specified operations on specific database objects are intercepted. Detected events are then sent through CORBA Event Channels (Fig. 9) to Rule Managers. There are two kinds of Rule Managers, each related to one of two kinds of ECA rules:

- ECA rules governing active database objects, and
- ECA rules governing workflow activities.

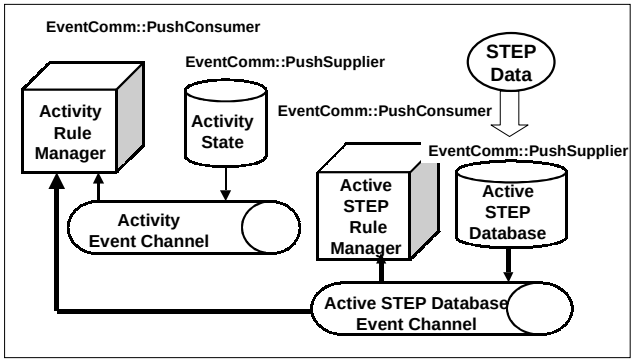

Fig. 9 Rule managers of SOMF

Upon detecting an event, the Rule Manager will search for matching ECA rules. Only matched rules are triggered. A triggered rule executes its actions, if and only if its

condition is satisfied. The coupling modes of ECA rules specify the execution model between the rule triggering and condition testing, as well as between the condition testing and action execution.

4.3. Integration with PDM systems

SOMF is not a PDM system. It is a set of server software components designed to be integrated with PDM systems (by using OMG standard PDM interface specification [9]) in order to extend some of their functions. More specifically, SOMF allows users of different PDM systems *to share STEP-based product model data* stored in a common active STEP database and to appropriately manage related sub-workflows. The STEP AP203 specification [3] specifies most of the product-related data that are normally managed by PDM systems (such as product structure, product configuration, versions, design activities, etc.). SOMF stores PDM application data received from integrated PDM systems in its STEP AP203 database, and delivers requested data to integrated PDM systems and other systems. PDM systems are responsible for user administration, authentication, and communications with users, as well as data delivery. STEP AP 203 database of SOMF (managed by CSOM) contains replicated PDM data that need to be shared. As these data are in ISO STEP format, they may be used and shared by different PDM and CAD/CAM systems (Fig. 10). When a user modifies a PDM data (such as design object, or an activity), the PDM system modifies its application data and meta data repositories (as usual), and via the PDM interface of SOMF. SOMF may further propagate data modifications (such as design changes) to other PDM or CAD/CAM systems authorized to share its STEP databases. In this way SOMF may link users of different PDM or CAD/CAM systems.

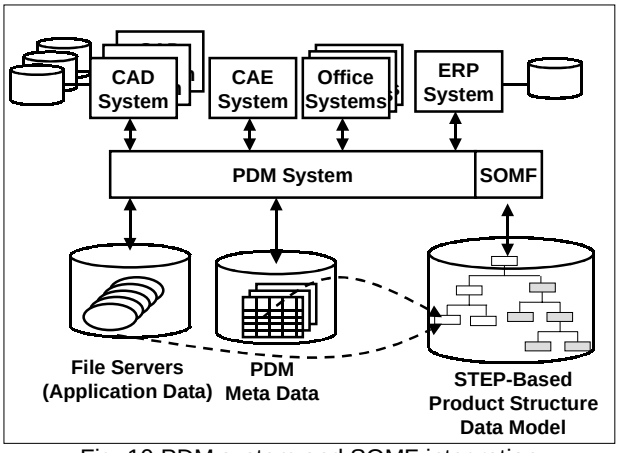

Fig. 10 PDM system and SOMF integration

4.4. Integration with CAD/CAM systems

Different CAD/CAM systems may share product data stored in STEP database managed by SOMF/CSOM. They can be directly linked with CSOM via its IDL interface. The level of integration depends of the CAD-CSOM client software. A specially developed CAD/CSOM interface may provide a tight integration and seamless data exchange between two systems. Currently, SOMF provides Java applets that any authorized user can download in order to send STEP data to CSOM (check-in operation), to get needed STEP data (check-out operation) or to get any STEP database object (query operation). In these cases, standard STEP Part 21 Data File [5] is used to transfer data between two systems. The only requirement for CAD/CAM users is that their CAD/CAM system supports STEP AP203 interface (most CAD/CAM systems now provide such interfaces) and that they use a Java-enabled Web browser that is also CORBA enabled. (Otherwise, the client platform must use an ORB). Figure 11 demonstrates a STEP file check-in operation. A user gets an HTML page from the Web Server of SOMF with needed applets. As these applets use CORBA, further communication with SOMF requires the IIOP protocol [1]. By using CORBA FTP, the STEP file from the client machine is transferred over the Internet to the SOMF machine, where data from the STEP file are converted into C++ database objects and stored onto STEP AP203

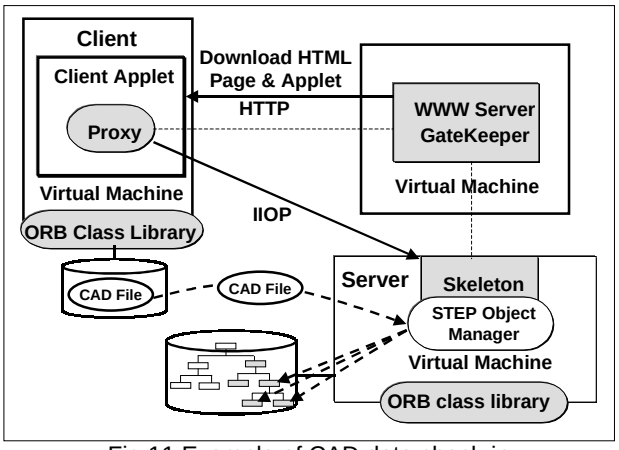

Fig.11 Example of CAD data check-in

database.

Using this loose integration of CAD/CAM systems with SOMF, different systems may share product designs stored in STEP databases managed by SOMF/CSOM.

5. Implementation issues

CSOM module is developed with C++ and use C++ based ObjectStore OODBMS in order to provide efficient retrieval of product data. As integrated product models are very complex, contain many data types and have complex

relationships among classes, STEP data models are not only very big (with a lot of data) but also very complex (with many relationships and classes). Object-oriented databases provide efficient management of complex data models, and C++ provides a fast navigational access to stored STEP objects. The Rose class library of ST Developer [21] is used to handle STEP data (EXPRESS compilation, STEP file reading and writing, etc.). The CORBA-based IDL API provides a universal interface between CSOM and the other components of SOMF and with external applications (such as CAD/CAM systems).

ECA rules are specified in Java, but condition and action parts (known as rule body) are specified in C++ in order to tightly integrate with ObjectStore operations and improve performance (Fig. 12). A set of rule bodies may be grouped together and use an ObjectStore client to communicate with ObjectStore database directly. ObjectStore database stores not only product application

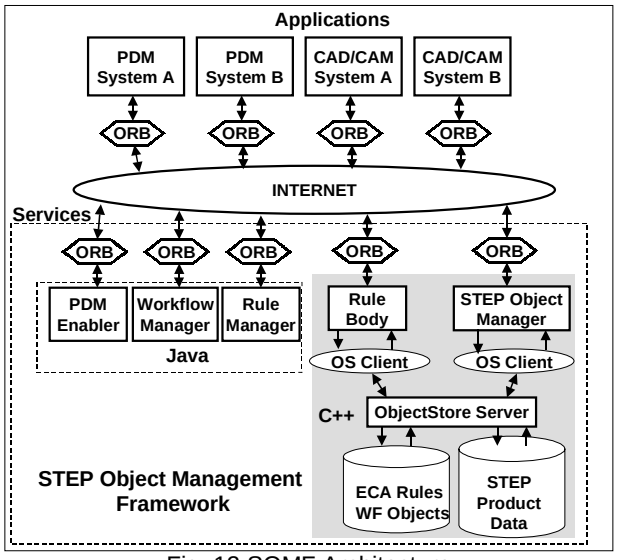

Fig. 12 SOMF Architecture

and meta data in STEP format, but also SOMF-specific workflow data and ECA rules, as first class Java objects. When writing ECA rules, users need to develop C++ methods used in "rule body" (condition and action part of the ECA rule). Obviously, users need not only to know C++, but also STEP AP203 database schema. Writing ECA rules therefore may not be convenient for end-users of SOMF (e.g. engineers), as they may not be able to write ECA rules. To overcome this, rules may be written by a rule administrator (trained to write rules), or using a special tool developed for this purpose (currently such tool does not exist in SOMF).

VisiBroker CORBA ORB [22] is used to provide the communication between distributed CORBA objects (objects with IDL interfaces). Besides basic ORB functions, the following CORBA services are also used: Naming, Event, and Transaction (planned but not yet implemented) services.

The project team plans to convert all Java SOMF components into Enterprise JavaBeans (EJB) [23, 24] in order to allow their easy installation on EJB applications servers, and their integration with other EJB application components. JBuilder IDE [25] is used for Java programming and for development of GUI applets that allow users to query STEP databases managed by CSOM. Rational Rose [26] is used for analysis, design and implementation of distributed objects of SOMF.

6. Benefits

The STEP Object Management Framework (SOMF) provides sharing of product model data represented in standard STEP data formats and provides some features of collaborative workflows, such as modification of running sub-workflows related to use of STEP-based product databases. SOMF can be used directly by CAD/CAM systems, but CAD/CAM users may also use SOMF services through their PDM system.

Most current PDM systems support STEP AP203 protocol and provide STEP AP203 interfaces. These interfaces may be used to exchange some PDM data between PDM systems and applications (such as CAD data for CAD applications). As this is only an interface, linked applications and PDM systems need to manage STEP files (data translation, data delivery and storage). STEP files are used to transfer application data between systems, and applications manage their own data. When data is modified (e.g., a design is changed), the PDM system needs to create a STEP file with the modified product data and to deliver it to all affected applications, where local copies of the same data need to be updated. This management of product data model consistency is very complicated and difficult in cases when many users/applications use and modify application data. As SOMF uses a central database server to store all product model data that is needed to be shared, it is much easier to maintain its consistency and integrity, as this is a function that is performed well by database management systems (DBMS). As SOMF is an active, event-driven development environment, design changes may be propagated immediately and sharing of product data is easily and reliably implemented.

SWMF components of SOMF when integrated with PDM systems may provide a good collaborative workflow management tool for design and engineering groups, where engineering processes are not well structured and require flexibility, exception handling, and adaptability to process changes.

SOMF uses relevant standards that support distributed and heterogeneous computer systems (e.g. CORBA) and different data models (e.g. STEP). It can be easily integrated with other systems that support the same standards. SOMF components can be added and linked with other application server components (distributed or centralized) providing an appropriate computer environment for global product development that is essential for virtual enterprises, i.e. inter-organizational collaboration. When implemented as Enterprise JavaBeans, SOMF components, will be highly distributed and scaleable. They can be installed on any computer with a standard Java Virtual Machine.

7. Conclusions

The STEP Object Management Framework (SOMF) described in this paper, supports engineering collaboration in heterogeneous computer environments. Such an environment is essential to Inter-organizational collaborative product development that relies heavily on Internet-based technology. SOMF supports collaboration of engineers from different organizations working on joint product development projects. It may be integrated with different PDM systems used by the collaborating organizations and can add new functionality, such as STEP product data sharing and event-driven collaborative/dynamic workflow management.

Acknowledgment

This paper presents some of initial results of the research project “Advanced Database Technologies for Concurrent Engineering” of Gintic Institute of Manufacturing Technology. This on-going four-year project that started in 1997, is carried out in collaboration with the School of Applied Science and the School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore.

References

[1] The Common Object Request Broker: Architecture and Specification, Revision 2.1, 97-019-01, OMG, August 1997.

[2] IS 10303 Product Data Representation and Exchange - Part 1: Overview and Fundamental Principles, ISO, 1994

[3] IS 10303 Product Data Representation and Exchange - Part 203: Application Protocol: Configuration Controlled Design, ISO, 1994

[4] IS 10303 Product Data Representation and Exchange - Part 11: Description Methods: The EXPRESS Language Reference Manual, ISO, 1994

[5] IS 10303 Product Data Representation and Exchange - Part 21: Clear Text Encoding of the Exchange Structures, ISO, 1994

[6] Java 2 Platform, v.1.2.2, Sun Microsystem Inc.

[7] ObjectStore, Release 4, User Guide, Object Design Inc., Burlington, MA01803, USA., 1996

[8] ST-ObjectStore, Version 1.5.0, Reference Manual, STEP Tools Inc., Troy, USA, 1996

[9] PDM Enablers, Revised mfg-02-02, OMG

[10] Workflow Management Facility, bom/98-06-07, OMG, 1998.

[11] Buchmann A.P., Active Object System, A. Dogac, M.T. Özuu, A. Biliris and T. Sellis (editors), “Advances in Object-Oriented Database Systems, NATO ASI Series, Series F: Computer and System Sciences, Vol. 130, pp. 201-224, Berlin Heidelberg: Springer-Verlag, 1994

[12] Dayal, U., Buchmann, A.P., McCarthy, D.R., Rules Are Objects Too: A Knowledge Model For An Active, Object-Oriented Database System, Proc. 2nd Int. Workshop on Object-Database System, K.R.Dittrich (Ed.), Lecture Notes in Computer Science, Springer, 224:129-143, 1988

[13] Dayal. U., Active Database Management Systems, Proc. 3rd Int. Conf. on Data and Knowledge Bases, , pp: 150-157, Jerusalem, Israel, June 28-30, Morgan Kaufman, 1988

[14] Hsu, M., Ladin, R., McCarthy, D.R., An Execution Model for Active Data Base Management Systems, Proc. 3th Int. Conf. on Data and Knowledge Bases, pp.: 171-179, Jerusalem, Israel, June 28-30, Morgan Kaufman,1988

[15] Gatziu, S., Dittrich, K.R., Events in an Active Object-Oriented Database System, Rules in Database System , Eds.: Paton, N.W., Williams, M.H., Springer-Verlag, pp :23-39, 1993

[16] Kim, W., Lee, Y.-J., Seo, J., A Framework for Supporting Triggers in Object-Oriented Database Systems, Int. Journal of Intell. and Cooperative Information Systems, Vol.1, pp. 1:127-143, 1992

[17] Chakravarthy S., Nesson S., Making an Object-Oriented DBMS Active: Design, Implementation, and Evaluation of a prototype, Proc. “Extending Database Technology”, 1990, Lecture Notes in Computer Science 416, Springer-Verlag, pp. 391-406, 1990

[18] Domazet, D.S., Chong F.N., Sng D., Ho N.C., Lu, S.C.-Y., Active Data-Driven Design Using Dynamic Product Models, Annals of the CIRP, Vol. 44/1:109-112, 1995

[19] D.Domazet, Q.Z. Yang, Y.Z. Zhao, PIKS: Product information and knowledge servers for concurrent engineering environments, Proceedings of the 4th International Conference on Computer Integrated Manufacturing, Volume 2, Editors: A. Sen, A.I. Sivakumar,

R. Gay, Springer, Singapore, pp.1071-1080, 21-24 October 1997

[20] Q.Z.Yang, D.Domazet, Y.Z. Zhao, Development of a STEP-based information server for concurrent engineering applications, *Advances in Concurrent Engineering - CE97*, Edited by: Subra Ganesan, Series Editor: Beren Prasad, presented at Fourth ISPE International Conference on Concurrent Engineering: Research and Applications, Oakland University, Rochester, Michigan, USA, Technomatic Publishing Co., pp. 255-262, August 20-22, 1997

[21] ST-Developer, Version 1.5, ROSE Library Reference Manual, STEP Tools Inc., Troy, USA, 1996

[22] VisiBroker , Version 3.2, Inprise, 1998

[23] Enterprise JavaBeans, Version 1, Sun Microsystems, March 21 1998

[24] Orfali R. and Harkey D., Client/Server Programming with Java and CORBA, Second Edition, John Wiley & Sons, 1998

[25] JBuilder , Version 2.1, Inprise, 1998

[26] Rational Rose, User Manual, Version 4.0, Rational Software Corporation, Santa Clara, 1997