

Wise-ShopFloor: A Web-Based and Sensor-Driven e-Shop Floor*

Lihui Wang**

e-mail: lihui.wang@nrc.gc.ca

Weiming Shen

Sherman Lang

Integrated Manufacturing Technologies Institute, National Research Council of Canada, 800 Collip Circle, London, Ontario N6G 4X8, Canada

Targeting the remote monitoring and control of shop floors, this paper proposes a new framework called Wise-ShopFloor (Web-based integrated sensor-driven e-ShopFloor) that can be applied to distributed manufacturing environments. It utilizes the latest Java technologies (Java 3D and Java Servlet) for system implementation. This Web-based framework allows users to monitor and control a distant shop floor device with visual helps enabled by Java 3D models instead of camera images. The behavior of a 3D model is driven by sensor signals of its physical counterpart. A prototype system is developed to demonstrate its application on shop floor monitoring and control.

[DOI: 10.1115/1.1647122]

Keywords: Web-Based Monitoring, Remote Control, Java 3D, Sensors, Distributed Manufacturing

1 Introduction

Since its debut in early 1990s, the Web has gained a wide acceptance in both academia and industry, and has been used by many as a medium to share information and knowledge. Today, it is widely used for the development of collaborative software environments to support dispersed working groups and organizations because of its platform, network and operating system transparency, and its easy-to-use user interface—Web browser. In addition to the Web technology, Java has brought about a fundamental change in the way that applications are designed and deployed. Java's "write once, run anywhere" feature has reduced the complexity and cost traditionally associated with software develop-

ment on multiple distinct hardware platforms. With Java, the browser paradigm has emerged as a compelling way to produce manufacturing applications over the Internet.

The objective of this research is to develop a Web-based digital shop floor framework called *Wise-ShopFloor* (Web-based integrated sensor-driven e-ShopFloor) for distant shop floor monitoring and control. The *Wise-ShopFloor* can serve real-time data from bottom up, as a constituent component of e-manufacturing. The framework is designed to use the popular client-server architecture and VCM (view-control-model) design pattern with secured session control. The proposed solutions for meeting both the user requirements demanding rich data sharing and the real-time constraints are: (1) using interactive Java 3D models instead of bandwidth-consuming camera images for visualization; (2) transmitting only the sensor data and control commands between models and device controllers for monitoring and control; (3) providing users with thin-client graphical interface for navigation; and (4) deploying major control logics in a secured application server. A proof-of-concept prototype system is developed on top of the framework to demonstrate shop floor monitoring and control. It utilizes the latest Java technologies, including Java 3D and Java Servlets, as enabling technologies for system implementation. Instead of camera images, a physical device of interest is represented by a Java 3D model with behavior control nodes embedded. Once downloaded from an application server, the Java 3D model works on behalf of its counterpart showing behaviors for visualization at a client side, but remains alive by connecting with the physical device through low-volume message passing. The goal of our combined Web-based and sensor-driven approach is to significantly reduce network traffic, while still providing end users with an intuitive environment. The largely reduced network traffic makes real-time monitoring, control, inspection, and troubleshooting practical for users on relatively slow hook-ups such as modem connections. In the near future, open-architecture devices (such as OpenPLCs and Open-CNC Controllers, etc.) will have Web servers and Java virtual machines embedded. This will make the proposed *Wise-ShopFloor* framework more efficient for real-time monitoring and control in distributed manufacturing environments.

This paper presents fundamentals of the framework for building Web-based collaborative systems that can be used in distributed manufacturing environments. It first reviews the related work, followed by the description of the concept and architecture of the framework. The *Wise-ShopFloor* concept is then demonstrated and validated through a case study on device modeling, monitoring, and control.

2 A Brief Literature Review

Since 1993 shortly after the emergence of the Web, a number of methods and frameworks have been proposed for building Web-based systems. Most of them are developed for collaborative design, Web-based rapid prototyping, project management, and conflict resolutions during collaboration, e.g. *WebCADET* [1] for distributed design support, *CyberCut* [2] for Web-based rapid machining, and *NegotiationLens* [3] for conflict resolution. In terms

*Copyright 2003 National Research Council Canada. To copy, republish, post on servers or redistribute to lists requires prior specific permission from the National Research Council Canada.

**Corresponding author.

Contributed by Embedded/Ubiquitous Computing Committee for publication in the JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING. Manuscript received Feb. 2003; Revised Dec. 2003; Associate Editor: P. Wright.

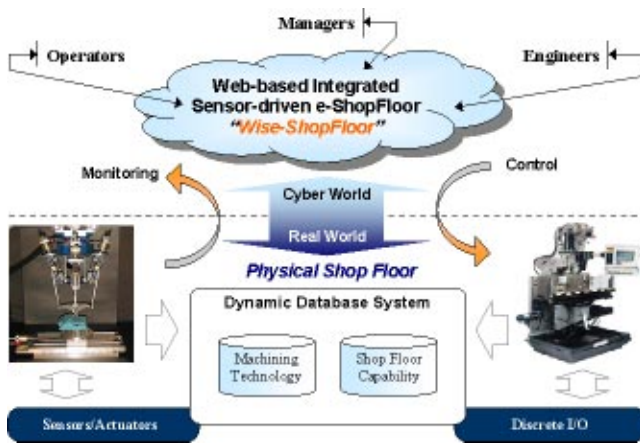


Fig. 1 Concept of Wise-ShopFloor

of technologies used in the existing systems, HTML, Java applets, ActiveX, and VRML (virtual reality modeling language) are widely adopted for developing client-side user interfaces. At the server side, technologies including JSP (JavaServer Pages), Java Servlets, and XML are quickly obtaining attentions for system developments. To facilitate viable collaborative systems, application servers must engage users in a 3D graphical interaction in addition to the dialog-like data sharing, because remote users need active and visual aids to coordinate their efforts in a distributed environment. Today, collaborative manufacturing tops the wish list for many manufacturers. Unfortunately, most of the manufacturing equipment of today does not have the built-in capability to transmit and receive data. Few of the available Web-based systems are designed for shop floor monitoring and control or for advanced factory automation. Some related systems listed below are limited in their functionality and platform requirements.

In the area of event monitoring, the latest *Cimplicity* from GE Fanuc Automation (USA) allows users to view their factory's operational processes through an XML-based *WebView* screen, including all alerts on every *Cimplicity* system [4]. The *Factory-Flow* from Unigraphics Solutions (USA) is an off-line factory-floor layout planning, material handling, and simulation package [5]. By most estimates, the number of CNC machines capable of linking to the Internet is less than 10% of the installed base [5]. Seeking the opportunity in linking CNC machines with the Internet, MDSI (Ann Arbor, MI, USA) uses *OpenCNC* [6], a Windows-based software-only machine tool controller with real-time database, to automatically collect and publish machine and process data on a network. In 1999, Hitachi Seiki (Japan) introduced *FlexLink* [7] to its turning and machining centers. Working together with *PC-DNC Plus* from Refresh Your Memory (USA), *FlexLink* is able to do in-process gauging, machine monitoring, and cycle-time analysis. Since 1998, Mazak (Japan) has operated its high-tech *Cyber Factory* concept [8] at its headquarters in Oguchi, Japan. The fully networkable *Mazatrol Fusion controllers* allow Mazak machines to communicate over wireless factory networks for applications including real-time machine tool monitoring and diagnostics. In addition, Japan-based Mori Seiki introduced a *CAPS-NET* system that polls machine tools on Ethernet at settable increments, usually five-second or longer, for engineers to get updates on machine tools' run-time status in production [9]. To bring legacy machine tools with only serial ports on-line, e-Manufacturing Networks Inc. (Canada) introduced its *ION Universal Interface* and *CORTEX Gateway* [10] to help the old systems go online, and to monitor information flow and the status of the CNC machine tools on the network.

Despite all the accomplishments, the available systems are either for off-line simulation or for monitoring only. Most systems require a specific application to be installed instead of a standard

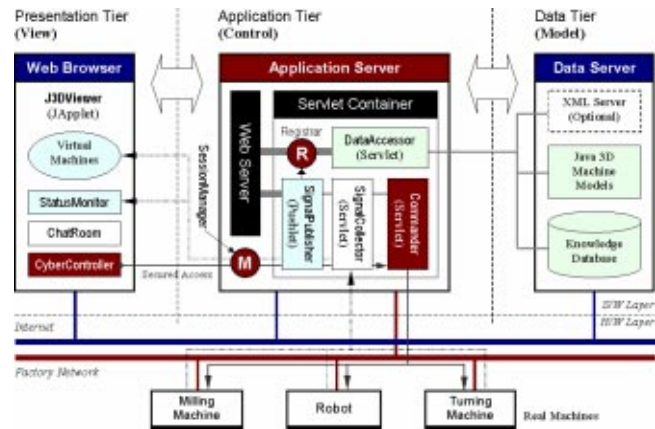


Fig. 2 Three-tier architecture of Wise-ShopFloor

Web browser, which eliminates a system's portability. Advanced and distributed shop floor monitoring and control remain impractical as Web-based applications due to the real-time constraints. Reducing network traffic and increasing system performance are the major concerns in Web-based system developments.

3 Concept of Wise-ShopFloor

The Wise-ShopFloor is designed to provide users with a Web-based and sensor-driven intuitive shop floor environment where real-time monitoring and control are undertaken. It utilizes the latest Java technologies for system implementation. Instead of using camera images (usually large in data size), a physical device of interest (e.g. a milling machine or a robot) can be represented by a scene graph-based Java 3D model in an applet with behavior control nodes embedded. Once downloaded from an application server, the Java 3D model is rendered by local CPU and works on behalf of its remote counterpart at a client side. It remains alive by connecting with the physical device through low-volume message passing (sensor data and user control commands). The 3D model provides users with increased flexibility for visualization from various perspectives, such as walk-through or fly-around that are not possible by using stationary optical cameras. The significantly reduced network traffic makes real-time monitoring and control, etc. practical for users on relatively slow hook-ups through a shared *Cyber Workspace* [11]. By combining virtual models with real devices through synchronized real-time data communications, the Wise-ShopFloor allows engineers and shop floor managers to assure normal shop floor operations and Web-based troubleshooting—particularly when they are off-site.

Figure 1 shows the Wise-ShopFloor concept. Although the Wise-ShopFloor framework is designed as an alternative of camera-based monitoring systems, an off-the-shelf Web-ready camera can easily be switched on remotely to capture unpredictable real scenes for diagnostic purposes, whenever it is needed. In addition to real-time monitoring and control, the framework can also be extended and applied to design verification, remote diagnostics, and virtual machining. It is tolerant to hostile, invisible or non-accessible environments (e.g. inside a nuclear reactor or outside a space station).

4 Architecture Design

As shown in Fig. 2, the framework is designed to use the popular client-server architecture and VCM (view-control-model) design pattern with built-in secure session control.

The proposed solutions for meeting both the rich visual data sharing requirements and the real-time constraints are listed below.

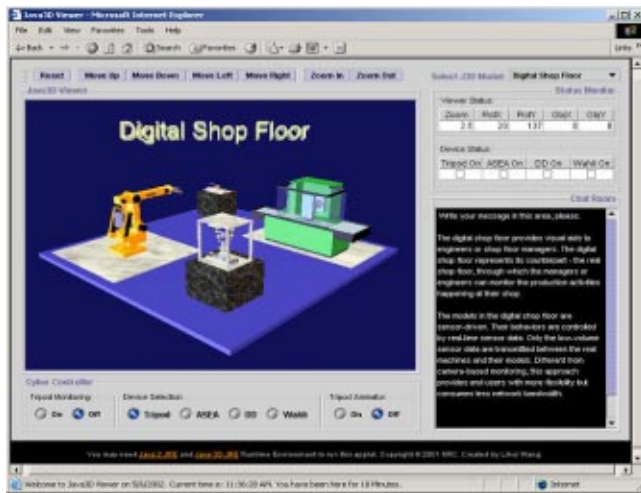


Fig. 3 Web user interface for shop floor monitoring and control

1. Using interactive scene graph-based Java 3D models instead of bandwidth-consuming camera images for visualization;
2. Transmitting only the sensor data and control commands between models and device controllers for remote monitoring and control;
3. Providing users with thin-client graphical user interface (GUI) for shop floor navigation; and
4. Deploying major control logics in a secured application server.

The mid-tier application server handles major security concerns, such as session control, viewer registration, data collection/distribution, and real device manipulation. A central *SessionManager* is designed to look after the issues of user authentication, session control, and sensitive data logging. All initial transactions need to pass through the *SessionManager* for access right authorization. In a multi-client environment—the *Wise-ShopFloor*, different clients may require different sets of sensor data for device monitoring. It is not efficient to have multiple clients sharing the same model talk with the same device directly at the same time. Instead, a publish-subscribe design pattern is adopted to collect sensor data and distribute them to the right clients, efficiently. As a server-side module, the *SignalCollector* is responsible for sensor data collection from networked physical devices. The collected data are then passed to another server-side module called *SignalPublisher* that in turn multicasts the data to the registered subscribers (clients) through applet-servlet communication. A *Registrar* is designed to maintain a list of subscribers with the requested sensor data. A Java 3D model thus can communicate indirectly with sensors no matter where the clients are, inside a firewall or outside. HTTP streaming is chosen as the data communication protocol for the best combination between applets and servlets. For the same security reasons, a physical device is controllable only by the *Commander* that resides in the application server. Another server-side component called *DataAccessor* is designed to separate logical and physical views of data. It encapsulates JDBC (Java DataBase Connectivity) and SQL codes and provides standard methods for accessing data.

Although the global behaviors of Java 3D models are controlled by the application server based on real-time sensor signals, remote users still have the flexibility of monitoring the models from different perspectives, such as selecting different 3D machine models and changing viewpoint location (translation, rotation, orbiting, zooming) or orientation (panning, tilting), through *J3DViewer* at a client's side. Authorized users can submit control commands through *CyberController* to the application server. The *Com-*

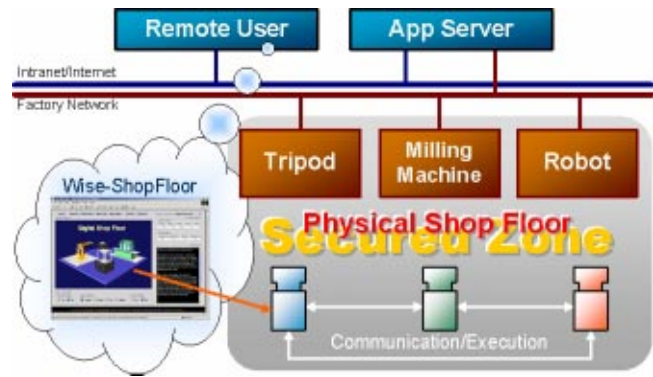


Fig. 4 Indirect secure access to physical shop floor

mander at the server-side then takes over the control for real device manipulations. Another client-side module called *Status-Monitor* can provide end users with a view of run-time status of the controlled device. For the purpose of collaborative troubleshooting, a *ChatRoom* is included in the framework for synchronized messaging among connected users.

A proof-of-concept prototype is developed to demonstrate its application on remote monitoring and control. Figure 3 shows one snapshot of the Web user interface of the prototype. A more detailed discussion from device modeling to control is provided in Section 6 through a case study.

5 Shop Floor Security

According to an NCMS report [12], there is a growing consensus that linking shop floor hardware to the Internet will become the backbone technology for collaborative manufacturing. However, a major concern of implementing Internet or Web-based collaborative manufacturing systems is the assurance that proprietary information about the intellectual property owned by the organization or information about the company's operations is available only to those authorized individuals. Any Web-based collaborative systems must accommodate privacy of the individuals and organizations involved in collaborative activities. In a highly competitive manufacturing environment, the information about the operations of or the information provided by individuals or organizations should only be shared by those involved. Clearly, it is also important to avoid security disasters of hardware. Web-based remote monitoring and control typically involve sharing information in the form of detailed run-time operations, as well as real-time and mission-critical hardware controls. For general acceptance of the *Wise-ShopFloor*, the secrecy of the proprietary information must be properly maintained. To meet security requirements, our approach depends on a security infrastructure built into the Java platform. This security infrastructure consists of byte-code verification, security policies, permissions, and protection domains. In addition to the security infrastructure, other security and privacy issues are considered in the framework for implementation, including digital rights management for information access and sharing, data encryption, and process confidentiality protection.

Figure 4 shows how a remote user can get access indirectly to the real shop floor without violating shop floor security policy. All data communication between the end user and a shop floor device goes through the application server, and is processed by a server-side module before passing the data to its receiver. As mentioned in Section 4, only the server-side modules are allowed to collect sensor data or manipulate devices within their limits. On the other hand, all end users are physically separated from the real shop floor by using segmented networks (Intranet/Internet, and Factory Network) with the application server as a gateway.

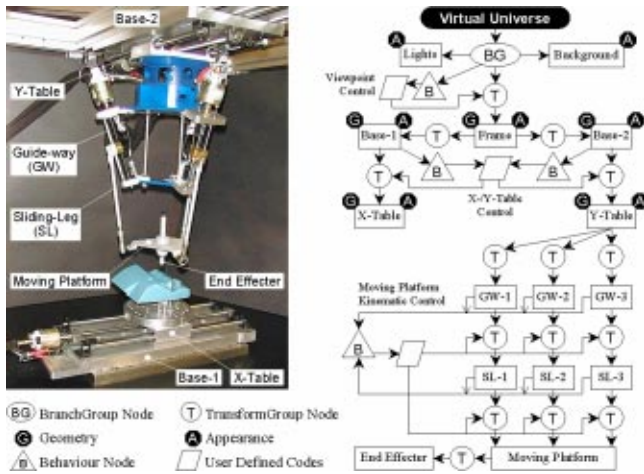


Fig. 5 Java 3D scene graph architecture of Tripod

6 Case Study

This section describes how a physical device is modeled, monitored, and controlled by applying our Wise-ShopFloor concept. The Tripod is one type of parallel kinematic machines developed at authors' lab [13]. Instead of using camera images, the Tripod is modeled by using the scene graph-based Java 3D models with behavior control nodes embedded. The 3D model behaves in the same way of its counterpart for remote monitoring at client-side, facilitated by the model-embedded kinematics and sensor signals of the real Tripod.

6.1 Java 3D Modeling for Tripod. Java 3D is designed to be a mid to high-level fourth-generation 3D API [14]. What sets a fourth-generation API apart from its predecessors is the use of scene-graph architecture for organizing graphical objects in the virtual 3D world. Unlike the display lists used by the third-generation APIs (such as VRML, OpenInventor, and OpenGL), scene graphs can mercifully hide a lot of the rendering details from users while offering opportunities for more flexible and efficient rendering. Because Java 3D is part of the Java pantheon, it assures users ready access to a wide array of applications and network support functionality [15]. Java 3D differs from other scene graph-based systems in that scene graphs may not contain cycles. The individual connections between Java 3D nodes are always a direct relationship: parent to child. It is worth of mentioning that in addition to Java 3D, OpenSceneGraph [16] and OpenSG [17], as emerging open source standards, provide similar multi-platform supports for scene graph model creations. Figure 5 illustrates the Java 3D scene graph architecture of the Tripod. This test bed is a gantry system, which consists of an x-table and a Tripod unit mounted on a y-table. The end effector on the moving platform is driven by three sliding-legs that move along three guide-ways, respectively.

As shown in Fig. 5, the scene graph contains a complete description of the entire scene with a virtual universe as its root. This includes the geometry data, the attribute information, and the viewing information needed to render the scene from a particular point of view. All Java 3D scene graphs must connect to a *Virtual Universe* object to be displayed. The *Virtual Universe* object provides grounding for the entire scene. A scene graph itself, however, starts with the *BranchGroup* (BG) nodes (although only one BG node in this case). A *BranchGroup* node serves as the root of a sub-graph, or branch graph, of the scene graph. The *TransformGroup* nodes inside a branch graph specify the position, orientation, and scale of the geometric objects in the virtual universe. Each geometric object consists of a *Geometry* object, an *Appearance* object, or both. The *Geometry* object describes the geometric

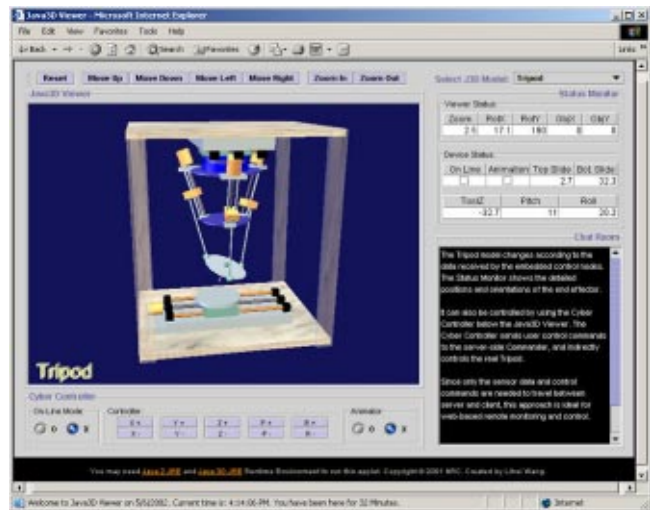


Fig. 6 Web-based remote monitoring and control of Tripod

shape of a 3D object. The *Appearance* object describes the appearance of the geometry (color, texture, material reflection characteristics, etc.). The behavior of the Tripod model is controlled by *Behavior* nodes, which contain user-defined control codes and state variables. Sensor data processing can be embedded into the codes for remote monitoring. Once applied to a *TransformGroup* node, the so-defined behavior control affects all the descending nodes. In this example, the movable objects (X-Table, Y-Table, and Moving Platform unit) are controlled by using three control nodes, for on-line monitoring/control and off-line simulation. As the Java 3D model is connected with its physical counterpart through the control nodes by low-volume message passing (real-time sensor signals and control commands, etc.), it becomes possible to remotely manipulate the real Tripod through its Java 3D model.

6.2 Kinematics Modeling for Tripod. Kinematics studies the geometric properties of the motion of points without regard to their masses or to the forces acting upon them. While the scene graph is the emergent standard hierarchical data structure for computer modeling of 3D worlds, kinematic models of physical devices or mechanisms that have external constraints or constraints that span interior nodes do not fit comfortably into its open-branched tree topology. In the case of our Tripod monitoring and control, models of both constrained kinematics and inverse kinematics are solved separately and embedded into the behavior control nodes in a scene graph to calculate the motions of respective components. Typically, constraints can be expressed in a number of equations or inequalities that describe the relationships among Tripod components. Based on sensor signals collected from the real Tripod, both constrained kinematic model and inverse kinematic model of the Tripod are needed to calculate the positions and orientations of the three sliding-legs and moving platform for 3D Tripod model rendering. A detailed description of the Tripod kinematic model can be found in [18].

6.3 Remote Monitoring and Control. Web-based remote device monitoring and control are conducted by using the *Status-Monitor* and *CyberController*, which communicate indirectly with the device controller through an application server. In the case of Tripod monitoring and control, they are further facilitated by the kinematic models derived in section 6.2, to reduce the amount of data traveling between Web browsers and the controller. The required position and orientations of the moving platform are converted into the joint coordinates by the inverse kinematics for both

client-side Java 3D model rendering and server-side device control. The three sliding-legs of the Tripod are driven by three 24V DC servomotors combined with three lead screws. Each actuator has a digital encoder (1.25 $\mu\text{m}/\text{count}$) for position feedback. The position data of the sliding-legs are multicast to the registered clients for remote monitoring, whereas only one user at one time is authorized to conduct remote control. A sampling rate of 1 kHz is used in the case study. Figure 6 shows a snapshot of the Web-based monitoring and control of the working Tripod.

7 Conclusions

This paper presents a Wise-ShopFloor framework and detailed three-tier architecture. The goal of our combined Web-based and sensor-driven approach is to reduce network traffic using Java 3D models for real-time applications, while still providing users with intuitive environments for conducting their work. Participating in the Wise-ShopFloor environment, users can not only feel reduced network traffic by real-time interactions, but also obtain more flexible and location-transparent control of their real shop floors. The Wise-ShopFloor framework enables a mobile solution for distributed manufacturing and frees engineers and shop managers from their dedicated computers. A Tripod case study demonstrates its feasibility and shows promise of this novel approach to distributed shop floor environments. The case study can be easily extended to a general case problem, as the server-side modules are designed for generic purpose in terms of data communication. The only unique issue to be addressed by the users is Java 3D model (and kinematic model, if any) creation. This can be achieved by using a third-party scene graph editor, similar to any CAD systems.

As decentralization of business increases, a large application potential of this research is anticipated. In addition to real-time monitoring and control, the technology can also be applied to collaborative design, remote inspection and trouble-shooting, as well as virtual manufacturing.

References

- [1] Caldwell, N.H.M., and Rodgers, P.A., 1998, "WebCADET: Facilitating Distributed Design Support," IEE Colloquium on Web-based Knowledge Servers, U.K., pp. 9/1–9/4.
- [2] Smith, C.S., and Wright, P.K., 1996, "CyberCut: A World Wide Web Based Design-to-Fabrication Tool," J. Manuf. Syst., **15**(6), pp. 432–442.
- [3] Adelson, B., 1999, "Developing Strategic Alliances: A Framework for Collaborative Negotiation in Design," Res. Eng. Des., **11**, pp. 133–144.
- [4] Cimplicity, GE Fanuc Automation. (<http://www.geindustrial.com/cwc/gefanuc/software.html>).
- [5] Waurzyniak, P., 2001, "Electronic Intelligence in Manufacturing," SME Manufacturing Engineering, **3**, pp. 44–67.
- [6] OpenCNC, MDSI, (<http://www.mdsi2.com/products/opencnc.htm>).
- [7] Flexlink, Hitachi Seiki, (<http://www.hitachiseikiusa.com/tradeshows/Westec/flexlink.html>).
- [8] Mazak Cyber Factory, Yamazaki Mazak Corporation, (<http://www.mazak.co.jp/English/SMT&IT/frame2-factory.html>).
- [9] CAPS-NET, Mori Seiki Co., Ltd., (<http://www.moriseiki.co.jp/english/index.html>).
- [10] e-Manufacturing Networks Inc., (<http://www.e-manufacturing.com/>).
- [11] Wang, L., Wong, B., Shen, W., and Lang, S., 2002, "Java 3D Enabled Cyber Workspace," Commun. ACM, **45**(11), pp. 45–49.
- [12] NCMS, 2001, "Factory-Floor Internet: Promising New Technology or Looming Security Disaster," Manufacturing in Depth, National Center for Manufacturing Sciences, USA.
- [13] Xi, F., Han, W., Verner, M., and Ross, A., 2001, "Development of a Sliding-leg Tripod as an Add-on Device for Manufacturing," Robotica, **19**(3), pp. 285–294.
- [14] Barrilleaux, J., 2001, 3D User Interfaces with Java 3D, Manning Publications Co., Greenwich, CT, USA.
- [15] Sowizral, H., Rushforth, K., and Deering, M., 2001, The Java 3D API Specification, 2nd Edition, Addison-Wesley, Boston, MA, USA.
- [16] OpenSceneGraph, (<http://www.openscenegraph.org/>).
- [17] OpenSG, (<http://www.opensg.org/>).
- [18] Wang, L., Sams, R., Verner, M., and Xi, F., 2002, "Web-Based and Sensor Driven Device Monitoring and Control Using Java 3D," Rob. Comput.-Integr. Manuf., **19**(1–2), pp. 13–19 (2003).

On STEP-NC and the Complexities of Product Data Integration

Martin Hardwick

Professor of Computer Science, RPI, Troy, NY 12180
President STEP Tools, Inc.

For twenty years the manufacturing domain has been seeking to share product model data by defining an entity relationship model covering the life cycle of geometrically defined products. The data sharing is implemented by selecting subsets of these entities and relationships to define data exchange standards for CAD, CAE, CAM, CNC and PDM systems. The approach requires agreement on how data will be reused across the domains so an organization has been meeting to manage the required data definitions. Considerable success was achieved in 1995 when a standard was proposed and implemented by industry as a way to move 3D geometry between CAD systems. Now, a new protocol for exchanging manufacturing process information between CAM systems and CNC devices is being released. This protocol reuses much of the data defined for the other domains and will allow CNC manufacturing tools to process 3D data, but the complexity of the specification is causing controversy. In this paper we report on the new specification and analyze the advantages and disadvantages of its approach to defining CNC control programs.

[DOI: 10.1115/1.1641188]

1 Introduction

Like many data sharing efforts [1], STEP, the Standard for Product Model data exchange, started with a burst of enthusiasm when a new technical capability was demonstrated. In its case the new capability was a specification called IGES (Initial Graphics Exchange Specification). IGES made it possible to move drawing data between CAD systems. Because of this initial success, industry decided it wanted to define standards to enable data exchange between systems across the complete life cycle of a product. The life cycle was defined to be all the information required from initial conceptual design to detailed design, to manufacturing, to maintenance and final disposal. The systems to be supported were to include all kinds of engineering design systems including Computer Aided Design (CAD), Computer Aided Engineering (CAE), Computer Aided Manufacturing (CAM), Computerized Numerical Control (CNC) and Product Data Management (PDM) systems [2].

STEP started in 1983 and grew into a large organization with over 500 people meeting four times a year at various locations in the USA, Europe and the Far East. Enthusiasm grew as a modeling language for defining and validating 3D geometry called EXPRESS was defined and software vendors started to support it. Geometry data can be complex with many relationships and much inheritance between entities. The overriding goal of EXPRESS and STEP was to define a language and model that could be used to define a Complete Unambiguous Product Model (or CUPM). Industry could relate to this goal because it implies that a supplier will be able to make a product completely and accurately. However, EXPRESS also became one of the first barriers between STEP and the wider community because while it is very comprehensive it is also harder to learn and fully implement. Unlike languages such as XML Schema and UML, EXPRESS is rarely used in other domains.

Contributed by the Engineering Informatics (EIX) Committee for publication in the JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING. Manuscript received August 2003; Revised November 2003; Associate Editor: R. Rangan.

Another barrier emerged when STEP adopted a “pre-planning” approach to data integration. In this approach data is integrated by mapping object models into a common set of integrated resources. These resources are planned up-front so that the different models can be integrated. The approach allows the STEP product models to grow over time and avoid islands of automation. However it also makes them harder to understand because programmers must understand the EXPRESS language, the object model and the mapping of the object model into the integrated resources.

Nevertheless the STEP developers persevered. An early success was achieved when a protocol called AP-203 was defined for exchanging 3D product model geometry between CAD systems. More than one million CAD stations now contain AP-203 translators. Other successes were achieved when protocols were defined for CAE, printed circuit boards, piping, building construction, shipbuilding and other domains but these had less impact because their applications have smaller numbers of users [3].

Now STEP is on the verge of a new success with the release of a specification for defining the data input to CNC controllers. The new specification is called STEP-NC and holds great promise because there are more than 500,000 machine tools with CNC controllers in the USA alone. Currently these controllers are driven by vector codes developed in the 1960's. STEP-NC will allow the data input to these controllers to be updated to 3D models annotated with design tolerances, manufacturing features, process sequence and cutting tool requirements [4,5]. As a result the time required for path planning may be reduced by 35%, the number of drawings required on the shop floor may be reduced by 75%, and for small job lots 50% faster machines can be used because they can do more checking in software [6].

However, the new specification is also very complex. It builds on twenty years of effort by the STEP development community. Some argue that a much simpler specification will get the same benefits and a rival standard has been developed as ISO 14649. This specification uses a simplified form of the EXPRESS language, makes the communication of 3D models to the CNC optional, and does not integrate data with the other STEP protocols. This paper tries to understand the issues by describing the STEP approach to integration. First we describe the STEP data integration methodology. Then we describe the arguments that have been occurring over the STEP-NC specification for CNC control. Finally we conclude with a discussion of the possible future directions for STEP.

2 The STEP Methodology

The STEP methodology for defining data exchange standards contains three principal components:

1. The EXPRESS language.
2. An architecture that maps the information requirements of an application into a set of integrated resources.
3. Implementation Methods for a variety of technologies and programming environments.

2.1 The EXPRESS Language. The STEP community invented the EXPRESS language to model 3D geometry. It was decided that a new language was desirable because of the extensive inheritance relationships that occur between geometric entities and because many mathematical rules can be defined to validate 3D geometry.

EXPRESS defines information models as schemas. Each schema contains entities first, and ancillary type, function and rule definitions second. An entity is the EXPRESS equivalent of a relation in a database or an element in XML. The ancillary type, functions and rules are defined to support the entity definitions. The simplest kind of ENTITY contains explicit attributes as shown below.

```
ENTITY point
  x   : REAL;
  y   : REAL;
END_ENTITY;
```

New types may be defined from other types and entities using the TYPE keyword. STEP recommended practice is to never use one of the EXPRESS pre-defined types in the definition of an entity because it is a missed opportunity to define a more specific role for the attribute. Therefore, in the next example a coordinate is defined to be a type of REAL and used to define the attributes of a point.

```
TYPE coordinate=REAL;
END_TYPE;
ENTITY point;
  X   : coordinate;
  Y   : coordinate;
END_ENTITY;
```

Some defined types need to be given more complex data structures. The structures available include five types of aggregates, a select type and an enumerated type.

```
TYPE axis2_placement=SELECT
  (axis2_placement_2d,
   axis2_placement_3d);
END_TYPE; -- axis2_placement
```

The five aggregate structures are LIST, LIST OF UNIQUE, BAG, SET and ARRAY. Each allows an attribute to be defined as an aggregation of another type. For example, the following code shows how STEP defines a point. This definition defines a Cartesian point to contain at least one length measure value and not more than three such values. We will not discuss why Cartesian point is defined in this way because this is a matter for geometry experts. The example is also the first to use the SUBTYPE keyword.

```
TYPE length_measure=REAL;
END_TYPE; -- length_measure
ENTITY cartesian_point
  SUBTYPE OF (point);
  coordinates : LIST [1:3] OF length_measure;
END_ENTITY; -- cartesian_point
```

The SUBTYPE keyword is followed by a list of the entities that are to be the super-types of the new entity. For example, the following code fragment defines a student nurse to be a subtype of nurse and student. This definition constrains every instance of student_nurse to have the type nurse and the type student as well as the type student_nurse.

```
ENTITY student_nurse
  SUBTYPE OF (nurse, student);
  rank      : STRING;
  seniority : INTEGER;
END_ENTITY;
```

In an EXPRESS data population, every entity instance has one or more types. Most data instances are defined by one type, but a significant number have several types, and in STEP these are often the most important entities in a model. All of the types must share a common super-type. Therefore, in our example student and nurse must both inherit from a common type such as Person. Provided this restriction is met instances can combine their types in any way not just the ways described by a SUBTYPE expression. The only exceptions are combinations prohibited by a SUPER-TYPE definition. The example below shows how a SUPER-TYPE definition can be used to constrain the types that can be in a person entity. The example defines a person to be a super-type of a driver or a pedestrian but not both. Also, the person instance

cannot exist on its own so it is defined to be an ABSTRACT SUPERTYPE.

ENTITY person

```
ABSTRACT SUPERTYPE OF (ONEOF (driver, pedestrian));
END_ENTITY;
```

The EXPRESS inheritance model was defined to allow continuous refinement over time. The initial STEP models defined a data exchange protocol for 3D geometry. The latest STEP models define a data input language for CNC control. In the time between different teams of experts have defined the information requirements of design tolerances, manufacturing features, manufacturing processes, manufacturing tooling and many other aspects of design and manufacturing.

The definition of manufacturing features shows why EXPRESS needs such a powerful model. A manufacturing feature can have both a parametric definition and a shape definition. The parametric definition describes the major properties such as a diameter and depth. The shape definition identifies the 3D surfaces that make up the hole. In many but not all CAD systems, the shape will be defined first by a designer as a cylinder (with one or two faces), then the hole will be defined when a planner decides how much material to remove in each drilling operation (there can be several). In other systems the planner may define the hole first in a design-by-feature operation and then geometry of the hole will be defined implicitly.

The requirement for strong rule definition in EXPRESS is also motivated by the geometrical content of product models. The rules can be simple such as a circle must have a positive radius, mildly complicated such as a loop of points describing a boundary must all be in the same plane, and very complicated such as the faces of a body must be topologically connected and closed. The definition of manufacturing features shows why EXPRESS needs such a powerful model. A manufacturing feature can have both a parametric definition and a shape definition as shown in Fig. 1.

The EXPRESS language has four levels of rules:

1. A rule is defined by the choice of data structure. For example, if a circle is required to have a radius and center then this can be determined by requiring both attributes in the definition of the entity.
2. A rule is defined by a local expression. For example, if there is no positive data type then a rule can be defined in the context of the circle by a simple expression such as "radius>0."
3. A rule is defined by a global expression. For example, if a data exchange protocol requires every circle to be associated with a plane (because our definition so far is 2D and this protocol is for 3D data) then a global rule can be defined that assures such an association exists for every instance of circle.
4. A rule is defined by an informal proposition (English language description). This is the least desirable alternative because of possible misinterpretation particularly in an International community but necessary because some geometric properties such as the requirement for a solid to be closed cannot be reduced to a finite algorithm.

Each level is harder to understand. The Level 3 language in particular hurt the popularity of EXPRESS because it requires predicate calculus style functions to search over an information base for the existence of entities. This style of function is not familiar to engineers. Giving the calculus functions semi-familiar sounding names was a compromise that suited nobody. A language called EXPRESS-X overcomes this problem by giving EXPRESS an SQL-like ability to query an information base but it is not widely used.

2.2 The STEP Data Integration Architecture. The most important and controversial component of STEP is its use of mapping tables and integrated resources to integrate data across appli-

cation domains. The idea is to allow STEP to grow over time to include more information about the product life cycle. A list of the data exchange standards currently defined for STEP is given in the Appendix. Each standard is called an Application Protocol (AP) to distinguish it from other internal standards that define infrastructure such as the EXPRESS language and the integrated resources.

The STEP data integration architecture requires a STEP Application Protocol to be developed in two phases. In the first phase an Application Requirement Model (ARM) is developed using an information modeling language. Today the language is usually EXPRESS, but in the early days languages such as IDEF1X and NIAM were also used. The next stage maps the ARM model into the integrated resources using mapping tables to create the Application Interpreted Model (AIM). The result is a much more robust model. However, the following aspects of this process cause controversy:

- The mapping tables are hard to understand. The notation used in the tables is mathematical but not complete because executing it against the ARM model will not produce the AIM model. This is because the AIM model contains more information than the ARM model (see the third bullet).

- The integrated resource models are normalized. This allows them to be expanded without affecting the existing AP's but it also makes the AIM models more difficult for applications to navigate.

- The information definitions are expanded as part of the mapping. For example, a simple definition in the ARM such as "shape" or "tolerance" is expanded in the AIM to the full definition computed by geometry or tolerance experts.

The definition of tolerances for the STEP-NC ARM model illustrates the methodology. In an ARM model a tolerance can be represented as a simple value that indicates the desire to use tolerances in a model without giving a detailed definition.

```
TYPE toleranced_length_measure=length_measure;
END_TYPE;
ENTITY round_hole
SUBTYPE OF (machining_feature);
  diameter: toleranced_length_measure;
  change_in_diameter: OPTIONAL taper_select;
  bottom_condition: hole_bottom_condition;
END_ENTITY;
```

When round_hole is mapped into the integrated resources it is necessary to describe a full definition for tolerances. There are at least two weaknesses. First there is no definition for the upper and lower limits for the tolerance value for the diameter. This is relatively easily fixed but the full definition is surprisingly complicated because of the wide range of techniques used to define tolerances by engineers. Second, there is no definition for the tolerance for the position of the hole. This is more difficult because the location of the hole is not shown in the definition. Instead it is inherited from one of the super-types. Such "missing values" occur quite frequently in ARM models because the developers are considering the information requirements of an application from one perspective only. In this case the machining perspective. From this perspective it is desirable to model the location of each feature in a common super-type with coordinates in the center of the feature and at the top of the stock because this is how the machinist will measure the feature when the part is manufactured.

In the integrated resources a positional tolerance is modeled with respect to three datum faces. These faces are likely part of the model and probably define another manufacturing feature as shown in Fig. 2. However, it is also quite possible for them to belong to another model such as a fixture. Plus each kind of feature needs its own kind of tolerances making it difficult for common machining properties such as the axis to be inherited from a common super-type. One of the advantages of the STEP architec-

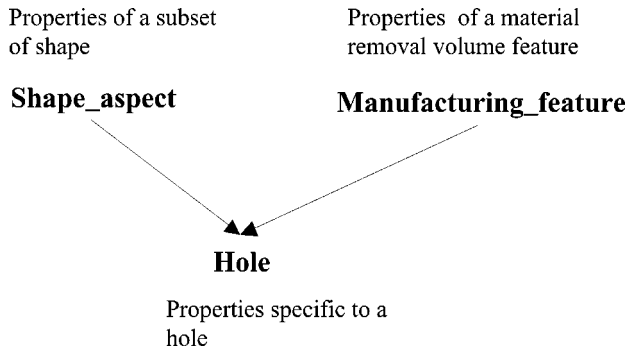


Fig. 1 EXPRESS Inheritance Example

ture is that it allows a team to define a detailed model of their unique requirements and make light reference to other information already known to be in STEP. This is widely accepted when that other information is geometry, but more controversial when the other resource is tolerance or PDM data.

2.3 Implementation Methods. Each STEP information model is implemented by mapping the integrated definition of the model into an implementation technology.

The available implementation technologies include:

- An EXPRESS driven file format called Part 21.
- A variety of programming language bindings that allow an application programmer to open EXPRESS defined information sets and access values in entity instances. Bindings have been developed for C, C++ and Java.
- A new configuration language for mapping EXPRESS into an XML Schema.

The Part 21 file format is currently the most popular implementation method. The programming language implementation methods have been the inspiration for a number of tool kit products.

The XML implementation method is new and interesting because it may help bridge the gap between the ARM and AIM models of STEP.

The XML binding is known as Part 28. There have been two editions. The first edition defined three algorithms for mapping EXPRESS defined data into XML data defined by DTD's. Each algorithm selected a different subset of the available information and mapped it into an XML form without much concern for the resulting organization. The idea was that the XSLT language would be used to map the information into a desired organization. However, the STEP information models are large and the XSLT mappings became very difficult for third parties to understand so the documentation of the STEP meaning of the XML became unclear. As a result it was decided to try again using a language to configure the STEP information into XML defined by XML Schema.

The new language lets an information modeler annotate an EXPRESS schema for the purpose of developing an XML Schema. The language lets the information modeler configure three qualities:

1. Determine the owner element and child element in nesting relationships.
2. Pick a tag-name for each element from the available EXPRESS names.
3. Pick a layout format for aggregate data structures.

The advantage of the configuration language is that it can produce STEP XML data that is easier to read as the code fragment from AP-203 in Fig. 3 perhaps demonstrates.

The configuration language lets an information modeler or application developer tailor the description of the XML for an application either to make the data easier to read or easier to process. However, the configuration may also create incompatibilities between models because one modeler may decide that X should be the parent of Y while the other decides the opposite. This negates some of the value of the STEP integration architecture because code written for one model will not be applicable to the other

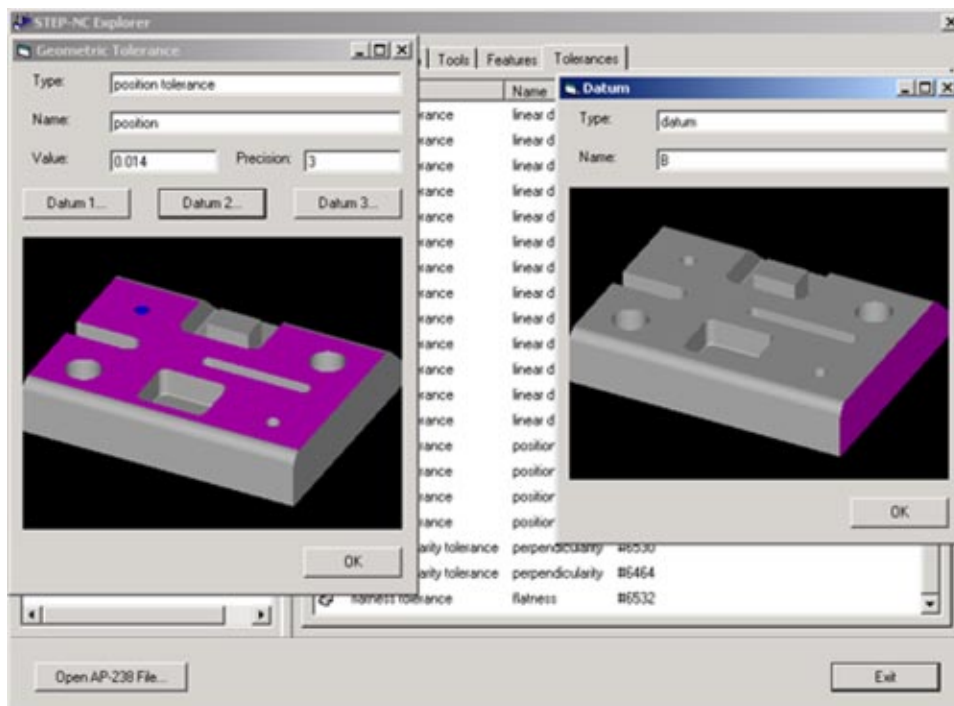


Fig. 2 A hole position tolerance with two of its three datums shown

EXPRESS (Simplified AP-203)	Part 21 (first) XML after configuration (second)
<pre> ENTITY product; id : identifier; name : label; END_ENTITY; -- product ENTITY person_and_organization_assignment ABSTRACT SUPERTYPE; assigned_person_and_organization : person_and_organization; role : person_and_organization_role; END_ENTITY; ENTITY cc_design_person_and_organization_assignment SUBTYPE OF (person_and_organization_assignment); items : SET [1:?] OF product; END_ENTITY; ENTITY person_and_organization; the_person : person; the_organization : organization; END_ENTITY; ENTITY person; last_name : OPTIONAL label; first_name : OPTIONAL label; END_ENTITY; ENTITY organization; name : label; description : text; END_ENTITY; </pre>	<pre> -- Part 21 #34=PRODUCT('prod', 'A Product'); #35=CC_DESIGN_PERSON_AND_ORGANIZATION_ASSIGNMENT(#36, #37, (#34)); #36=PERSON_AND_ORGANIZATION(#38, #39); #37=ROLE('design_supplier'); #38=PERSON('Smith', 'John'); #39=ORGANIZATION('WidgetsUnlimited','Stuff Made'); -- XML after configuration <product id="j34"> <id>prod</id> <name>A Product</name> <design_supplier> <last_name>Smith</last_name> <first_name>John</first_name> <organization>WidgetsUnlimited</organization> <org_type>Stuff made</org_type> </design_supplier> </product> </pre>

Fig. 3 STEP data in XML as defined by Part 28 Edition 2

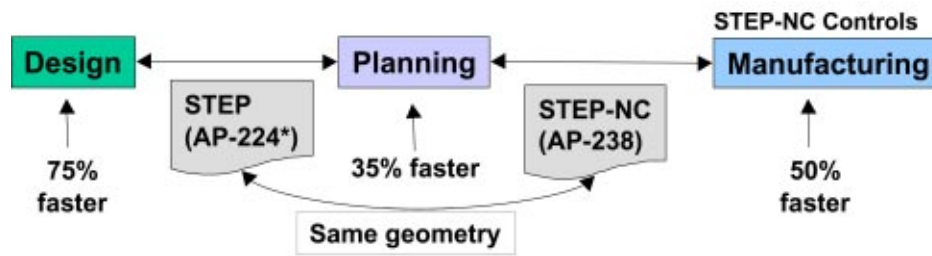
model unless the processor understands the configuration language. It remains to be seen whether this will be acceptable to the STEP community.

3 STEP Integration and STEP-NC

The STEP-NC model defines a CNC part program as a series of operations that remove material defined by features. The features supported include holes, slots, pockets and removal volumes defined by 3D surfaces. Each operation contributes to the manufacture of a feature by defining the volume of material to be removed, the type of tool required and some basic characteristics such as whether this is a roughing or finishing operation. The operations are then sequenced into a work plan that converts the stock into the final part. The work plan may be sophisticated and include conditional operations that depend on the results of probing operations, and it may be divided into sub-plans to be executed concurrently on machines that have multiple cutting heads.

Whether as an object model or as an Application Protocol, STEP-NC is intended to replace an existing language for machine control known as RS274D in the USA and ISO 6983 in the rest of the world. In RS274D a part program is described as a sequence of linear and circular tool movements. There is no information about the part being machined, the tool requirements, or the features in the data so the only strategy that a machine tool controller can use to make the part is to execute each instruction as accurately as possible. This has led to a machining environment where many machine tools quite literally resemble dinosaurs because they compensate for their small brains by being as big as possible so that they will be as rigid as possible when they execute the G code instructions. It has also ensured that manufacturing continues to rely on drawings because people cannot understand the intent of RS274D files.

Figure 4 shows how the design to manufacturing process can be implemented using STEP-NC. Design creates the specification for



*224 is best, 214 next best, 203 acceptable provided process planning includes feature recognition.

Fig. 4 STEP-NC defines a new interface between Planning and Manufacturing

Table 1 Summary of differences between the STEP-NC models

	STEP Compliance	EXPRESS compliance	3D Geometry	Design Integration	Complexity
AP-238	Full	Full	Required	Full	More
ISO 14649	Partial	Partial	Optional	Little	Less

a product. Planning decides how to manufacture the product. Manufacturing controls production. The job of design can be assisted using a CAD (Computer Aided Design) system. The job of planning can be assisted using a CAM (Computer Aided Manufacturing) system that may or may not be part of the same integrated system as the CAD system. The job of manufacturing is usually controlled using a CNC (Computerized Numerical Control) system. STEP-NC allows the systems to be given new interfaces so that full fidelity 3-D geometry annotated with all the necessary manufacturing information is sent into the manufacturing control.

Sending full fidelity 3D data into the controls has three fundamental advantages for manufacturing industry:

1. It makes developing a CNC part program more efficient because the machining instructions can be defined more concisely. (In 3-D a command can be "rough mill this pocket" instead of move the tool to this location, then here, then here etc).
2. It allows a CNC to optimize and check a part program for the tooling available at the time of manufacturing instead of having it fixed at the time of planning so manufacturing can become more efficient, safe and flexible. (With 3-D data the control can analyze what is possible).
3. It eliminates the requirement for drawings on the shop floor because a STEP-NC file describes both the process and the part including all the required tolerances. (An intelligent browser can give the operator more information than a drawing).

However, there are two STEP-NC models: the ARM model developed by ISO subcommittee TC184/SC1 as ISO 14649 and the AIM model developed by ISO subcommittee TC184/SC4 as AP-238. The ARM model reuses the STEP geometry models and is harmonized somewhat with the STEP feature model, but it has its own model for tolerances and PDM.

The SC1 team started STEP-NC and was attracted to the EXPRESS language because CNC control is a complex domain requiring complex data definitions and STEP has a definition for NURB surfaces. However, they decided to not use the full inheritance model of EXPRESS and made only light use of the rule language. Originally they did not call their new object model STEP-NC either, but found the models popularity increased considerably when they adopted this name. However, they never planned to use STEP integration and this is now a big issue between the teams because the SC4 team can claim that SC1 has adopted its "trademark" without following the conventions that give the trademark meaning.

The SC1 team argues that it should not have to use integration for the following reasons:

- a. The object model describes data that is easier for a person to read and hand-edit using a text editor.
- b. The object model describes data that can be parsed more quickly which is an important advantage for high-speed machining.
- c. The SC4 integrated resources were not designed for manufacturing.

The SC4 team counter argues as follows:

- a. Both models describe data that is too difficult for an average machine tool operator to hand edit so there will have to be

graphical interfaces on the CNC. The integrated STEP model makes these interfaces more powerful because each working step can be shown in the context of the part feature it manufactures, the current geometry of the part and the tolerances required by that geometry. In the ARM model these features, geometry and tolerances are not available so they cannot be shown.

- b. High Speed CNC's already make extensive use of caching. If the integrated model is slower to parse then this can be overcome using another level of caching.
- c. The SC4 integrated resources are normalized to make them easily extendible. If specific weaknesses can be identified then they should be extended for manufacturing. However, the editors of the STEP-Manufacturing Application Protocols (AP-224, AP-219 and AP-240 in addition to AP-238) have not yet identified any weaknesses.

The difference between the models is illustrated most clearly by the link between features, geometry and tolerances. In the SC4 model, the tolerance data is defined by the GD&T model developed for AP-203 Edition 2, AP-214 and AP-224 (see Appendix for the title of these AP's). This allows an application program to traverse the data from a feature, to the faces in that feature, to the design tolerances that apply to those faces, to the datums that define the tolerances, to the plane that defines each datum, to another feature that contains a face on that datum plane and so on.

The SC4 team argues that the differences (summarized in Table 1) matter because they affect the fundamental business benefits. If the STEP-NC model includes the STEP tolerance model then there will be greater traceability between design and manufacturing. Similarly, if the STEP-NC model uses the STEP model for manufacturing features then CNC programming systems will be able to receive these features from design or manufacturing. Thirdly, if the CNC machine tool receives the design product model then there are many quality checks that can be performed on the machine tool such as determining if the selected tool and speeds and feeds will produce the right surface finish.

4 Current Status and Future Directions

STEP defines a large Entity Relationship model for product data and then uses subsets of that model to exchange data between applications. The main advantage of this approach is that data is reused across application domains. The main drawback is that a lot of effort is necessary to get consensus on which entities are necessary for each domain. This can make the standards seem more complex than necessary.

Everyone agrees that AP-238 is more functional than ISO 14649. Therefore, it would be better if industry implemented AP-238 but SC1 argues that it will not because of the extra complexity. Over the years STEP has learned the following about deployment of its Application Protocols:

- The protocol must deliver useful (non-trivial) data from a sending system to a receiving system.
- The protocol must describe its data in a manner that lets algorithms in the receiving system process that data deterministically (protocols that *require* attributes are easier to process than those that allow attributes to be *optional*).
- The protocol must have well defined rules to distinguish

between good data and bad data so that systems that send bad data can be identified (bad receiving systems are self-evident).

- The protocol must have at least one conformance class (i.e. subset defined in the standard) that is easy to implement so that early success can be demonstrated to management.

AP-238 is an order of magnitude more complex than AP-203 because it contains both design and manufacturing information. Therefore the fourth rule may be the key to implementing STEP-NC as AP-238.

In AP-203 the simple conformance class allowed 3D models to be exchanged as facets. Moving data between CAD systems in this way is not very useful because such data can be moved using VRML and STL. However, the value of the simple conformance class was that it allowed the CAD vendors and users to show early success to management. After this was achieved permission was given to tackle the more difficult problem of moving full fidelity 3D information between systems.

AP-238 deployment can use the same approach by defining a conformance class for tool paths. Moving data in this way is also not novel because RS274D already defines tool paths as M and G codes. However, value will be added because the tool paths can be associated to features and design data so more powerful editing and tracking will be possible. Plus the technology to convert the tool paths back to M and G codes is not difficult so there is much less risk for the early end user.

After the initial Conformance Class has been implemented then the community can progress into the more complex feature driven conformance classes. Gradually considering all the interactions that can occur between the different kinds of features until any kind of feature based CNC model can be passed between the systems. The same activity occurred for AP-203 after the initial success with faceted models. In its case the range of geometric representations was gradually increased until all reasonable and accurately defined geometry could be passed between CAD systems.

If STEP-NC is successful then there will be several more years of STEP development as STEP-NC models are created for many different CNC controlled manufacturing processes. After that the path for STEP is not so clear. Some would argue that for STEP to

grow further it must adopt an XML file format to make its data more understandable. The Part 28 Edition 2 specification is showing the way but it is also shows that the price of making the data more understandable may be a return to islands of automation.

Others argue that before STEP can be applied to larger problems it must become more modular [7]. In this proposal Application Protocols are rapidly constructed by users from a library of modules that each implement a unit of functionality. Today these units of functionality are developed by the STEP modelers for each AP as part of defining the ARM model. Teams within STEP then ensure compatibility between the units by identifying commonalities. The new proposal is to develop the units in advance as both ARM and AIM models and then assemble them as necessary. It remains to be seen if the module developers can usefully anticipate the requirements of the AP developers in advance.

Another promising line of research is to combine aspects of the modules initiative and the XML implementation method. Early results are showing that STEP data can be divided into many small fragments each with an XML Schema definition and an RDF catalog description. The Application Protocol then assures all of the fragments add up to a valid model for processing on a tool such as a CNC. In this approach the XML and RDF can be distributed across multiple directories and organizations leading to some interesting search and integration applications. For example, Organization A might define and maintain the data for operation 1, Organization B might do the same for operation 2, and Organization C might maintain the design and final assembly data. A search engine might then test this data and create the associations required to define a complete product.

It remains to be seen what will happen next. Soon after the successful definition of AP-203 it was asserted that STEP could not be extended into CAM without defining new methods for representing processes. This was not the case. Now there are questions over whether STEP can be extended into the maintenance phase of the life cycle without new additions. What they might be and whether they will prove to be desirable or necessary remains to be seen. It is undeniable however that as STEP grows to include more functionality it will become more complex, but with the new XML implementation methods it may become easier to hide this complexity from beginners.

Appendix—STEP Application Protocols

Part 201 Explicit Drafting
Part 202 Associative Drafting
Part 203 Configuration Controlled Design
Part 204 Mechanical Design Using Boundary Representation
Part 205 Mechanical Design Using Surface Representation
Part 206 Mechanical Design Using Wireframe Representation
Part 207 Sheet Metal Dies and Blocks
Part 208 life Cycle Product Change Process
Part 209 Design Through Analysis of Composite and Metallic Structures
Part 210 Electronic Printed Circuit Assembly, Design and Manufacturing
Part 211 Electronics Test Diagnostics and Remanufacture
Part 212 Electrotechnical Plants
Part 213 Numerical Control Process Plans for Machined Parts
Part 214 Core Data for Automotive Mechanical Design Process
Part 215 Ship Arrangement
Part 216 ship Molded Forms
Part 217 Ship Piping
Part 218 Ship Structures
Part 219 Dimensional Inspection Process Planning for CMMs

Part 220 Printed Circuit Assembly Manufacturing Planning
Part 221 Functional Data and Schematic Representation for Process Plans
Part 222 Design Engineering to Manufacturing for Composite Structures
Part 223 Exchange of Design and Manufacturing DPD for Composites
Part 224 Mechanical Product Definition for Process Planning
Part 225 Structural Building Elements Using Explicit Shape Rep
Part 226 Shipbuilding Mechanical Systems
Part 227 Plant Spatial Configuration
Part 228 Building Services
Part 229 Design and Manufacturing Information for Forged Parts
Part 230 Building Structure frame steelwork
Part 231 Process Engineering Data
Part 232 Technical Data Packaging
Part 233 Systems Engineering Data Representation
Part 234 Ship Operational logs, records and messages
Part 235 Materials Information for products
Part 236 Furniture product and project
Part 237 Computational Fluid Dynamics
Part 238 Integrated CNC Machining
Part 239 Product Life Cycle Support

References

- [1] Hardwick, M., Spooner, D. L., Morris, K. C., and Rando, T., February 1996, "Sharing Manufacturing Information in Virtual Enterprises," *Commun. ACM*, **39**(2), pp. 46–54.
- [2] ISO 10303-1, 1994, "Industrial Automation Systems and Integration Product Data Representation and exchange—Overview and Fundamental Principles," International Standard, ISO TC184/SC4, Geneva Switzerland.
- [3] Brunnermeier, S. B., and Martin, S. A., March 1999, "Interoperability Cost Analysis of the U.S. Automotive Supply Chain," Research Triangle Institute, Research Triangle Park, NC.
- [4] Suh, S. H., Cho, J. H., and Hong, H. D., January 2002, "On the Architecture of Intelligent STEP-Compliant CNC," *International Journal of Computer Integrated Manufacturing*, **15**(2), pp. 168–177.
- [5] ISO 14649-1, 2001, "Industrial Automation Systems and Integration Physical Device Control-Part 1: Overview and Fundamental Principles," Draft International Standard, ISO TC184/SC1, Geneva, Switzerland.
- [6] Hardwick, M., 2003, "Digital Manufacturing using STEP-NC," *Machining Technology*, **14**(1).
- [7] Barnard, Feeney A., 2002, "The STEP Modular Architecture," *J. Comput. Inf. Sci. Eng.*, **2**(2).

A Web-based Product Structure Manager to Support Collaborative Assembly Modeling¹

Li Chen,² Tingjin Wang, and Zhijie Song

The University of Toronto, Design and Manufacturing Integration Laboratory, Department of Mechanical and Industrial Engineering, 5 King's College Road, Toronto, ON, Canada M5S 3G8
e-mail: chen@mie.utoronto.ca

Collaborative CAD systems enabling collaboration in computer-aided design processes among distributed designers are gaining more and more attention. Yet, such systems, especially in support of collaborative assembly modeling, are hardly achievable. In an effort to bridge this gap, we are dedicated to developing a collaborative CAD system with aim at 3D assembly modeling. As part of this effort, this paper addresses one function module of the system, a Web-based Product Structure Manager, which enables the Collaborative Product Structure Management (CPSM) in collaborative assembly modeling. In particular, CPSM facilitates product data sharing among distributed designers and supports collaboration in product structure creation and modification. A bench clamp assembly is used as an example to illustrate the Product Structure Manager for supporting collaborative assembly modeling. [DOI: 10.1115/1.1666894]

Keywords: Product Structure, Product Data Management, Collaborative Assembly Modeling, Collaborative CAD, STEP

1 Introduction

Usually real-life product design is carried out by a group of designers based on teamwork. However, most of the available CAD software only supports solitary design activities of a single

designer, for example, in product assembly modeling. That is, it is only one designer that is allowed to build a 3D assembly model on a single computer. In a team design scenario, this will inevitably incur more design iterations with more CAD data exchanges among designers (or computers) and, oftentimes, between different CAD systems. As a result, it adds to the difficulties in identifying early-stage design conflicts and thereby prolongs product design cycle.

To cope with this challenge, collaborative CAD systems enabling collaboration in computer-aided design processes appear appealing and have gained growing attention. Unlike conventional CAD systems, a collaborative assembly modeling system is such a collaborative CAD system that allows a group of collaborating designers, often geographically dispersed, to work concurrently and seamlessly on product assembly modeling in a distributed computing environment. With the aid of such a system, assembly-induced design conflicts arising from the outsourcing of design activities can be identified and resolved in the early stages of team-based design.

Unfortunately, successful development of a collaborative assembly modeling system is seldom reported in the literature. Nevertheless, there still exist two parallel types of collaborative CAD system, though very few, that somewhat support collaborative design activities, namely, CAD conference systems and Internet-based CAD systems.

CAD conference systems enable designers to hold virtual design meetings via the Internet [1,2]. In such systems, audio and video communications are two typical modes to support interactions among designers. These systems also provide a shared "whiteboard" that allows designers to collaboratively view and annotate 3D objects. Although they support collaborative reviews of 3D models, the systems do not lend themselves to real-time CAD modeling activities. On the other hand, Internet-based CAD systems have been carefully designed to enable designers to access a feature-based 3D modeling system via the Internet [3–5]. Although such Internet-based CAD systems make it possible for distributed designers to share a feature-based CAD system, the issue of real-time multi-user interaction has not been addressed in depth.

To this end, our research effort is underway toward an Internet-enabled collaborative CAD system dedicated to platform-independent 3D assembly modeling [6,7]. The main goal is to provide a team design environment enabling a group of designers to collaboratively build an assembly model in real time. As part of this effort, this paper addresses one function module of the system, a Web-based Product Structure Manager, that enables the Collaborative Product Structure Management (CPSM) in collaborative assembly modeling. In particular, CPSM facilitates product data sharing among distributed designers and supports collaboration in product structure creation and modification.

Two specific considerations have been incorporated in developing the Product Structure Manager [6]. That is,

- *Heterogeneous application systems and data formats are being used by designers;*
- *Designers are situated in different areas of the world.*

In this context, STEP standard [8] has accordingly utilized to address the first issue; meanwhile, the Product Structure Manager has been settled as a "Web-based" function module to accommodate the second issue.

The rest of this paper is organized as follows. Section 2 describes basic concepts and components in the framework of CPSM. Sections 3 and 4 elaborate the activity and data models of CPSM, respectively, that are built upon IDEF0 technique and STEP standard. In Section 5, a prototype of the Product Structure Manager is addressed with emphasis on 3-tier client-server architecture and function implementation. Section 6 illustrates the ap-

¹Modified from the paper presented in the 2003 ASME/DETC-CIE (Paper No. DETC2003/CIE-48265)

²Corresponding author.

Contributed by the Engineering Informatics (EIX) Committee for publication in the JOURNAL OF COMPUTING AND INFORMATION SCIENCE IN ENGINEERING. Manuscript received September 2003; Revised December 2003. Associate Editor: S. Szykman.

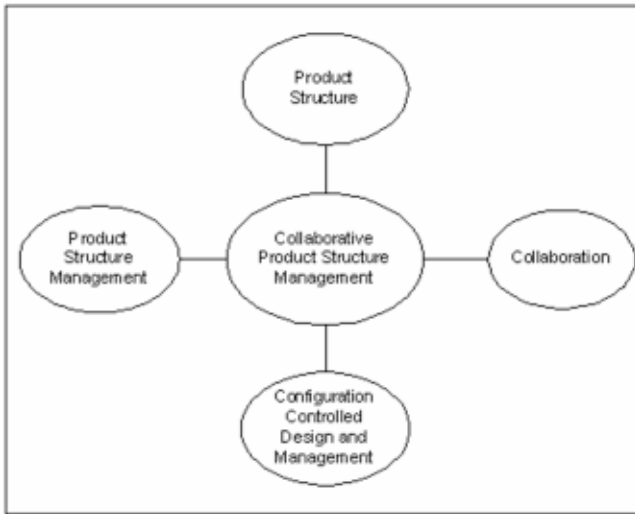


Fig. 1 A framework of the CPSM method

plication of the Product Structure Manager, through a design example, to a collaborative assembly modeling scenario. The paper is concluded in Section 7.

2 Basic Concepts and Components

Collaborative Product Structure Management (CPSM) refers to an approach and means to collaborative computerization of product data in support of collaborative assembly modeling. With it, distributed designers can collaboratively create and modify the product structure to form design variants synergistically. At this point, the Web-based CPSM system should make it achievable to seamlessly share and exchange product data in a worldwide scope with ease and efficiency.

CPSM involves activities relating to many aspects (or subjects), which form a framework of CPSM as shown in Fig. 1. This framework also provides a basis for constructing both activity and data models of CPSM. The basic components/concepts involved in this framework are described below.

2.1 Product Structure. Product Structure is a hierarchical breakdown of a product. It is focused on the aspects of product design that define a product in terms of nested decomposition of its components. Each component has data, known as meta-data, to describe it, and can also be associated with some documents such as CAD/CAE files. Figure 2 illustrates an example of the product structure of an office chair together with its meta-data and associated documentation [9]. Therefore, except for the meta-data of a product structure, product definition data, such as 3D models, 2D drawings or process planning documentation of parts and assemblies, are linked to each component in a product structure. This linkage provides a rapid and efficient way for locating design information and tracking design changes (versions and variations).

2.2 Product Structure Management. Product Structure Management (PSM) is a term describing the activities needed to handle a product structure. Since a product structure is part of product information in the entire product development process, its studies have received close attention recently and the results have been integrated into state-of-the-art enterprise information systems such as PDM systems [10,11]. According to CIMdata [10], PSM facilitates the design configuration and management. As configuration changes over time for a product, the PSM systems can track design changes, versions, affectivities or evolutions. In short, the PSM serves as activities to

- Facilitate creating and managing product configuration;
- Track versions, affectivities and design variations;

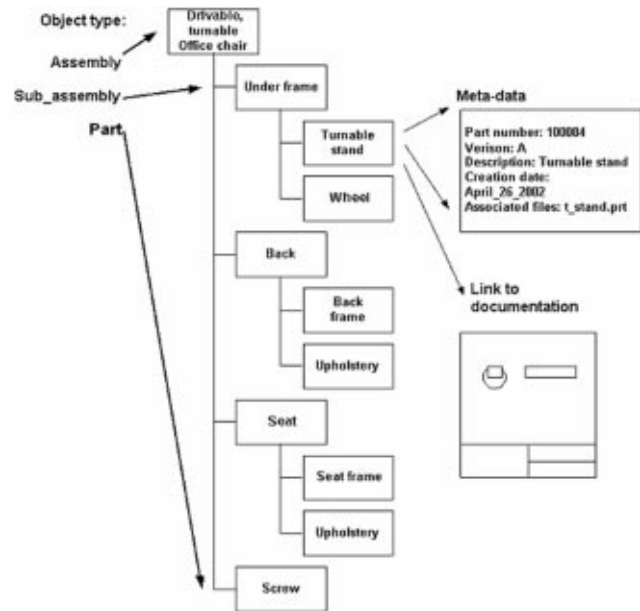


Fig. 2 Product structure of an office chair

- Link product definition data to a product structure;
- Allow various domain-specific views of a product structure; and
- Transfer product structure and other data in both directions between the PDM and ERP systems.

2.3 Configuration Controlled Design and Management

The product structure forms a basis to support configuration-controlled design and management tasks. Configuration is a process of selecting and maintaining a set of components and their relationships, which is contributive to forming a design solution [11,12]. The primary task of configuration management is to ensure that a complex product, when designed from detailed specifications, is realized as intended. The configuration management involves four main activities: configuration identification, configuration change control, configuration status audit, and configuration review. It is therefore a widespread process involving intensive information and extensive activities. In this context, this work is concerned with only the data relating to design and management of 3D product configurations in a product family.

2.4 Collaborative Product Structure Management. Collaboration applies to a process in which a group of collaborators work together to synergistically achieve a common goal. The notion of collaborator is context dependent and can be referred to an individual, a team or an organization. Oftentimes, the collaborators are geographically dispersed. Nowadays, the collaborative product development approach in team paradigm has been widely encouraged and adopted in industry. Essentially, there are two main functions that must be supported in a computerized collaborative design process:

- Document Sharing-Collaborators can view and edit documents that are stored in a common space but may originate from multiple geographically dispersed systems. These documents can be dynamically updated in real time.
- System Sharing-Collaborators can share a common system to create or modify product data. For example, sharing a PSM system enables collaborators to access available functions of the same system to manipulate product structures.

Note that embodiment of the above collaborative functions should depend on domain-specific applications. Therefore, in application

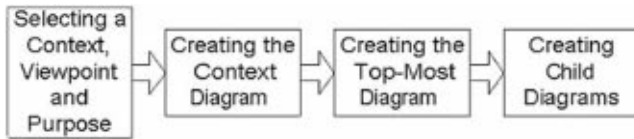


Fig. 3 Steps in process modeling

to the collaborative assembly modeling aided by CPSM, these functions are further specified as to fulfill the requirements of CPSM on top of the traditional PSM, namely,

- Product Data Sharing-Product data that are generated on heterogeneous CAD platforms can be integrated and shared among collaborators; and
- PSM System Sharing-The PSM system is open to each collaborator, enabling collaborators to perform the PSM activities synchronously and interactively.

To this end, CPSM is defined as a collaborative process for the Product Structure Management among/across a group of collaborating participants. In fact, CPSM may be viewed as the result of combining collaboration features into the traditional PSM.

3 CPSM Activity Model

In this section, IDEF0³ technique is used to create the activity model of the Product Structure Manager in three steps according to Fig. 3.

STEP 1-Creating a Context Diagram. To begin with, the context of the Product Structure Manager activities is prescribed in terms of purpose and viewpoint, thus profiling the development of the activity model. Figure 4 shows the context diagram of CPSM, denoted A-0, in which the single box defines the overall function that covers an entire scope of the functionality of the Product

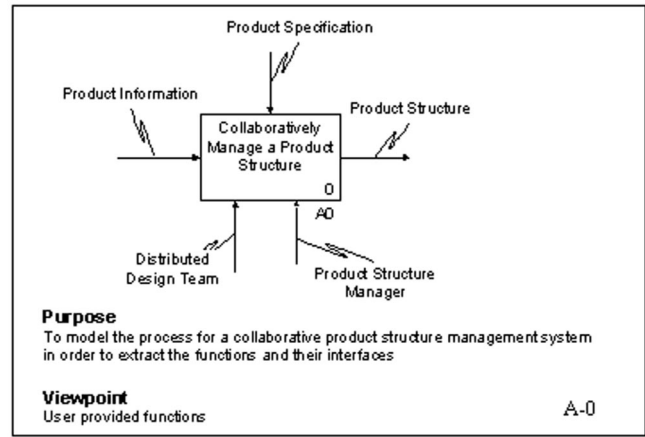


Fig. 4 Context diagram of CPSM

Structure Manager. Specifically, the function, named “Collaboratively Manage a Product Structure,” forms the context of the activity model, together with another five function components representing the data and object interfaces of the system to its environment: “Product Information,” “Product Structure,” “Product Specification,” “Distributed Design Team,” and “Product Structure Manager.” As indicated in the diagram, this model purports to model the CPSM process to extract the functions and their interfaces for a CPSM system (Product Structure Manager) from a system user’s viewpoint.

STEP 2-Creating the Topmost Diagram. The context diagram bounds the context of the Product Structure Manager. To embody the overall function defined in A-0 diagram, the diagram in Fig. 5, denoted A0, shows how the function in Fig. 4 can be decomposed into four major sub-functions, denoted A1 through A4, as follows:

³Information about IDEF0 standard is available on <http://www.idef.com>.

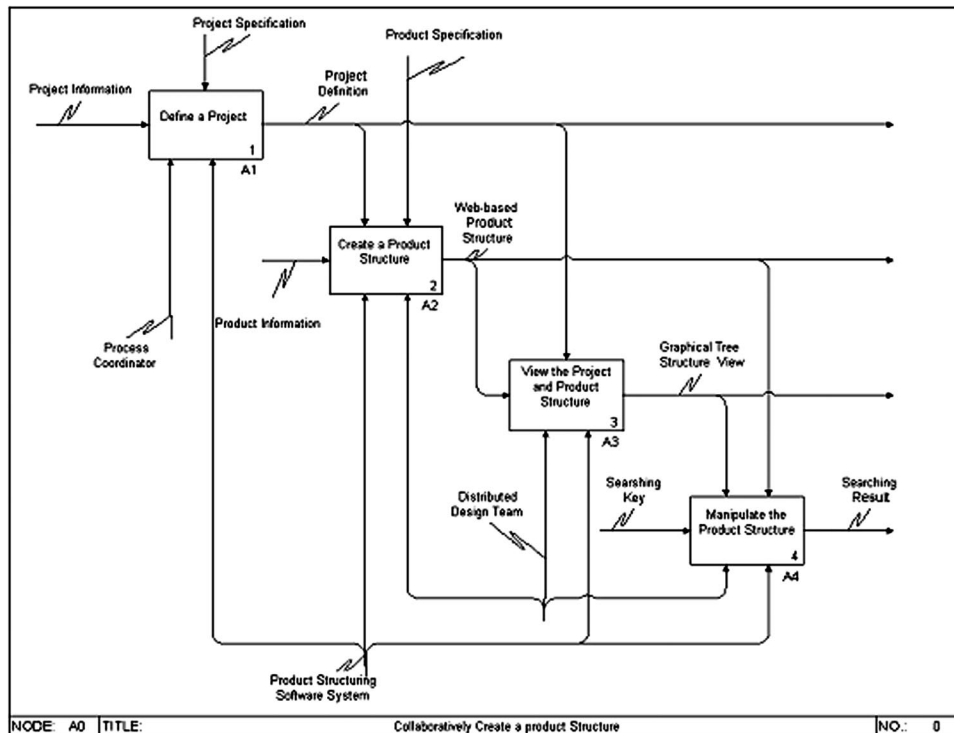


Fig. 5 Topmost diagram of CPSM

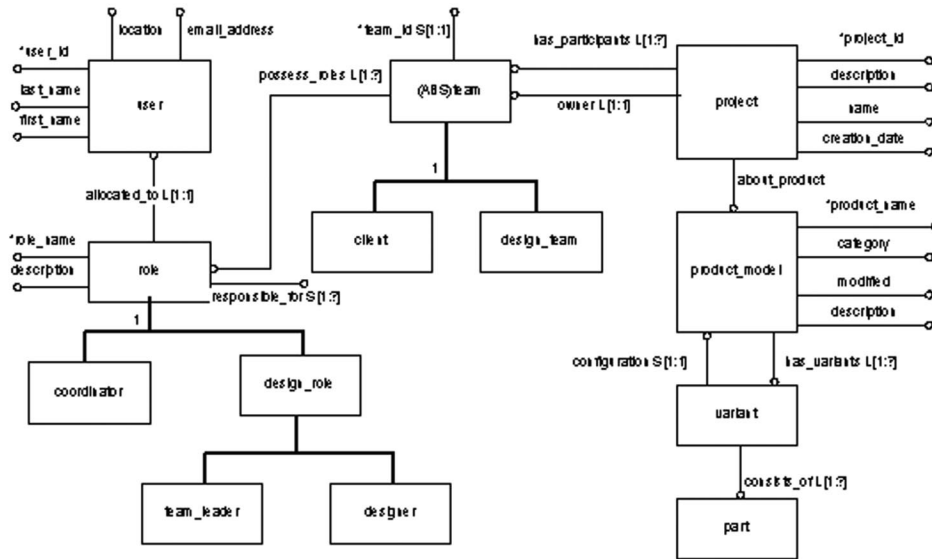


Fig. 6 EXPRESS-G Model of project management UoF

- *Define a Project (A1):* A live session is started where participants initiate collaboration in product structure related activities.
- *Create a Product Structure (A2):* Participants, with assigned privileges, collaboratively create a product structure across a family of products.
- *View the Project and Product Structure (A3):* The resulting project and product structure information are made available and accessible to each participant.
- *Manipulate the Product Structure (A4):* Participants can track the product configurations and access the definition data of each component through an index.

STEP 3-Creating Child Diagram. To elaborate the functions of the Product Structure Manager, decomposition of each function (within the A0 diagram) into more child diagrams is performed subject to the IDEF0 modeling protocols. This process continues until the content of each child diagram reaches the level of detail at which further clarification on a function is no longer necessary. As a result, a complete activity model can be finally obtained for the Product Structure Manager.

4 CPSM Data Model

To further capture the information contained in the resulting activity model, EXPRESS and EXPRESS-G [13] are used to describe the data model of the Product Structure Manager. In general, this data model, which is built upon the STEP standard [8,14], consists of two levels of abstraction in two models: Application Reference Model (ARM) and Application Interpreted Model (AIM).

4.1 Application Reference Model (ARM). This model is defined upon the basis of ARM of AP203 [15] and thus used to describe basic application objects of concern. The ARM contains 24 application objects that are distributed in four Units of Functionality (UoF), namely, Project Management, Bill of Material, Part Identification, and Design Information.

Project Management. As shown in Fig. 6, Project Management UoF contains 12 application objects that capture the information required for project creation, team (user) management, and product family creation for a specified product.

In Fig. 6, the **project** entity consists of several attributes, including *project_id*, *name*, *description*, *creation_date*, *owner*, *about_product* and *has_participants*. While the *project_id* is

unique to a specific application, the *owner* identifies a **team** to which this project belongs. The *about_product* specifies a **product_model** with which the collaboration is concerned. Each instance of the **product_model** may have a set of **variants** (affiliated to a product family), each in turn consisting of its constituent **parts**. The *has_participants* specifies the **teams** participating in the **project**.

The **team** itself is an abstract entity with a unique *team_id* attribute. There are two kinds of team in consideration: **client** and **design_team**. Both teams could have a single or multiple instances of **role**, and be responsible for a set of **parts** pertaining to a specified **variant**. Similarly, **role** is an abstract entity whose instance is either the instance of **coordinator** or the instance of **design_role**. In turn, **design_role** itself is a super type of **designer** and **team_leader**. Each instance of **role** belonging to a specific instance of **team** is allocated to an instance of **user**.

Bill of Material and Part Identification. As shown in Fig. 7, Bill of Material UoF contains five application objects, namely, **alternate_part**, **substitute_part**, **engineering_assembly**, **engineering_next_higher_assembly** and **engineering_promissory_usage**. On the other hand, Part Identification UoF includes three application objects, namely, **part**, **part_version**, and **design_discipline_product_definition**. These objects not only convey the information essential in creating the structure of a specified product, but also capture the relationships that associate product definition data with their components in the product structure.

An **engineering_assembly** is defined by a set of attributes such as *security_code*, *parent_component* and *child_component*. The *parent_component* specifies the parent constituent of an assembly, and the *child_component* the child constituents of the assembly. These constituents in turn are defined by **design_discipline_product_definition**.

An instance of **design_discipline_product_definition** has several attributes as described in the following. The *discipline_id* attribute uniquely identifies an object. The optional *cad_filename* attribute specifies the design documents containing product definitions. Each instance of **part** is identified by a unique *part_number* attribute and may have more than one **part_version**. On the other hand, an instance of **part_version** is defined by one or more instances of **design_discipline_product_definition**.

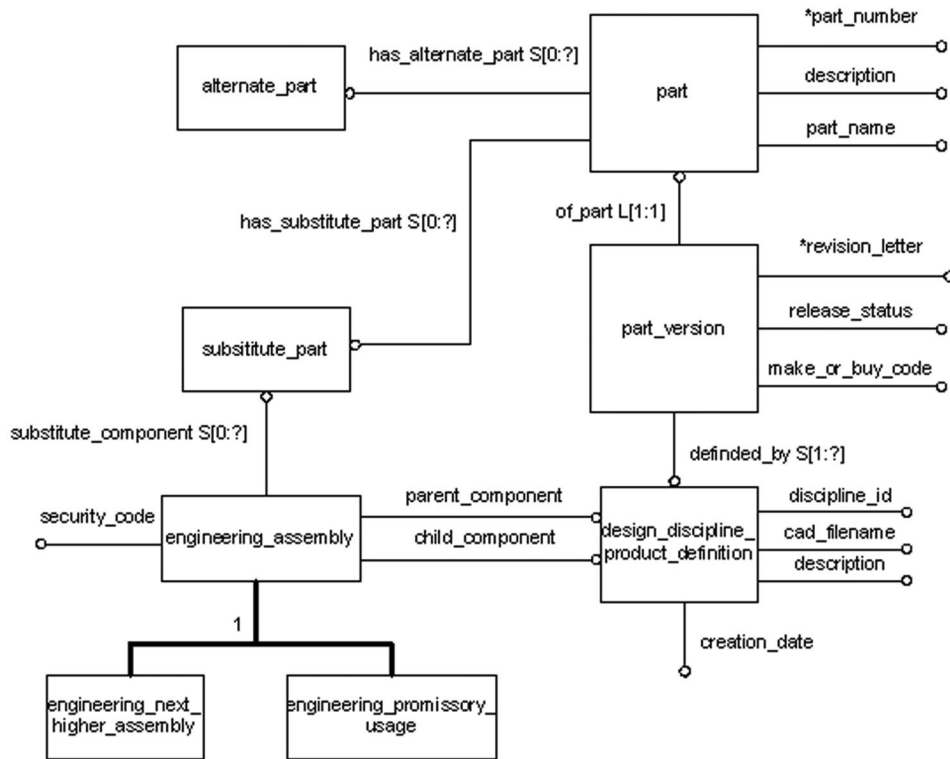


Fig. 7 EXPRESS-G Model of bill of material and part identification UoFs

Design Information. Design Information UoF includes four application objects, as shown in Fig. 8, which convey information on the specification of a *product*. The instance of **additional_design_information** contains add-on information, which is a collection of **specification** instances, for a single or

multiple instances of **design_discipline_product_definition**. Since **specification** is an abstract entity, each **specification** instance is either a **design_specification** instance or a **material_specification** instance.

4.2 Application Interpreted Model (AIM). This model is nothing more than an EXPRESS based information model that formally describes the application objects according to the existing definitions available in the STEP library. AIM represents the same set of information as ARM does but its focus is on the lower level of abstraction. In this work, both AIM and ARM share the same EXPRESS model for the Project Management UoF. Apart from those in the Project Management UoF, entities in AIM are built into two schemas, product_definition schema and product_structure schema, each providing one specific aspect of product definitions. The detail definition of each entity in these two schemas can be referred to AP 203 [15].

Product Definition Schema. The entities defined in product_definition schema contribute to a generic representation to convey how a product can be described by a set of definitions and be grouped by available versions. Figure 9 illustrates, in this schema, the identification and definition of a product, the grouping of multiple versions of a product and the definition of relationships between products.

In the engineering environment, a product may be viewed as the identification and description of a physical object. This kind of objects is all defined as instances of the **product** entity, each of which is characterized by several attributes such as *id*, *name*, *description* and *frame_of_reference*.

The purpose of the **product_definition_formation** entity is to support the identification of a specified version of a product. The *of_product* attribute of **product_definition_formation** specifies the product it belongs to.

STEP uses **product_definition** to provide a specific *life_cycle_stage* view for the product data whose attributes are *id*, *description*, *formation* and *frame_of_reference*. Thus the association between **product_definition** and

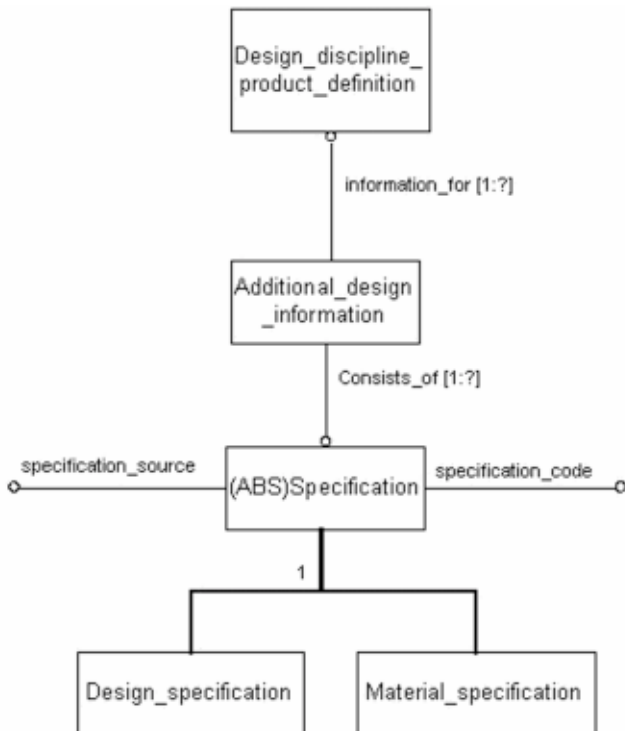


Fig. 8 EXPRESS-G Model of design information UoF

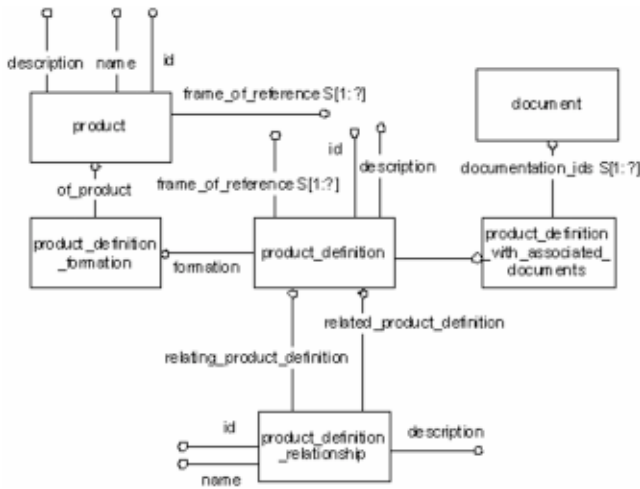


Fig. 9 Product definition schema in EXPRESS-G format

product_definition_formation is established through the *formation* attribute. Each instance of **product_definition** may relate to an instance of **document** that contains, for example, a CAD model through its sub-type **product_definition_with_associated_documents**. Each instance of **document** has an *id* that uniquely identifies its source of the file.

An instance of **product_definition_relationship** characterizes the relationship between instances of **product_definition** within a product. Based on the same product definition, the relationship of the parent-children elements would be specified as an instance of the **product_definition_relationship** entity. Using this entity as a reference, product structure schema can be further established and described (in the next section). The attributes of **product_definition_relationship** include *id*, *name*, *description*, *relating_product_definition* and *related_product_definition*.

Product Structure Schema. To convey how a product is comprised in association with other information, product structure schema is defined to describe a compositional relationship of the product. This relationship is a sub-type of **product_definition_relationship** defined in product_definition schema. Figure 10 exhibits the relationships among entities of this

schema in EXPRESS-G format.

In Fig. 10, **product_definition_usage** is a sub-type of the **product_definition_relationship** entity. Its sub-type, **assembly_component_usage**, is defined to describe the assembly relationship of a product in terms of its lower-level components. This entity, on the other hand, is a super-type of another four entities representing different assembly relationships: **quantified_assembly_component_usage**, **next_assembly_usage_occurrence**, **promissory_usage_occurrence**, and **specified_higher_usage_occurrence**, which are discussed as follows.

The **quantified_assembly_component_usage** entity indicates how many components are required to make an assembly. The *reference_designator* attribute inherited from **assembly_component_usage** uniquely identifies each component according to a proper reference assigned. The **next_assembly_usage_occurrence** entity specifies the relationship between a child component and its immediate parent assembly in a product structure. Instances of this entity in the same assembly can be grouped to construct a one-level Bill_of_Material (BOM) structure. Consequently, a multi-level BOM structure can be formed by appropriately concatenating all levels of **next_assembly_usage_occurrence** instances. The **promissory_usage_occurrence** entity specifies the relationship between a component and an assembly if the assembly is not the immediate parent of the component. This is often used when the product structure is not completely defined but it is clear that the component belongs to the assembly. Also, the **specified_higher_usage_occurrence** entity specifies the recursive relationship between a component and an assembly if the assembly is not the immediate parent for the component.

4.3 Mapping From AIM to ARM. ARM and AIM describe the same set of information in different forms and levels. To be specific, ARM specifies information requirements in high level, and AIM describes those entities conveying the information requirements in terms of generic resources in low level. In view of the implementation of the Product Structure Manager, ARM would be embedded in the form of tables in a relational database and AIM be incorporated in STEP physical files. At this point, a mapping from AIM entities to ARM objects is needed accordingly.

The mapping is defined in such a way that every object in ARM could be obtained from one or more AIM entities. As an example

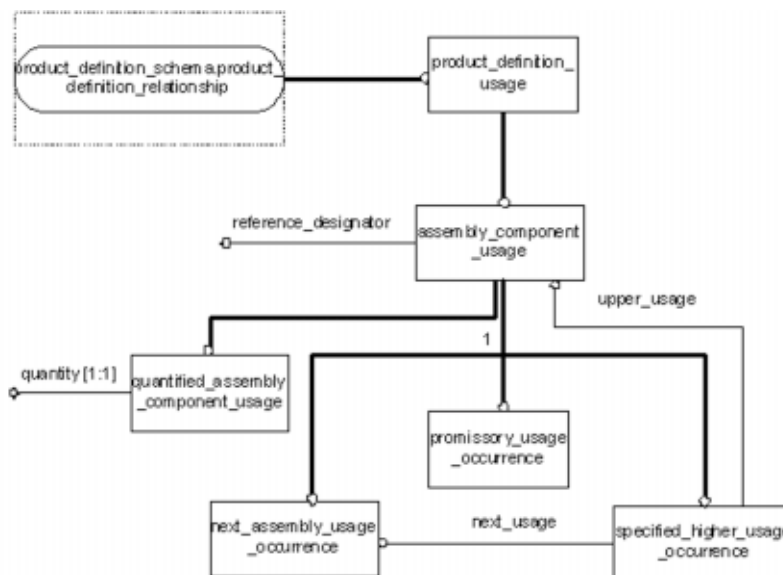


Fig. 10 Product structure schema in EXPRESS-G format

Table 1 Mapping table-bill_of_material UoF

ARM ELEMENT	AIM ELEMENT	SOURCE	REFERENCE PATH
ENGINEERING_ASSEMBLY	assembly_component_usage	44	
parent_component	product_definition	41	
child_component	product_definition	41	
security_code	security_classification_level	44	
ENGINEERING_NEXT_HIGHER_ASSEMBLY	next_assembly_usage_occurrence	44	
reference_designator	assembly_component_usage.reference_designator	44	next_assembly_usage_occurrence= assembly_component_usage assembly_component_usage.reference_designator
ENGINEERING_PROMISSORY_USAGE	promissory_usage_occurrence	44	
ALTERNATE_PART	Product	41	product {product<- alternate_product_relationship.alternate}
SUBSTITUTE_PART	Product	41	product {product_definition_usage=> assembly_component_usage<- assembly_component_usage_substitute.substitute}

shown in Table 1, **engineering_assembly**, an ARM object, is mapped from **assembly_component_usage** in AIM, while **engineering_next_higher_assembly** is obtained from **next_assembly_usage_occurrence**.

5 Implementation of Product Structure Manager

Based on the activity and data models obtained, this section discusses the implementation of CPSM through a prototype of the Product Structure Manager.

5.1 Manager Architecture. Figure 11 displays three-tier client-server architecture for implementation of the Product Structure Manager. According to this architecture, the Product Structure Manager is based on two kernels: CPSM STEP server and CPSM database.

The CPSM STEP server is composed of several main function modules. Of these modules, STEP Loader, coded in C++ upon ST-Developer from STEPTOOLS™ supports two main functions: up-loader and parser. While the up-loader uploads STEP physical files from distributed heterogeneous CAD systems to a temporary location on the CPSM central server, the parser analyzes the uploaded files and stores the resulting instances of the STEP-compatible non-geometry product data in the CPSM database. The other three modules-project management, product structure management, and product structure graphic tree viewer-support functions in collaboration such as collaborative creation, modification and deletion of a project(s) and the related product structure(s). These functions are implemented using Java programming language.

On the other hand, the collaborating participants can access the

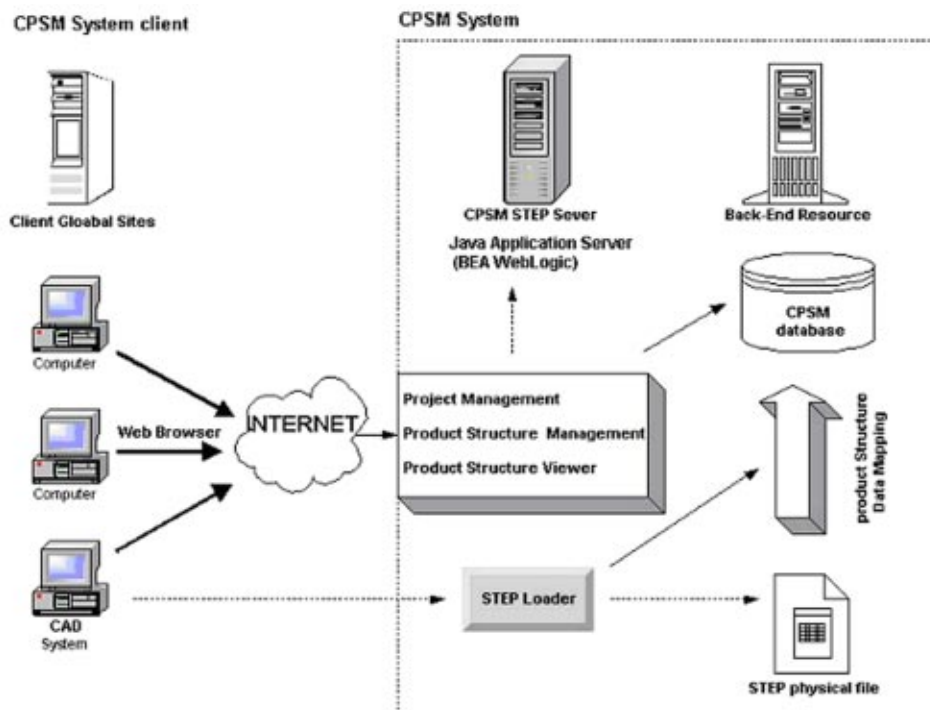


Fig. 11 Three-tier client-server architecture of product structure manager

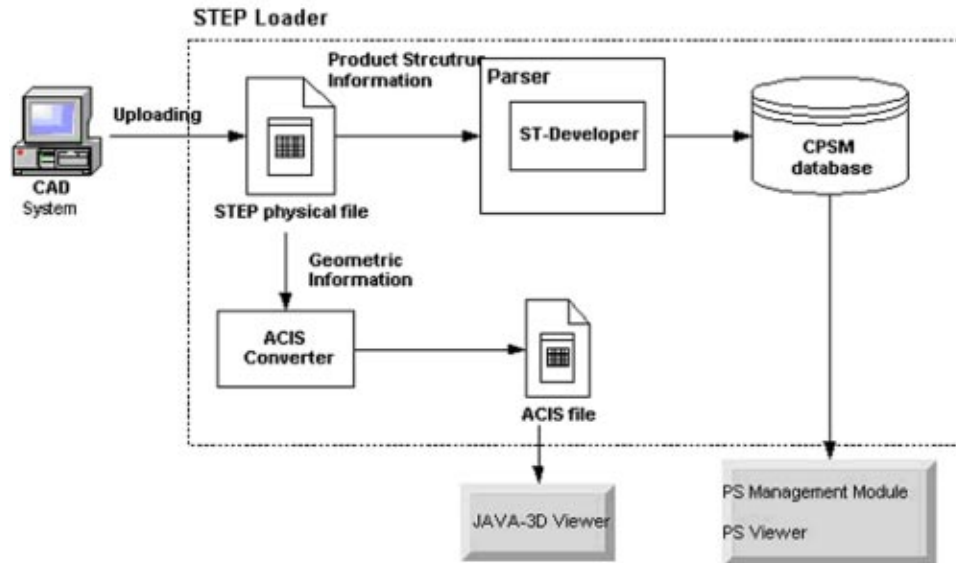


Fig. 12 Architecture of STEP loader

client-side GUI of Product Structure Manager through a Web-browser. In this architecture, ROSE library API, which is a collection of C++ class definitions provided by ST-Developer, is used for STEP file access and Java for database access and GUI implementation. The whole system is built upon a platform using such development tools as Oracle 8.1.5, ST-Developer 9.0 and BEA WebLogic Server 6.1.

5.2 STEP Loader. The STEP Loader is purported to upload STEP physical files from clients' locations to the central server and then to parse these files to obtain the corresponding ARM objects. These objects are in turn mapped to the system database. The data flow involved in this process is depicted in Fig. 12. As shown, the loader handles the geometric and non-geometric (product structure) information separately. When a STEP physical file is uploaded to the central server, non-geometry data is processed by

the parser and then stored in the system database. The information-extracting module of the parser is written in C++ upon ROSE library.

In Fig. 12, the parsing action is done through a user interface implemented in Java. Therefore, Java Native Interface (JNI) is utilized to integrate C++ modules. On the other hand, the database access module of the parser is implemented through the API of Java Database Connection (JDBC) that provides database access in Java programs.

The geometry data contained in a STEP file is transformed into an ACIS file via the STEP-to-ACIS translator. As designers export their original designs in STEP format, a manual control is necessary in order to generate STEP AP 203 CC6 data only [15]. Based on a set of C++ classes, this translator performs the translation from the STEP AP203 CC6 data to an ACIS SAT file. The gener-

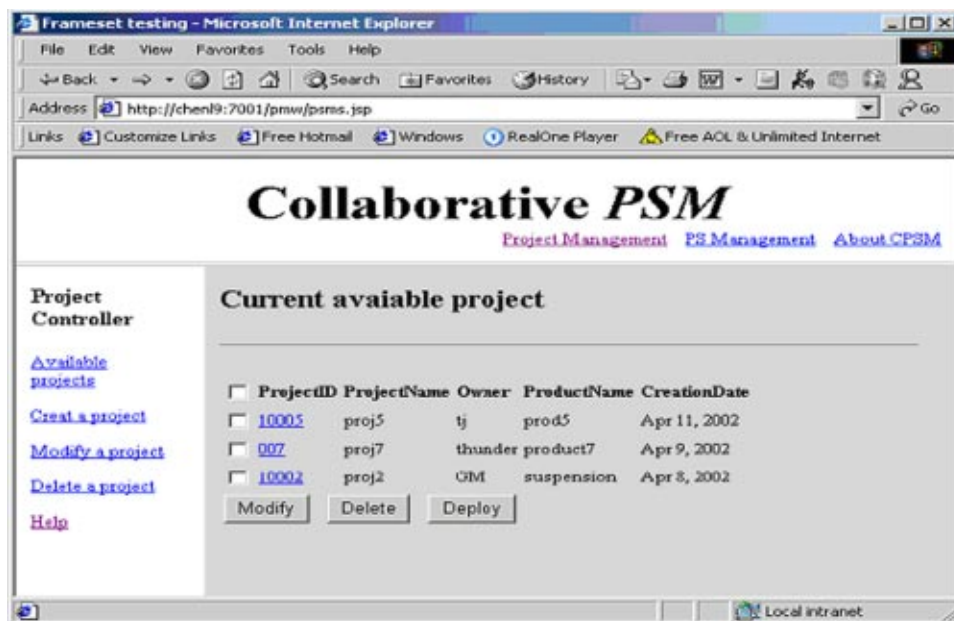


Fig. 13 Project management page

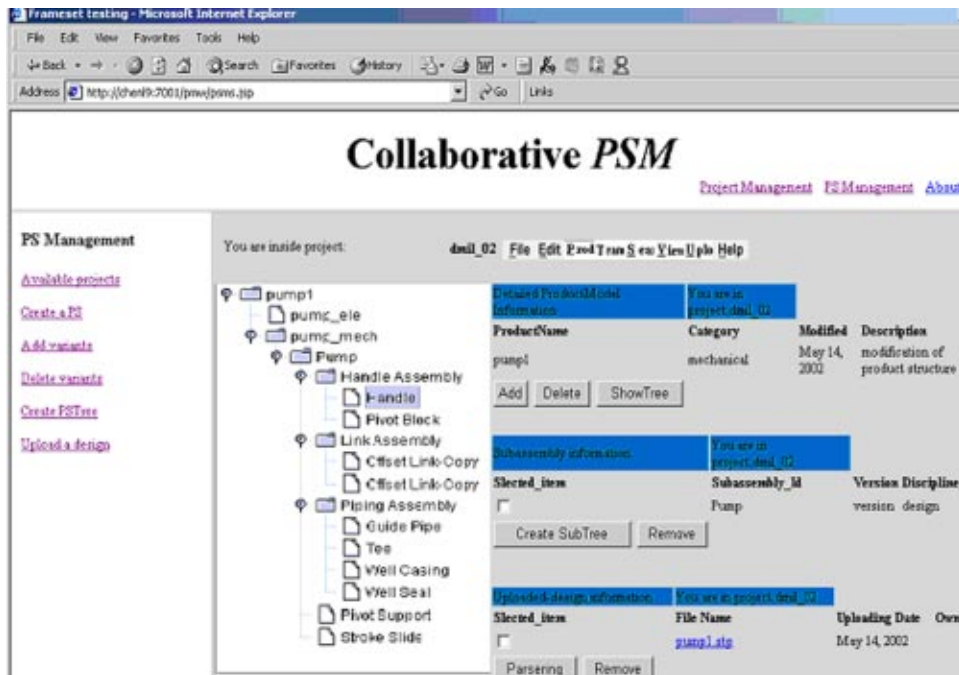


Fig. 14 Project implementation page

ated ACIS files are then processed by another translator that is newly introduced in this work. This new translator is composed of a set of C++ classes that make use of the ACIS Faceter component, and thus able to generate an approximate polygonal representation, named “Web-Rep” [16], for visualization of a B-rep model. The use of Web-Rep greatly reduces the volume of geometry data transmitted for visualization. Using the ACIS persistent ID mechanism, Web-Rep also maintains links between its own entities and the original B-rep data structure entities in ACIS. Such links make it possible to perform geometry manipulation through Web-Rep.

The parser also associates the components of a product structure with their corresponding geometrical definition data such as CAD models. In this way, one can easily access the geometry and product definition of a particular component via the product structure.

5.3 Function Realization. The implementation system contains three main function modules, namely, project management, product structure management (definition and creation), and product structure graphic viewer.

• **Project Management**

This module sets up a project for collaboration. The functions contained in this module are integrated in the Project Management Page displayed in Fig. 13.

• **Product Structure Management**

This module provides functions to support product structure creation and modification activities. The functions contained in this module are integrated in the Project Implementation Page displayed in Fig. 14.

• **Product Structure Graphic Viewer**

This module supports implementation of a graphic tree for a product structure. Compared to the other two modules, the product structure viewer has no separate interface. This module is implemented as a Java applet and thus plugged into the Project Implementation Page as exhibited in Fig. 14.

6 Illustrative Example

This section illustrates the application of the Product Structure Manager to the collaborative assembly modeling of a bench clamp. The objective is to create a new design variant to the existing assembly model shown in Fig. 15. Table 2 lists the complete set of *nine* components in the existing assembly model. Furthermore, this application scenario assumes that there are three *distributed* members (individuals or companies) participating in this collaborative assembly modeling practice. Accordingly, Table 3 outlines the responsibility of each member for an assigned task, where each subassembly is created presumably using a different CAD system.

The new variant to the existing bench clamp assembly is created using the “collaborative assembly modeling” approach with the aid of the Product Structure Manager. The main steps involved in this practice are summarized as follows.

1. The team leader set up the bench clamp re-design project through the Product Structure Manager. In particular, the team leader defined the project, set project participants (3 members in this case) and provided some other information (see Fig. 16).

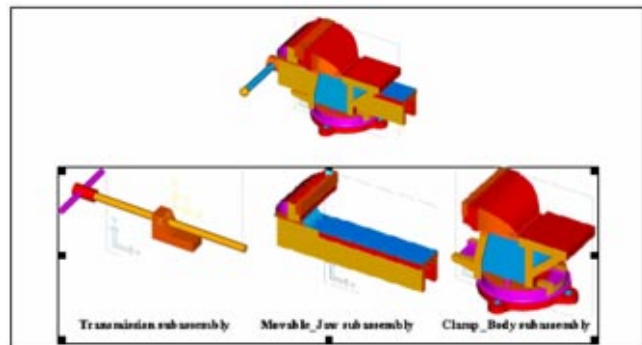


Fig. 15 Bench clamp assembly and subassemblies

Table 2 List of bench clamp assembly components










Shape	Name	Quantity
	Handlehead	2
	Handle	1
	Screw Rod	1
	Movable Jaw	1
	Fixed Jaw	1
	L-Nut	1
	Base Mount	1
	Movable Jaw Insert	1
	Fixed Jaw Insert	1

Table 3 List of design task allocations

Design Team	Subassembly	CAD System
Member A	Transmission subassembly	Unigraphics
Member B	Movable_Jaw subassembly	I-DEAS
Member C	Clamp_Body subassembly	Pro/Engineer



Fig. 16 Project creation page

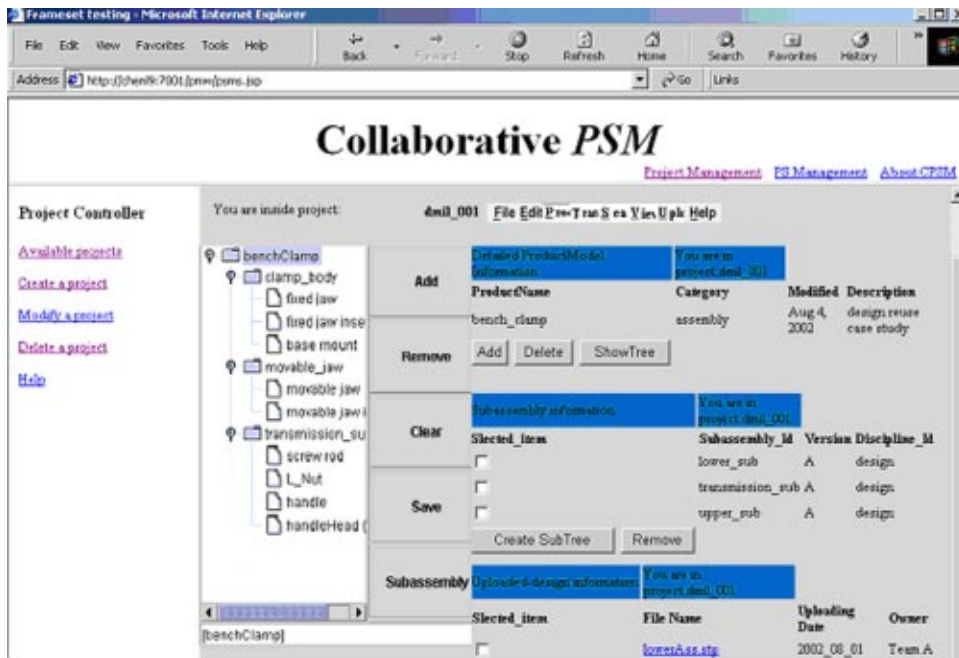


Fig. 17 Collaborative creation of the bench clamp structure

2. The team leader defined the global product structure (highest-level hierarchy for the new bench clamp) and assigned design tasks (e.g. subassembly modeling) to the participants. In this example, the team leader divided the bench clamp into three subassemblies with distribution to three participants according to Table 3.

3. The participants worked out their own subassembly individually and then stored the resulting CAD model respectively in STEP data format.

4. The participants uploaded their STEP CAD data file individually onto the Product Structure Manager. Then the STEP loader in the Product Structure Manager was used to analyze each uploaded STEP file, extract the product structure of each subassembly from the file, and finally insert each extracted structure into the global product structure. The final result is displayed in Fig. 17.

5. The participants collaboratively assemble the new bench clamp in real time via “collaborative assembly modeling.” Note that the successful realization of this step also requires some other collaborative CAD modeling tools, in addition to the Product Structure Manager, which will be addressed separately in a future paper.

For illustration, one exemplary scenario for the demand of a new design variant to the existing model is considered in this example. Table 4 outlines the modifications done to the existing design according to Table 2. In specific, one change is to add flat faces at the end of the *Screw Rod* so as to facilitate the alternate screwing. However, this change decreases strength in the resulting rod for the reduced wall thickness. To maintain the desired strength in the re-shaped rod, the diameter of the hole in the *Screw Rod* and the diameter of the *Handle* are both reduced cor-

Table 4 List of the modified components

Name	Existing Component	Redesigned Component	Comment
Screw Rod			Add flat surface on the end of screw rod to facilitate alternative driving approaches
Handle			Reduce the diameter of handle accordingly to fit the hole on the screw rod
Movable Jaw Insert			Add circular feature on both Movable Jaw Insert and Fixed Jaw Insert to allow for holding cylindrical workpieces.
Fixed Jaw Insert			Similar to the above

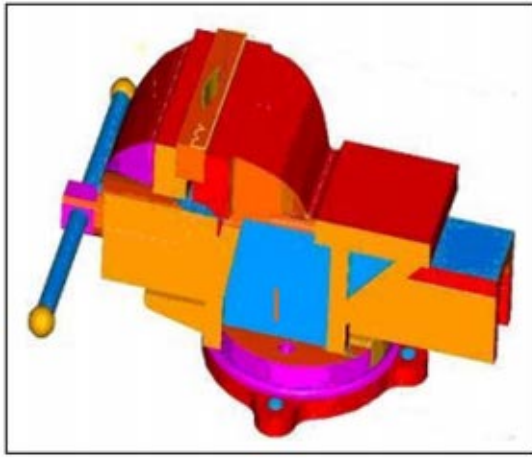


Fig. 18 New design of the bench clamp assembly

respondingly. Another change is to add a circular profile on both the *Movable Jaw Insert* and the *Fixed Jaw Insert* to accommodate the holding (clamping) of round-shaped workpieces. Figure 18 shows the new assembly model of the bench clamp as the result of the above changes.

7 Conclusions

This paper has addressed the Web-based Product Structure Manager that enables the Collaborative Product Structure Management (CPSM) to support collaborative assembly modeling. This Web-based CAD tool serves as one function module in our collaborative assembly modeling system. In this paper, both the activity and data models of CPSM have been presented and discussed. The development of a prototype of the Web-based Product Structure Manager based on the three-tier client-server architecture has been described. This work represents our continual effort toward a collaborative CAD system dedicated to platform-free 3D assembly modeling in a distributed team design environment.

The resulting Product Structure Manager provides an efficient CAD tool to support team design activities in collaborative assembly modeling. Nevertheless, the current scope of development does not encapsulate another two important issues: product data security and product data redundancy. The former issue is concerned with levels of detail in data management to account for different levels of access to product information in collaborative

assembly modeling. The latter one is concerned with efficiency and consistency in data storage to account for the growth of variants of a product in collaborative assembly modeling. To this end, one stream of our effort underway is devoted to a more concise yet elaborate product structure information model that combines the product variant/classification model with the product breakdown structure model. Last but not least, the approach of CPSM also provides an alternative way for synergistic reuse of design information in platform product design, which facilitates creating variants for a product family.

References

- [1] Pang, A., and Wittenbrink, C., 1997, "Collaborative 3D Visualization with Cspray," *IEEE Comput. Graphics Appl.*, **17**(2), pp. 32–41.
- [2] CoCreate, (<http://www.cocreate.com>).
- [3] Alibre, (<http://www.alibre.com>).
- [4] Agrawal, A. K., Ramani, K., Hoffmann, C. M., 2002, "CADDAC: Multi-client Collaborative Shape Design System with Server-based Geometry Kernel," *Proceedings of the ASME Design Technical Conferences and Computers and Information in Engineering Conference*, DETC2002/CIE-34465.
- [5] Lee, J. Y., Han, S. B., Kim, H., and Park, S. B., 1999, "Network-centric Feature-based Modeling," *Seventh Pacific Conference on Computer Graphics and Applications*, pp. 280–288.
- [6] Chen, L., Song, Z. J., and Liavas, B., 2001, "Exploration of A Multi-User Collaborative Assembly Environment on the Internet: A Case Study," *Proceedings of the ASME Design Technical Conferences and Computers and Information in Engineering Conference*, DETC2001/CIE-21291.
- [7] Chen, L., Song, Z. J., and Liavas, B., 2002, "Master Assembly Model for Real-Time Multi-User Collaborative Assembly Modeling on the Internet," *Proceedings of the ASME Design Technical Conferences and Computers and Information in Engineering Conference*, DETC2002/CIE-34456.
- [8] Owens, J., 1997, *STEP An Introduction*. 2nd ed, Information Geometers, UK.
- [9] Mckay, A., Erens, F., and Bloor, M. S., 1996, "Relating Product Definition and Product Variety," *Res. Eng. Des.*, **2**(1), pp. 63–80.
- [10] CIMdata Inc., 1998, "Product Data Management: The Definition," Ann Arbor, MI.
- [11] Svensson, D., and Malmqvist, J., 2002, "Strategies for Product Structure Management at Manufacturing Firms," *ASME J. Comput. Inf. Sci. Eng.*, **2**(1), pp. 50–58.
- [12] O'Donenell, F. J., MacCallum, K. J., Hogg, T. D., and Yu, B., 1996, "Product Structuring in Small Manufacturing Enterprise," *Comput. Ind.*, **31**(3), pp. 281–292.
- [13] International Standard Organization, 1994, *Industrial Automation Systems and Integration-Product Data Representation and Exchange-Part 11:Description Methods: The Express Language Reference Manual*, Geneva, Switzerland.
- [14] Peng, T. K., and Trappey, A. J. C., 1998, "A Step Toward STEP-Compatible Engineering Data Management: the Data Models of Product Structure and Engineering Changes," *Rob. Comput.-Integr. Manufact.*, **14**, pp. 89–109.
- [15] International Standard Organization, 1994, *Industrial Automation Systems and Integration-Product Data Representation and Exchange-Part 203: Configuration Controlled Design*, Geneva, Switzerland.
- [16] Chen, L., Song, Z. J., and Feng, L., 2003, "Internet-enabled Real-Time Collaborative Assembly Modeling via an e-Assembly System: Status and Promise," *Comput.-Aided Des.*, In Press, available online 27 October 2003.