CONCURRENT ENGINEERING: Research and Applications

Product Node Architecture: A Systematic Approach to Provide Structured Flexibility in Distributed Product Development

Charles Chan-Woo Chung,[1] Jun-Ki Choi,[1,*] Karthik Ramani[1] and Harshal Patwardhan[2]

[1]PRECISE, School of Mechanical Engineering, Purdue University, West Lafayette, IN 47907, USA
[2]School of Industrial Engineering, Purdue University, West Lafayette, IN 47907, USA

Abstract: Representation of the product/project information throughout the design life cycle is a critical aspect in engineering activities. The article begins with a discussion of the background research wherein the existing methodologies, which deal with product information representation, are reviewed. The article then proceeds to propose a new methodology for product management by presenting all the details of the system architecture. Product node (PN), which is introduced in this article, has a database-centric system at its core and an application built around it to support the various functions necessary to efficiently manage a distributed project environment. To aid the design life cycle, the system also enables distributed collaboration among people from different teams to support concurrent design activities. Effective management of product data using user-based control is also discussed. The detailed logic behind the system architecture and the practical implementation of the system are shown.

Key Words: product modeling, product life cycle management, product node, product and project management, concurrent engineering.

1. Introduction

The globalization of the manufacturing industry has caused companies to restructure their processes to become more efficient. To efficiently manage product data throughout its life cycle, many large companies have started to use information technology (IT) tools extensively. Rapid advances in related technologies, such as the Internet, databases, and advanced programming languages have further fueled the trend of IT usage by companies. Email and FTP are the most widespread and easy-to-use means of communication. Today, more complicated web-based or application-based solutions are becoming more common. Various solutions for the effective management of product data, collaboration, and project management have been suggested [1–3,21]. Product life cycle management (PLM) concepts represent a vision for solving the problems associated with the management of a company. Although new, the PLM/PDM (product data management) market generated revenues worth several billion dollars in 2001 and is expected to grow at a rate of 20–30% annually [4,5]. However, the PLM systems currently available on the market are still faced with many unsolved issues. Limited flexibility to expand to any type of product definition and compatibility problems in data exchange are the most common problems. The internal processes of a company are, in general, very complex, so it takes a lot of time for PLM vendors to analyze these processes to provide a customized service/solution. Eliminating the lead-time in the implementation requires identification of the common denominators of the system, thus accelerating the customization process. During the product design process, managing the digital formats throughout the process among multiple team members is very important. The design details of the product node (PN) system will be presented after an introduction of the important concepts related to the product and project modeling. As will be shown throughout this article, the database-based structure of the PN system shows 'structured flexibility,' enabling dynamic handling for the design processes. The later sections give the details of the PN structure and its implementation.

2. Product Model

2.1 Static System and Dynamic System

Present-day project and process management need a complete plan before initialization. If a plan must be completed before initialization of a project, this results in delays between the starting time and the actual runtime of the project. Elimination of the gap between

*Author to whom correspondence should be addressed.
E-mail: jkchoi@purdue.edu

1063-293X/05/03 0219–14 $10.00/0 DOI: 10.1177/1063293X05056472

the starting time and the actual runtime, and the synchronization of many processes from the concept generation stage to the lifetime management of the project are important issues since they are directly related to the revenue of a company. Unlike a parallel process, the asynchronous or serial process, in general, cannot use all the resources efficiently in time. Such inefficiency characterizes a static system in the product design process [6]. In a real world design process, constant modifications of individual products and process parameters take place. Such modifications happen more frequently in distributed design environments because more interactions often create more ideas for enhancement of the design. The evolutionary development of product parameters is intuitive, and creative solutions are possible in a changeable environment (dynamic system) where mistakes are acceptable [6]. The configurations of such changeable systems are those built up from manageably small modules, each of which is as far as possible independent of one another [7]. Enabling dynamic changes of the product breakdown structure (PBS) throughout the life cycle of a product will resolve the issues caused by the strict constraints of the system.

2.2 Product Family Classification Trees

Any product definition can be categorized in product family classification trees (PFCT). Similar to a class structure of object-oriented programming languages, it has characteristics, such as inheritance and generalization. The relationship from lower to upper level is 'a kind of' since the modules in the lower level of the tree structure inherit all the characteristics of the module in the upper level, i.e., a class of product is a kind of the superclass of product. A product definition can be treated as one of the instances in the class structure where instances are units stored in a database table. Allocating a product definition in PFCT will allow users to systematically classify and store product data and ease the effort for future use by searching through the organized data. Figure 1 shows the structure of a general PFCT [8]. A design repository structure based on the PFCT is shown in [9]. Object-oriented approaches for product design have also been suggested by [10,11].

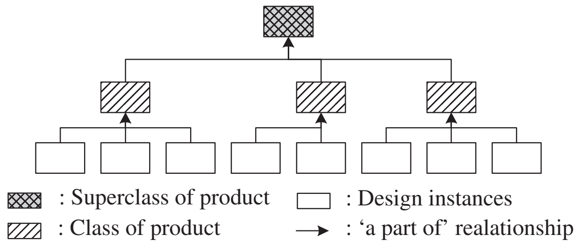


Figure 1. Product family classification tree (modified after [8]).

2.3 Product Breakdown Structure

Product breakdown structure (PBS) is a type of structure that can represent the hierarchy of a product [20]. It has 'a part of' relationship from the lower level modules to the upper level module. The upper level module includes all the definitions in the lower level module(s), and it has a 'has' relationship from the upper level modules to the lower level modules. Figure 2 shows the graphical interpretation of such a structure. The building process of PBS closely resembles the human thinking process of designing a product when it grows from the root to the leaves. We can start from an abstract definition of a product and go down further for details because this structure can be built by breaking a whole body into smaller modules. The process of building a PBS can be said to be a top-down process in contrast to a bottom-up design. The use of such breakdown structures is expected to increase the traceability of information throughout the product life cycle. Project breakdown structures have been found to be beneficial for engineering data management in large-scale projects [12].

2.4 System Modeling with Multiple Types of Interactions

Often the simple notation of an interaction is not enough to represent the many different types of interactions. For example, if there are multiple types of interactions, such as force and energy among components, they are hard to represent in a binary matrix [13]. In this case, multiple matrices must be used to show the relationships. Furthermore, there is a need to indicate to the users 'the property of interaction' clearly. Database definition of such a relationship can represent such a relationship among components.

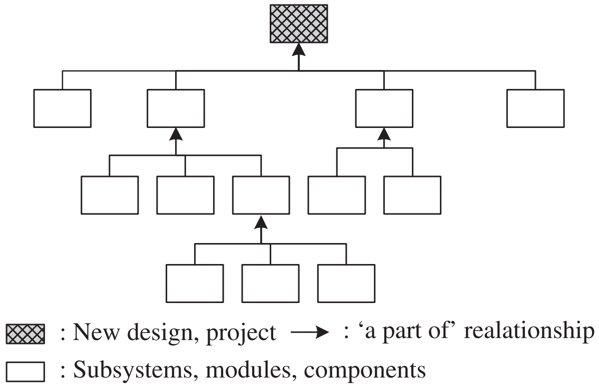

: New design, project
: 'a part of' realationship
: Subsystems, modules, components

Figure 2. Product breakdown structure (modified after [8]).

Figure 3 shows a PBS of a simplified R/C (remote control) car model. There are functional interactions among components in addition to the physical structure of the model. The binary matrix of the functional relationships is shown in Figure 4. Since the model was built based on the structural diagram from Figure 3, a permutation process is not necessary in this case. The flow of material is similarly defined in Figure 4, but in this case, there are two types of material flow, i.e., electronic current and mechanical force.

Figure 4 shows that from Component 1 to 5, the dominant flow is electronic current, while from Component 6 to 11, the dominant flow is mechanical force. The relational diagram of this interaction is shown in Figure 5, which shows two types of flow indicated by different types of arrows.

The multiple relationships cannot be represented in a single binary matrix model. The complex, possible multiple relationships can be represented in database relations (relation: table in a database system) relatively easily [14].

3. Product Node Architecture

The design of a dynamic product node (PN) architecture was developed with the intention to simplify the process of distributed project and product management.

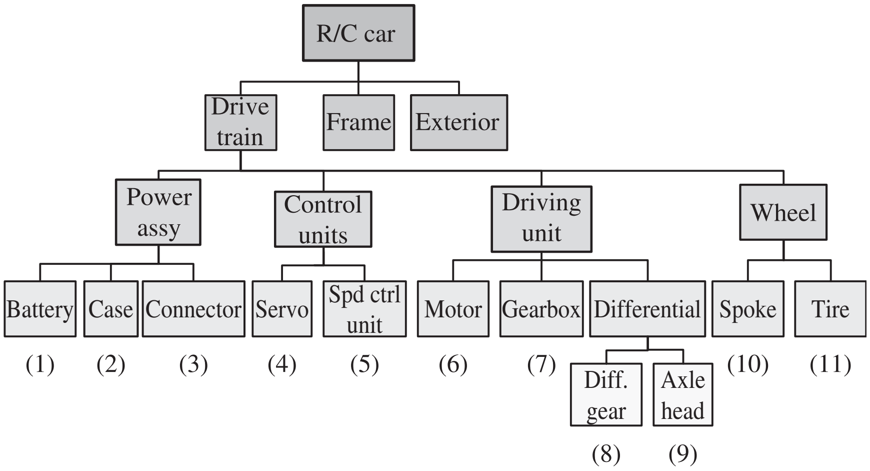
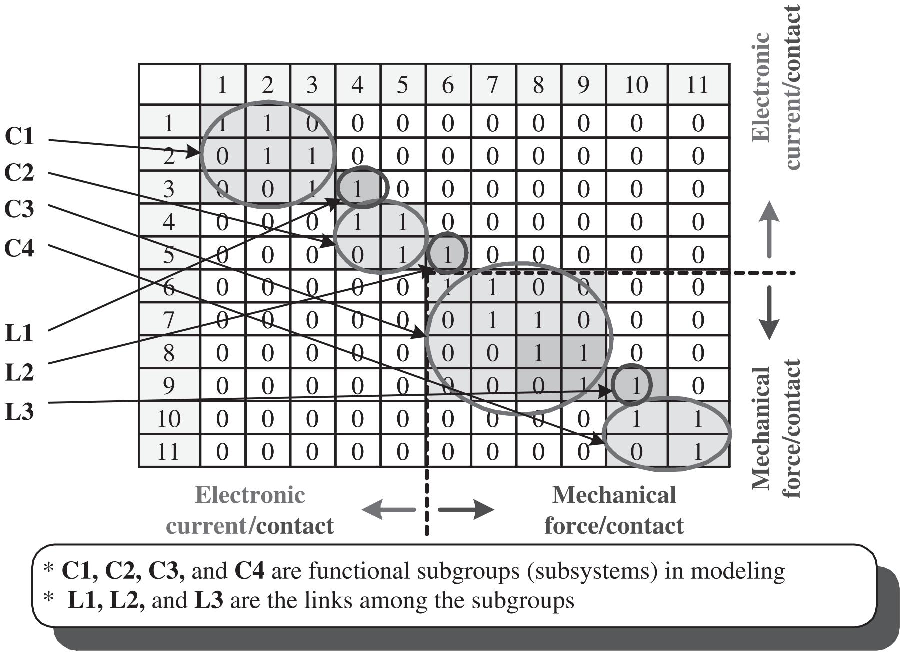


Figure 3. Product structure of an R/C car model.



	1	2	3	4	5	6	7	8	9	10	11
1	1	1	0	0	0	0	0	0	0	0	0
2	0	1	1	0	0	0	0	0	0	0	0
3	0	0	1	1	0	0	0	0	0	0	0
4	0	0	0	1	1	0	0	0	0	0	0
5	0	0	0	0	1	1	0	0	0	0	0
6	0	0	0	0	0	1	1	0	0	0	0
7	0	0	0	0	0	0	1	1	0	0	0
8	0	0	0	0	0	0	0	1	1	0	0
9	0	0	0	0	0	0	0	0	1	1	0
10	0	0	0	0	0	0	0	0	0	1	1
11	0	0	0	0	0	0	0	0	0	0	1

* **C1, C2, C3,** and **C4** are functional subgroups (subsystems) in modeling
* **L1, L2,** and **L3** are the links among the subgroups

Figure 4. Binary matrix of R/C car model.

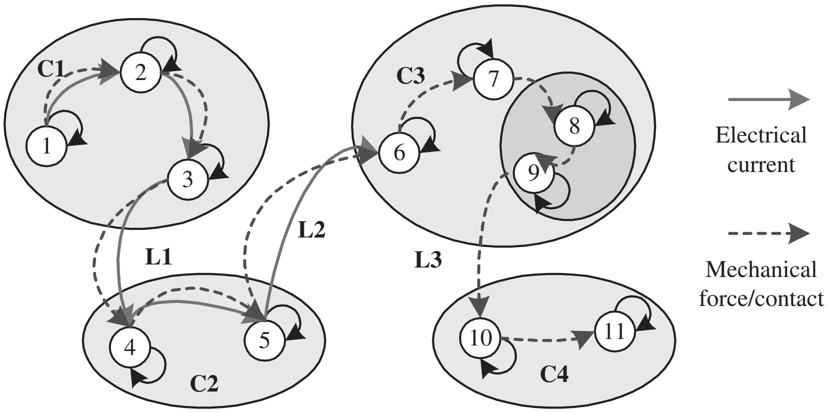


Figure 5. Relational diagram of R/C car model.

The PN structure is a customized representation and specialized use of PBS and is introduced to represent the project and product structure. Using the PN architecture, a company can manage the data related to a product with enhanced flexibility. With the use of efficiently designed database relations and the dynamic structure, the manipulation, expansion, change, or history tracking of the design data are realized. In addition to the flexible manipulation of the structure itself, the ability to add any product-related information around the structure enables companies to expand their product definition. An individual PN can have any information necessary to perform product development activities. PN architecture can be implemented in any database system in conjunction with any computer programming languages; Java^{TM} programming language and ORACLE^{TM} database system are used in our work. Since the PN is defined in a database system to set as a relationship between each node, it can expand its contents by associating more relations with it without having restrictions. Evolution of the PN structure occurs by creating, copying, and removing the PNs. Any addition or change of a specific PN occurs by controlling the records in the database table, PRODUCTNODE (PRODUCTNODE: a database table definition to represent PN). Another advantage of this type of structure is the ease in arranging and searching for any data systematically with the help of the database manipulation that is often performed with the use of structured query language (SQL). The major difference between the PN structure and other tree-type structures is that the PN structure treats any level of PN as a complete entity, so it allows any PN unit to contain any data whenever necessary. Recent developments in various fields have necessitated modeling schemes, which extend beyond the part geometry and include relevant information regarding the product [15]. However, most CAD or PDM systems still focus mainly on managing the CAD parts geometry of a product. Similar to the traditional role of drawings, they provide tools to mimic the paper drawings and to put in some auxiliary information and ultimately try to replace them. On the other hand, any node unit in the PN structure has an association with the other project-related information. For example, user management and marketing information can be a part of a component of a PN along with the parts information. The information needs in a PN structure can vary from company to company. The typical data that can be used in PN is listed below:

- Name of the product node (PN)
- Part number or assembly number
- Duration, schedule, and progress (Gantt chart)
- Requirements, specifications, functional descriptions
- Drawings (conceptual, CAD, assembly)
- People involved (users)
- Test and analysis
- Manual and instruction
- Bill of material (BOM)
- Vendor information
- RFQ, quotation
- Manufacturing information
- Marketing research
- Budget
- Change request
- History of the design
- Other documents

3.1 Hierarchy of PN Architecture

The PN architecture has six levels in its hierarchy. The global PN system itself stays at the top level with the name of ROOT, and the GROUP comes next. GROUP is a unit representing a small or medium enterprise (SME) or a division of a big company. A GROUP is a fundamental unit of the PN system in practice. Actual PN units can exist within a GROUP domain. PROJECT follows GROUP and is a unit of any actual project (work). PRODUCT, PART, and COMPONENT can be arbitrarily defined by the users to perform the

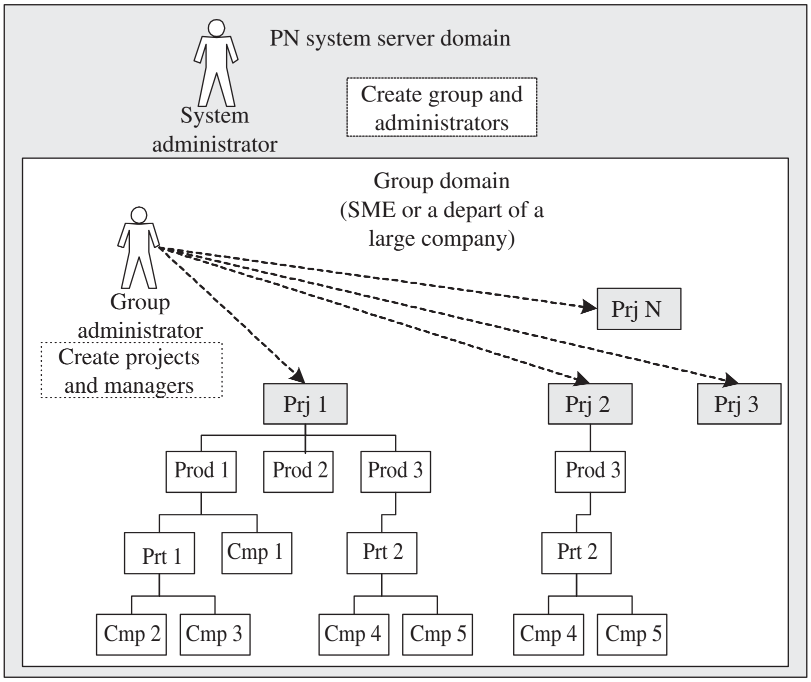


Figure 6. Overall hierarchy structure of PN system.

product design project. The schematic of the overall hierarchy of the PN system is shown in Figure 6.

Users will only see the projects within the GROUP domain that they are associated with since users do not need to have system administrator privilege designed to create group and administrators. PN system server domain is only 'ROOT' for the GROUP domain. Detailed transactions within the GROUP domain will be explained in the next sections.

4. Types of Product Node

The creator of a PN determines its type. Type allows users to know what type of PN they are working on. There are PROJECT, PRODUCT, PART, and COMPONENT in PN types. Table 1 shows a list of the types of PN and their roles. A PN type, PROJECT, is named and treated separately in the system since it is the starting point of project and it is only parent-PN. Although the structure of a PROJECT-PN is almost identical to that of the other PN types, it forms the basis of any design activity. An administrator or manager with privilege can create a PROJECT-PN to initiate a project.

4.1 Data Definition of Product Node

The PN has multiple attributes and include field names, primary and foreign keys. The relation, PRODUCTNODE, has basic fields needed since other related information is linked to it by the use of association tables (table: different name of relation).

Table 1. Types of product nodes in 'NODETYPE'.

Node type	Role
Project	Fundamental unit in PN structure, starting point of any project in a company (group level)
Product	Unit of a product within a project
Part	Part, combination of zero to many components
Component	A unit component which cannot be divided any further, leaf node

The attributes and definition of the relation, PRODUCTNODE, in the PN system are shown in Table 2. ID is a Primary Key (Primary Key: a unique ID of a database table; often shown with solid underline) of the relation, PRODUCTNODE, to make an instance (instance: a record of a database table) of the relation a unique one. Other multiattributes concerning history tracking, and reuse of a PN will be discussed in later sections of this article with implementation examples.

5. Parent–Child Relationship

One of the attributes of a PRODUCTNODE is the Foreign Key (Foreign Key: Index to the other table's Primary Key), ParentID, to link the current PN to the Parent-PN. The parent–child relationship forms the basis of the PN structure. The structure grows from one parent (i.e., PROJECT-PN) by creating children-PNs (i.e., PRODUCT-PN, PART-PN) underneath. The relationship is shown in Figure 7.

Table 2. Definition of PRODUCTNODE table.

Field name	Definition
ID	Node ID, Primary Key
Version	Version of the Node in incremental Number
ParentID	Parent Node's ID, Foreign Key
PreviousID	ID of a previous version Node, Foreign Key
NodeTypeID	Type of Node, Foreign Key
CurrentFlag	Flag to indicate if the Node is the current (latest) one or not. 1 (yes) or 0 (no)
Name	Name of the Node, Optional
TimeBuilt	The time PN was created
TimeStart	Start time of the Node, Duration starting point given by users
TimeEnd	End time of the Node, Duration ending point given by users
TimeInitiation	Time of Initiation of a Node due to a new version
TimeTermination	Time of Termination of the Node due to a new version
Description	Description of the Node
KeyWord1	Key Word field to assist searching
KeyWord2	Key Word field to assist searching

(_______): Primary Key; (_ _ _ _ _ _ _): Foreign Key.

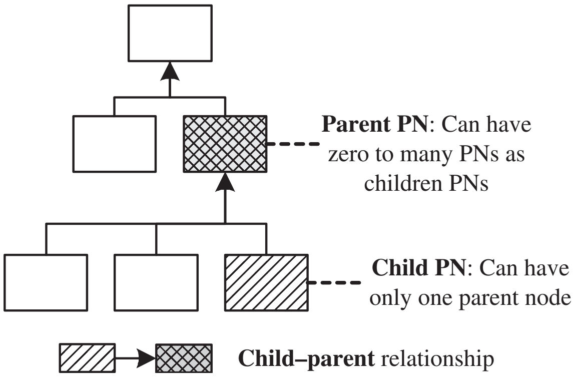

Parent PN: Can have zero to many PNs as children PNs
Child PN: Can have only one parent node
Child–parent relationship

Figure 7. Parent–child relationship of PNs.

Among the many types of PN, only PROJECT-PN can exist without having any Parent-PN associated since PROJECT-PN is the starting point of any project. In the system level (internal logic of the software), GROUP plays the role as the parent of any PROJECT-PN within the group. Figure 8 shows the detailed relationships between parent and child-PNs at the database level. The foreign key, PID, of a child-PN refers to the primary key ID of a parent-PN.

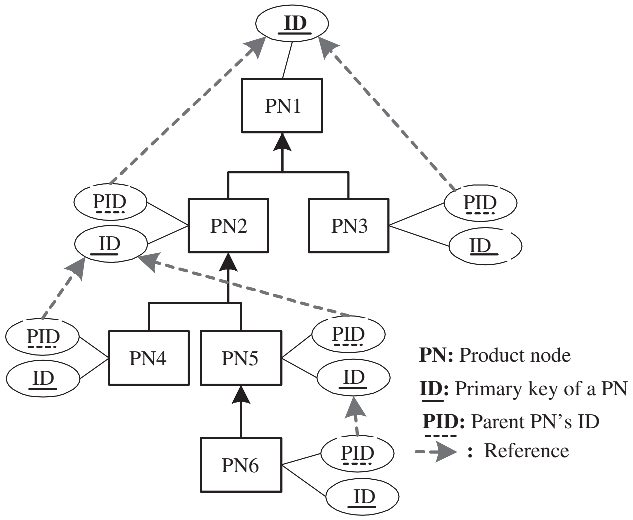


Figure 8. Product node structure and parent–child relationship.

6. Manipulation of PNs

Dynamic project management entails many changes in the product structure. Based on manipulations (adding, changing, and removing) of the PN structure, the product definition will grow or shrink. After a PROJECT-PN is created by an administrator to initiate a project at the GROUP domain level, additional functional menus to input other critical information, such as users and tasks, will be available to the users based on the privileges given. Creating a PN is the same as adding one conceptual unit or a physical part in the PN structure. From any location of the PN system, users can create a child-PN of it. With the command to create, the PN system will ask the user to input some attributes of the relation, PRODUCTNODE. Name, StartDate, EndDate, NodeType, and Description are the input needed by the system, and other attributes will automatically be assigned in the table, PRODUCTNODE. An ID will be assigned with a unique number and the Version will start from 1.0. After confirming the transaction, a new child Node will show up in the PN structure. As shown in Figure 8, the foreign key ParentID will point to the ID of the parent-PN after the creation.

6.1 History Tracking of PN

Previous design information is a critical resource for companies and often differentiates a company with a long history from new start-ups. The problem is that a big portion of the company knowledge exists in the heads of company employees and not within a company repository. So, as employees leave or retire from the company, the design know-how also gets lost. More than 50% of designers and manufacturing designers will leave their companies over the next decade [17]. Most companies are trying their best to maintain all their design and related documentation, but in many cases, they fail due to lack of proper tools. PN structure provides a means to store design-related information along with various other functionalities, thus easing the maintenance of this information for the users.

The reasons why companies have to keep a history of the design data and related information are listed below:

- Viewing design history
- Reuse of the previous design
- Re-launching of old products [18]
- Retrieval of old information due to mistakes made in the current design
- Training new employees
- Supporting managerial decision making

There are two ways to track the history of a PN. First, users can select a PN while browsing through the structure and retrieve the old versions of the Node to view its history. Second, the users can indicate a specific time to view the structure at that specific time in history. Figure 9 shows the structural change on a portion of the PN structure with time. Under Node 'P,' five child-Nodes have been generated, one deleted, and two modified with the progress of time. T1, T2, and T3 show the different structures to the users at different points in time. The structure in the history is shown based on the two attributes, TimeInitiation and TimeTermination.

6.2 Version Control

There are two ways to manage the version of a model. One way is to maintain different versions of the model as a whole, and the second way is to maintain different versions of the elements [16]. For a model having many elements, it is usually not a good option to choose the criteria of managing versions as a whole system since the amount of data can grow dramatically, especially in a dynamic system where the design changes so often. Also for a model having a large amount of information, even if the logic is simple, the size of the data is likely to overwhelm the storage capacity of the computer system in a short period of time. The holistic management of a version can best be used for small individual elements. In the cases when a system rarely changes, managing versions in parts of a system can impose more loads on the version management system than maintaining the whole body of individual elements. Nonetheless, from the maintenance point of view, it is better to maintain whole copies of different versions when the capability of storage allows. Maintaining versions based on partial changes is in general more complicated because the system has to track the changes according to the version changes even at the component level. But for a large model having many elements of data within, it is necessary to have such a system. The PN system is a dynamically evolving structure. So it needs to update the version of each PN when critical changes occur in it. The architecture of the PN structure uses a combination of both holistic and partial maintenance of versions of a model. The PN system applies a holistic maintenance to the component level changes, and it applies partial

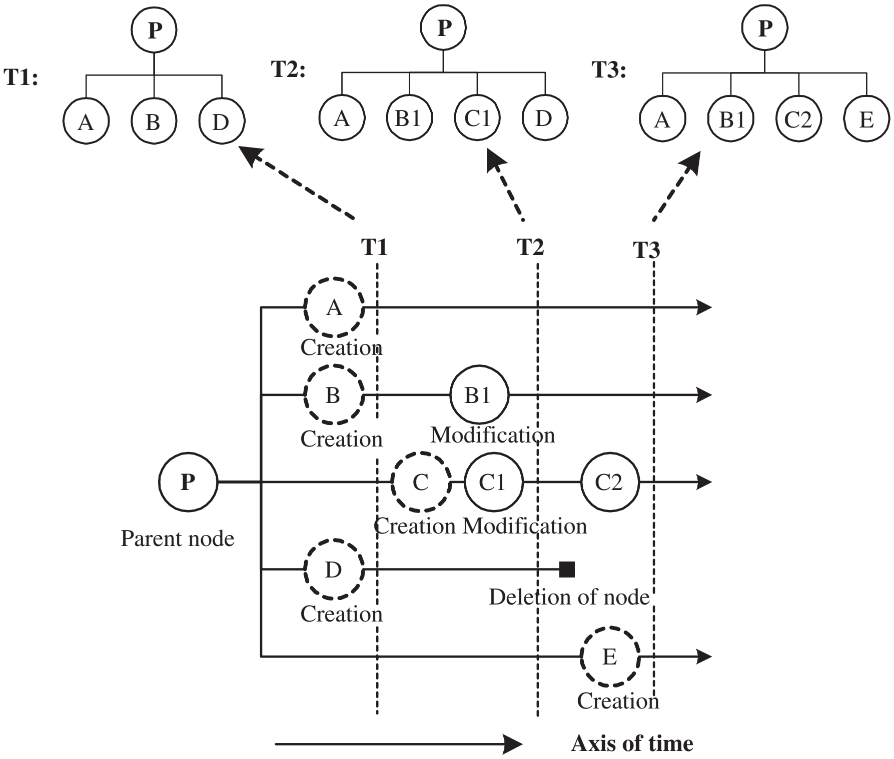


Figure 9. Change in the node structure with time under the same parent-PN.

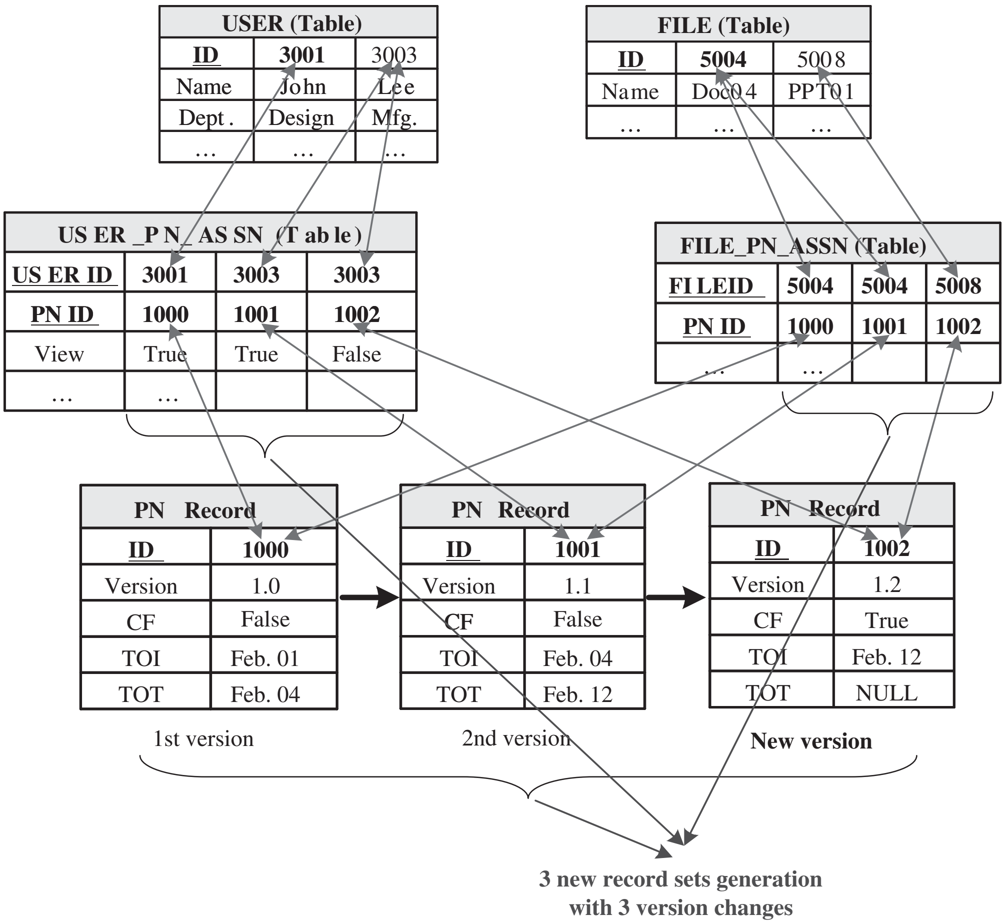


Figure 10. Changes in associated information based on upgrading versions of a PN.

maintenance for content having many subcomponents (sub-PNs or data) within. The data history of any PN should be maintained based on the version changes. Figure 10 shows the changes in information related to a PN when the version changes. Whenever a PN upgrades its version, the system generates one more instance of the associated information based on the new version. In Figure 10, the first version shows a PN created with a user and a file associated with it. The second version shows the status when the associated user changes; and the third version shows the record when an associated file changes.

6.3 Design Reuses

One of the main advantages which the small and medium enterprises (SMEs) and division of a big company will get from using the PN structure is the reuse of previous designs. As Figure 11 shows, 80% of design knowledge can be reused in the design process [19].

In the context of the PN structure, reuse of previous design history is equivalent to reusing the previous PN in a current activity. Creating a PN is also equivalent to creating a child-PN unless it is a PROJECT-PN. After finding an appropriate PN using the search operation, users can then copy this PN as a child of another PN, instead of creating a new one. The procedure of reuse is

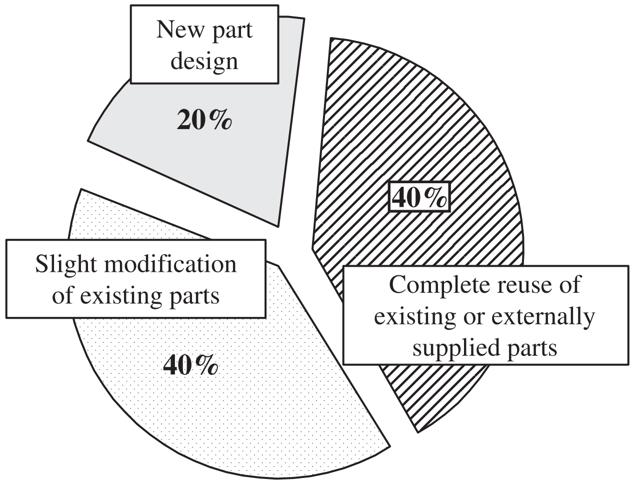


Figure 11. Design reuse of parts in OEMs.

described in Figure 12. As can be seen from the figure, when a user copies a PN, the system will generate a new PN with a unique ID, and the attribute ParentID of the new PN will be linked to the ID of the targeting PN (2 in Figure 12). If this PN has children associated with it, they will also be copied together with the copied PN.

The history of the copied PN can always be retrieved from the previous one (1 in Figure 12). All the information (i.e., database instances or files) associated with the reused PN will be duplicated and the new ID will be changed for the new PN (4 in Figure 12).

7. Management of Contents

7.1 User Management

There are two contradictory issues in product data management concerning the access privilege control of the user. It is a well-known fact that transparency and easy access to product data will boost the understanding of the product for all users and will increase the productivity of the team. In real-world practice, however, most of the design data is company property and, therefore, mostly confidential. But some information is shared with suppliers on a need basis. When it comes to a multicompany and multiuser project, successful control between these two conflicting issues is one of the key concerns. Hence, providing the means for a precise control on access to information has been an important consideration in the design of the PN structure. The user is one piece of information associated with a PN, which has information such as name, email, and the company associated with it. In general, there are many different types of users who may use the PN system. For example, in a design team at a manufacturing company, there is a leader, design team members, and most likely, a supervisor. In each division or department of a company, there are various hierarchy-based roles for employees, who all have different authority statuses. Well-defined roles and privilege management are necessary not only in intercompany collaboration processes but also in intracompany processes. There are many users associated with a PN and vice versa. In such a case, creating an association table to link those two relations (i.e., USER and PRODUCTNODE) is a general practice. Figure 13 shows the relationship between the USER table and the PRODUCTNODE table.

There are six main categories which have been defined to represent different types of users. With a combination of user types and accessibility to product data, the PN system can simulate and generate roles of users similar to those of real-world companies. Administrator (Group

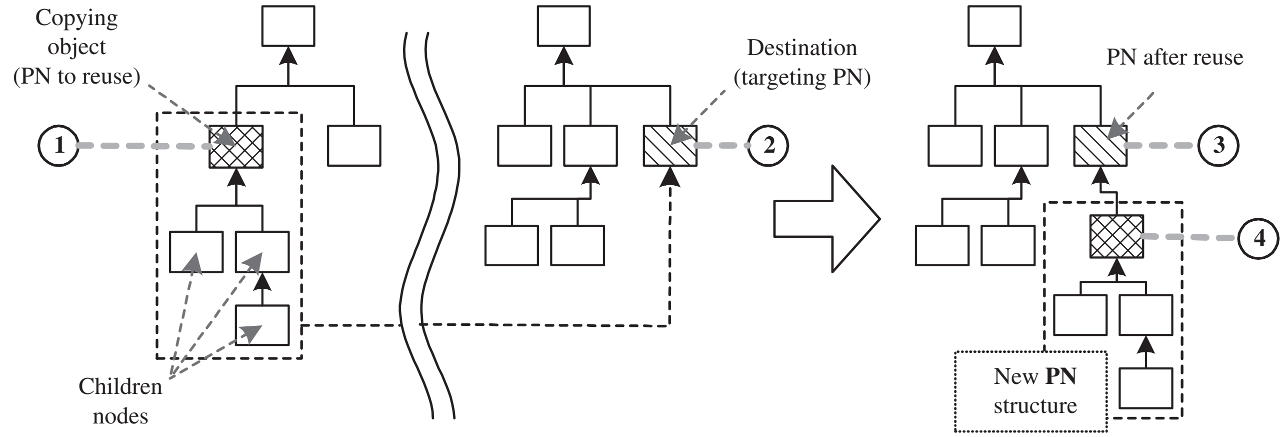


Figure 12. Schematic diagram of reusing a product node.

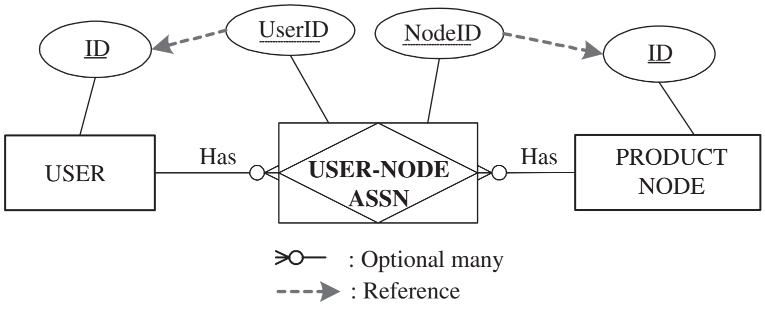


Figure 13. ER diagram of user–node (PN) associations.

Table 3. Privilege of users and the scope of control.

User type	Privilege	Roles/description
Administrator	+ Creation of projects, assign managers on projects	Creation of users, creation of a project, assigning a manager to a project, changing user type
Manager	+ Creating and assigning users on a project, changing privileges of users assigned on the project	Manager is a user who manages a project in general. This user has a privilege to read and write. Managers can assign users to a project from the list of users. Managers also assign privileges to users of a project.
Designer	+ Initial assignment to read a project, viewing/finding users	Designers are the actual workforce in performing a design task. Initially, they will be given viewing and change requesting rights on the node assigned
Vendor	+ No initial privilege	Vendors are people from outside a company. Manager can assign privileges (up to a designer) to this user but no privilege is assigned initially.
Viewer	+ Messaging with an account	Users having minimum privilege, viewing privilege can be assigned to the user
Guest	Limited messaging	Potential users

+ Means that the privilege is transferring from the user type below.

Administrator) is a user who has all the privileges and capabilities to control the users and projects in a group. The user types and privileges are described in Table 3. The user types are defined in a database table, USERTYPE.

7.2 File and Document Management

One of the problems in a multiproject-oriented business environment is streamlining the flow of information through the course of a project. A majority of product information exists in the form of documents. As a project progresses, there are frequent exchanges of documents between users and groups across companies and also within a company.

In a paper-based system, it is difficult to systematically or concurrently maintain these paper documents and make sure that the right document reaches the right person at the right time. The file management system is built on the PN structure. Most of the electronically formed data can be represented as a file. File definition in a database has a one-to-many relationship with a PN. As can be seen from Figure 14, a file can have only one PN related to it, but a PN can have many files associated with it.

It is important to have a separate copy of the file for each PN to prevent any possible problem since the data can change dynamically as the project proceeds. The relation, FILE, is defined in the database as shown in Table 4. The file system built around the PN system gives the users many advantages as listed below:

- Central management of files with the use of a database system
- Linking files with a specific product definition
- Access control based on the user type and user–node association relationship
- Systematic version control

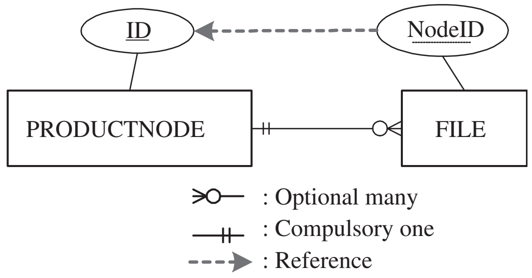


Figure 14. FILE and PRODUCTNODE.

Table 4. Definition of database table, FILE.

Field name	Definition
ID	ID, Primary Key
Version	Version of the file
Name	File Name; Saved as [ID]_[Random number]_[Name].[Ext]
PreviousID	ID of the Previous Version
NodeID	Associated Node's ID, Foreign Key
CreatorID	Creator's ID, Foreign Key
FileTypeID	ID of FILETYPE table, Foreign Key, referencing from the table, 'FILETYPE'
FileExtension	File extension
FileLocation	Physical location of the file
DateOfCreation	Date of Creation
Description	Description of the file
CurrentFlag	Flag to indicate if the file is the current one or not. 1 (yes) or 0 (no)
Keyword	Keyword will be used in searching

(________): Primary Key; (_______): Foreign Key.

Based on the assumption that the number of files to be used could be very big and the size of each file is also substantial, direct access to the file system was chosen in our application to ease the problem of maintaining the database and to use the database resources efficiently. The database system plays the role of a bridge between the users and the file system. The types of files are defined in a separate database table, FILETYPE as in Table 5. The table, FILETYPE, has a field to indicate the type of the file and the description within to explain the file. File type is used to identify where to locate the file in each associated PN. FILETYPE table identifies

Table 5. Definition of FILETYPE table.

Field name	Definition
ID	Primary Key
TypeName	File Type

the type of file stored in the system. Since most information exists in the form of files, the table indicates the kind of information the file has. Based on the reference to the ID of the table, FILETYPE, the system can identify the use of the files saved in the PN.

8. Implementation

8.1 Representation of PN

The overall program architecture of the PN system is shown in Figure 15. The prototype implementation was made to test the feasibility and the basic operations necessary to the PN system architecture. It consists of three layers – application layer (front-end), middle layer, and database layer. The information about the PN structure is retrieved from the database and stored in a neutral Extensible Markup Language (XML) format in the middle layer. This information is then presented to the user through the application layer consisting of Java Server Pages (JSP™), JavaScript™, and Hype-Text Markup Language (HTML). The application layer handles all the user interactions and updates its view accordingly. The default view of the application layer shows the root node (PROJECT-PN) with the corresponding node details. The root node can then be expanded to view the child-nodes associated with the project. To get the required information about the child-nodes, a query is made to the database with the node ID (i.e., the Project ID) as the query parameter. Subsequently, the information about the child-nodes (i.e., the nodes which have the passed query parameter (Project ID) as its parent ID) is then retrieved from the database. For each such child-node, a query is again made to the database system to find the nodes at the next level (i.e., child's child-node). This recursive query traverses through the entire database until it reaches the leaf-node (leaf-node: node having no child-node under). This information is made available to the middle layer, where the Microsoft® XML Parser (Microsoft® XML Parser comes with Microsoft® Internet Explorer™) puts all the retrieved information into XML format. The application layer uses the node structure in the XML format and then displays it in HTML format as in Figure 16.

8.2 Basic Operation

A child node can be added to an existing node using the 'Add Node' command as shown in Figure 16. The user can fill in the details for the child-node. This will then be added as a child to the node 'Drive Train.' Using this 'Modify' functionality, the user can make changes to the node details. The user first needs to select the appropriate node on the left side frame and then proceed to make the changes. The 'File Management' functionality allows the user to associate files with a particular node. The files are stored in the server under a folder structure, which reflects the node structure.

9. Conclusions

Various methodologies which deal with the representation of product information and their interactions have been discussed. Based on an analysis of the existing

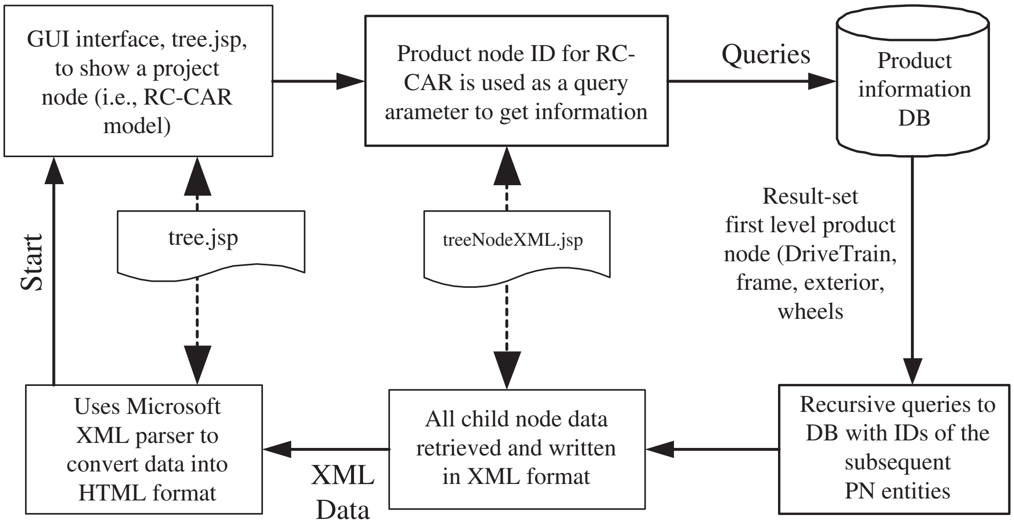
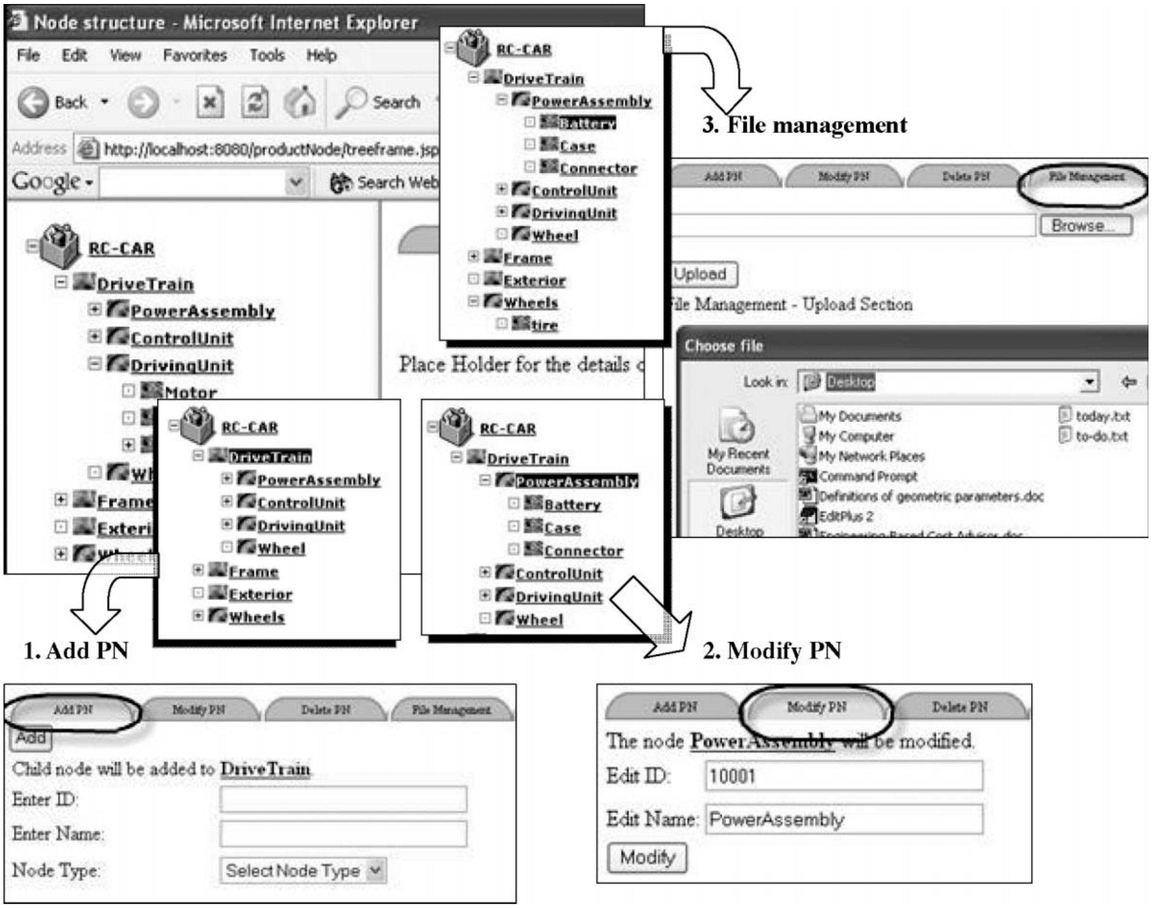

GUI interface, tree.jsp, to show a project node (i.e., RC-CAR model)
Product node ID for RC-CAR is used as a query arameter to get information
Queries
Product information DB
tree.jsp
treeNodeXML.jsp
Start
Result-set first level product node (DriveTrain, frame, exterior, wheels
Uses Microsoft XML parser to convert data into HTML format
XML Data
All child node data retrieved and written in XML format
Recursive queries to DB with IDs of the subsequent PN entities

Figure 15. Basic structure of the PN system.



Child node will be added to DriveTrain
Enter ID:
Enter Name:
Node Type: Select Node Type

The node PowerAssembly will be modified.
Edit ID: 10001
Edit Name: PowerAssembly

Figure 16. Expanded structure and basic operation view of PN.

systems, problems arising due to the static structure of information representation as well as non-concurrent operations have been identified. To address these issues, a new methodology of PN has been introduced. The dynamic structure of the PN system enables the synchronization of a project initiation and the actual progress of the project. Database structures of the PN system and the logic of transactions of the use have been shown in detail in this work. Manipulation of PN, such as changing the product structure, history tracking, reuse of previous design, file and document management, and user management show the 'structured flexibility' of the system. The prototype application shows the basic functions required by the PN system and the viability of the system structure.

Acknowledgments

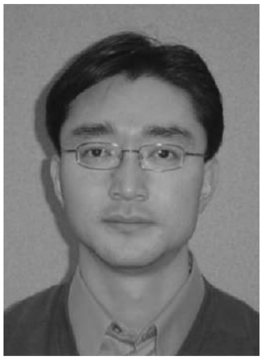
The support of the 21st Century Research and Technology Fund and the National Science Foundation Partnership for the Innovation Award to PRECISE (Purdue Research and Education Center for Information in Engineering) is gratefully acknowledged.

References

1. Frank, A., Sellentin, J. and Mitschang, B. (2000). TOGA – A Customizable Service for Data-centric Collaboration, *Information Systems*, **25**(2): 157–176.
2. Prasad, B., Wang, F. and Deng, J. (1997). Towards a Computer-Supported Cooperative Environment for Concurrent Engineering, *Concurrent Engineering: Research and Applications*, **5**(3): 233–251.
3. Qiang, L., Zhang, Y.F. and Nee, A.Y.C. (2001). A Distributive and Collaborative Concurrent Product Design System through the WWW/Internet, *International Journal of Advanced Manufacturing Technology*, **17**(5): 315–322.
4. Miller, E. and Koucky, S. (2002). Market Consolidation Raises Executive Awareness of PLM, *Machine Design*, **74**(14): 56.
5. Lingblom, M. (2001). EDS Forms Unit Devoted to PLM Solutions, *Computer Reseller News*, 10/8/2001(966): 22.
6. Naumann, T., Vajna, S., Speck, H. and Ende, A. (2002). Relationship Between Process and Product Structures – A New and Flexible Approach for an Integrated Dynamic Process Management, In: *Proceedings of DETC '02*, Montreal, Canada, September 29–October 2.
7. Gane, C. and Sarson, T. (1979). *Structured Systems Analysis: Tools and Techniques*, New York, NY 10019: Improved System Technologies, Inc.

8. O'Donnell, F.J., MacCallum, K.J., Hogg, T.D. and Yu, B. (1996). Product Structuring in a Small Manufacturing Enterprise, *Computers in Industry*, **31**(3): 281–292.
9. Szykman, S., Racz, J., Bochenek, C. and Sriram, R.D. (2000). A Web-based System for Design Artifact Modeling, *Design Studies*, **21**(2): 145–165.
10. Gorti, S.R., Gupta, A., Kim, G.J., Sriram, R.D. and Wong, A. (1998). An Object-oriented Representation for Product and Design Processes, *Computer Aided Design*, **28**(11): 489–501.
11. Liang, W.Y. and O'Grady, P. (1998). Design with Objects: An Approach to Object-oriented Design, *Computer Aided Design*, **30**(12): 943–956.
12. Hameri, A.P. and Nitter, P. (2002). Engineering Data Management through Different Breakdown Structures in a Large-scale Project, *International Journal of Project Management*, **20**(5): 375–384.
13. Warfield, J.N. (1974). *Structuring Complex Systems*, Battelle Memorial Institute, Columbus, Ohio.
14. Date, C.J. (2000). *An Introduction to Database Systems*, Addison Wesley Longman, Inc., Boston, MA.
15. Kumar, V., Burns, D., Dutta, D. and Hoffmann, C. (1999). A Framework for Object Modeling, *Computer Aided Design*, **31**(9): 541–556.
16. Proper, H.A. (1997). Data Schema Design as a Schema Evolution Process, *Data & Knowledge Engineering*, **22**(2): 159–189.
17. Crabb, H.C. (1998). *The Virtual Engineer, 21st Century Product Development*, New York: SME/ASME Press.
18. Stock, J., Speh, T. and Shear, H. (2002). *Many Happy (Product) Returns*, Harvard Business Review, pp. 16–18.
19. Rezayat, M. (2000). Knowledge-based Product Development Using XML and KCs, *Computer-Aided Design*, **32**(5): 299–309.
20. Prasad, B. (1999). *Concurrent Engineering Fundamentals: Integrated Product and Process Organization*, Vol. I, New Jersey: Prentice Hall PTR.
21. Prasad, B., Wang, F. and Deng, J. (1998). A Concurrent Workflow Management Process for Integrated Product Development, *Journal of Engineering Design*, **9**(2): 121–135.

Charles Chan-Woo Chung



Dr Chung received his PhD in Mechanical Engineering from the Purdue University. As a graduate student, he was a NSF/IGERT Fellow. He received his MS from Yonsei University, Korea in 1997. His research focus was on Rapid Tooling/Prototyping, CAD, Product Informatics, Product/Process Data Modeling, Knowledge Database Modeling, Engineering Advisory Systems, and Product Lifecycle Management System Design. Dr Chung is currently a Senior Consultant at Samsung Data Systems (SDS) in Korea.

Jun-Ki Choi

Jun-Ki Choi is Fredrik Andrews Environmental Fellow in the School of Mechanical Engineering at Purdue University. He received his MS degree from the University of Michigan, Ann Arbor in 1999. He is a PhD candidate in the School of Mechanical Engineering at Purdue University. His current areas of research include Design Methodology, Decision Making Analysis, Design for Environment, Environmental Supply Chain Management, and Collaborative product/process design. His research backgrounds also include machining, manufacturing processes, rapid prototyping, and tooling.

Harshal Patwardhan

Harshal Patwardhan received his MS degree in Industrial Engineering at Purdue University in 2004. He received his Bachelors Degree in Mechanical Engineering from Pune University, India in 2000. His current interests are in the areas of Product and Process Development, Cost Estimation, Supply Chain Management, and Logistics. He is currently working at Alpha Sigma Consulting, LLC at Chicago as a Consultant.

Karthik Ramani

Dr Ramani is a Professor in the School of Mechanical Engineering at Purdue University. He earned his BTech from the Indian Institute of Technology, Madras in 1985, an MS from The Ohio State University in 1987, and a PhD from Stanford University in 1991, all in Mechanical Engineering. He has been awarded the Dupont Young Faculty Award, the National Science

Foundation Research Initiation Award, the National Science Foundation CAREER Award, the Ralph Teetor Educational Award from the Society of Automotive Engineers, the Outstanding Young Manufacturing Engineer Award from the Society of Manufacturing Engineers, and the Ruth and Joel Spira Award for Outstanding Contributions to the Mechanical Engineering Curriculum. Dr Ramani's current projects are in the area of product and process design, information retrieval and management, and rapid tooling for future design and manufacturing systems.