OBJECT HIERARCHIES TO AID REPRESENTATION AND VARIANT DESIGN OF COMPLEX ASSEMBLIES IN AN AGILE ENVIRONMENT

Vishwanath Ramabhatta, Li Lin, Rakesh Nagi[1]

Department of Industrial Engineering, 342 Bell Hall

State University of New York at Buffalo, Buffalo, NY 14260

Telephone: 1-(716)-645-2357 x 2103

Fax: 1-(716)-645-3302

E-mail: nagi@eng.buffalo.edu

[1] To whom correspondence should be addressed

ABSTRACT

Group Technology (GT) has been recognized for its ability to reduce the long lead times of new product development and process planning. The work presented here focuses on developing a product model that can describe an end-product as an aggregation of constituent sub-assemblies and components in an open data-structure so as to enable development of end-product variants using variant design. The environment is an agile enterprise where manufacturing partners share product related data to come-up with new, customized, high quality products at minimal lead-times. First, a systematic procedure to develop a product model as an object hierarchy is presented. Then, a retrieval system is proposed to accomplish the search of similar component or assembly objects in a distributed agile manufacturing setting. With access to partner databases, new products through such a product model can be developed at reduced lead times. Thus the utilization of the available resources in the agile framework can be maximized.

Keywords: Agile Manufacturing, Group Technology, Object Hierarchies, Bill-Of-Materials, Variant Design, Similarity Search.

1 INTRODUCTION

Today's global market is characterized by high competition, mass customization, high product variety and very low product batch sizes. It is important to realize the influence this environment has on the various design and manufacturing activities in an organization. Examples of these activities include product development, order entry, forecasting of demand for components to be assembled, assembly planning, after sales service, purchasing and costing. Long design and manufacturing lead times are typical of new product development and have to be minimized in order to get products to the market at the earliest, with reduced cost, and hence enable the company to remain competitive.

An agile manufacturing unit is a customer-centered organization geared for rapid delivery of new, high quality, and easily customized products. Collaborating with qualified partners with the necessary resources and capabilities in this virtual company setting, leads to smooth flow of product, process and business-related information (Minis *et al.*, 1993). However, this collaboration is enabled only by an effective Information infrastructure supported by efficient and functionally appropriate Information Technology (IT) solutions. These solutions should support proven technologies such as Group Technology (GT) and Concurrent Engineering (CE).

GT as a philosophy promotes identification and grouping of similar parts into part families and has since long been found beneficial in increasing the productivity of activities such as variant design and variant process planning. One way by which GT identifies part families is through the Classification and Coding technique (C&C). In C&C, a single piece part is represented as a string of alphanumeric characters that represent important design and manufacturing related attributes. Since C&C was an intensive process in the past, the applicability of GT remained limited.

Recently, automated C&C for piece parts has somewhat alleviated this problem. Despite these developments, at least two concerns remain: (i) how is the classification and coding scheme developed for a different class of piece parts, and more importantly for complex assembled products, and (ii) how can IT solutions enable and facilitate the modeling, representation and retrieval of assembled or piece parts. Finally, it would be of interest to base CE and all activities related to simultaneous design and manufacturing activities or variant customization on this IT-rich product model.

Product modeling techniques allow generation of an information reservoir of complete product data to support various design and manufacturing activities. When developing a comprehensive product model, apart from the basic requirements of representing actual data, facilitating product documentation, and offering decision alternatives another important issue that needs to be considered is the distributed and heterogeneous environment in the agile network of partners. The latter issue relates directly to the openness of the product model for access to relevant information by participating partners.

The work presented here details a systematic procedure to develop a product model that supports variant end-product design. In an agile environment, it is assumed that every participating partner has access to such a product model. Using this model, variant end-product design can be performed by searching sub-assemblies/components for ready use or customization from partner databases. A retrieval engine that retrieves information from such a product model is also discussed.

The organization of the paper is as follows. Section 2 briefly presents a literature review. Section 3 details the formal procedure for developing the object hierarchy scheme. Illustrations are presented at necessary points to make the concepts clear. Section 4 presents a retrieval engine that performs similarity searches to aid variant end-product design. The last section concludes the paper with recommendations for future work.

2 LITERATURE REVIEW

GT is a philosophy that implies the notion of recognizing and exploiting similarities by performing like activities together, standardizing similar tasks and efficiently storing and retrieving information about recurring problems (Hyer and Wemmerlov, 1985). It incorporates classification theory and practice in an industrial enterprise, applied in the design process to reduce product proliferation; and in manufacturing to enhance productivity, throughput, quality and profitability (Tiecholz and Orr, 1987; Allen, 1994). The benefits of GT most relevant to agile manufacturing are design retrieval, part classification, design standardization, variant process planning, and design and manufacturability evaluation.

As mentioned in the introduction, GT implementations that support variant manufacturing activities have employed Classification and Coding in the past. A GT code is a string of alphanumeric characters that represent important design and manufacturing related attributes. Many classification and coding schemes have been developed, such as Opitz (Opitz, 1970), DCLASS (Allen and Smith, 1980), MICLASS (OIR and Multi-M, 1986) and KK3 for mechanical parts. Although the format and coding classification of these codes are different, they all encapsulate information on six basic part characteristics: (i) Main Shape, (ii) Form features on

the main shape, (iii) Feature position, (iv) Dimensions, (v) Tolerances, and (vi) Material. For electrical parts, few coding schemes have also been proposed such as the one developed for printed wiring boards specially to facilitate CAD/CAM applications in their design and manufacturing (Roedecha and Bao, 1985, Ham *et al*., 1986) and the other developed for complex electrical/mechanical parts called microwave modules (Harhalakis *et al*., 1992). However, GT applications have been slow due to considerable effort required in coding of a company's part database and which often leads to inconsistencies and errors (Kinsey, 1992). Recent works that address this issue have involved automatic generation of GT codes such as the one based on Opitz scheme for mechanical parts (Shah and Bhatnagar, 1989), based on DCLASS for rotational parts (Henderson and Musti, 1988), and from a 3-D CAD model for Lockheed sheet metal parts (Bond and Jain, 1988). For mechanical and electrical parts, an automated code generating system that makes use of the PDES/STEP (Product Data Exchange using Standard for the Exchange of Product model data) as an input to yield mechanical and electrical GT codes and critical design information has been developed (Harhalakis *et al*., 1992). Some relevant work in similarity measures and coefficients have been proposed in the area of cluster analysis.

Automatic generation of GT codes still cannot alleviate the problem of a new variety of piece parts whose attributes do not fall under pre-defined classifications. Also developing GT codes for assemblies is complex because of the following reasons: (i) Classifications are present not only because of attributes, but also due to the constitution that make up end-products, (ii) there is a high variation in the constitution of end-product variants, and (iii) there exist relationships between sub-assemblies/components at various levels in the product structure. Modeling the GT classification scheme using database abstractions can eliminate some of the above problems.

With the emergence of engineering database as a critical factor in successful automation of a manufacturing enterprise, research has shifted towards developing product models that can be then mapped onto an enterprise database. This can be accomplished using the available database technology to store GT data, and over which GT applications can be built (Billo *et al.*, 1987). Modeling GT schemata using database principles like aggregation, classification, generalization, and associations and then implementing them on a standard database repository such as a RDBMS or a ODBMS has the benefit of flexibility in modifications without affecting the applications. The queries can be done using industry standards such as SQL for relational databases and OQL for object-oriented databases (Billo *et al*., 1988). In order to achieve many of the Computer Aided Design (CAD) and Computer Aided Manufacturing (CAM) related benefits, the engineering database should be tightly integrated with GT.

Many product models have been developed until now to support various applications like process planning, bill of material, etc., (Krause *et al*., 1993; Chung and Fischer, 1994). One specific work proposes a generic bill-of-material, a product model that captures the structure for all variants of a end-product family (Hegge and Wortmann, 1991). With this generic bill-of-material, it is possible to describe all the variants from the model. We use this concept in developing our product model. Using object-oriented principles during product modeling has the benefit of capturing the real world objects in a way that allows easy implementation into the database. Another work that addresses the representation of end-products proposes a local product model and is specific to the process planning application (Usher, 1993).

In this paper we make of the object-oriented principles that have been used in representing GT classification and coding techniques previously (Billo and Bidanda, 1995). However, it is our interest here to derive the object product model directly from existing and available product data.

The search module developed in the recent work (Iyer and Nagi, 1994) forms the basis for the retrieval engine which is presented briefly. The retrieval system being developed incorporates searching and ranking procedures for similar part matching based on database queries. The procedures developed in the previous work depend on the GT codes and have to be restructured to accommodate an object-oriented GT model.

3 PRODUCT MODELING

A procedure to develop a product model that is used to represent the end-product, a complex assembly, as an aggregation of constituent sub-assemblies and components with their significant attributes is detailed in this section. The systematic procedure consists of sequential steps as shown in the Figure 1.

1. General Product Structure: end-product variants that are of interest or are identified to be critical are gathered and documented in a format that allows easy identification of real objects.
2. Clustering: similar sub-assemblies and/or components are grouped to identify high level classification based on constitution.
3. Generic Bill of Material: the general product structure is represented as a graphical product explosion using object-oriented modeling principles like abstractions, generalizations, and object identification (and definition).
4. Attribute collection: attributes for each of the objects in the generic bill-of-material are collected in a format that provides the information that are essential for the object's existence.
5. Classification: further classifications based on attribute values are identified. In the process, where necessary, attributes in the class object are separated, associations are modeled, and software abstractions are identified.

6. Object model: the generic bill-of-material is refined by modeling classifications, associations and abstractions to obtain the final product model.

During the process of object modeling however, there is some iterative work between the attribute collection and the classifications. The details with recommendations for each of the steps enumerated above are provided.

A software prototype that can enable automatic generation of the product model is being developed in Visual C++ environment and using Object Store PSE for C++ database for storing persistent data. We assume that the bill of material (BOM) is usually available before the design of the product model. To demonstrate the transition of BOM data from legacy systems (such as MRP), we read this information from relational tables in Microsoft Access. Some of the implementation aspects are briefly presented for completeness.

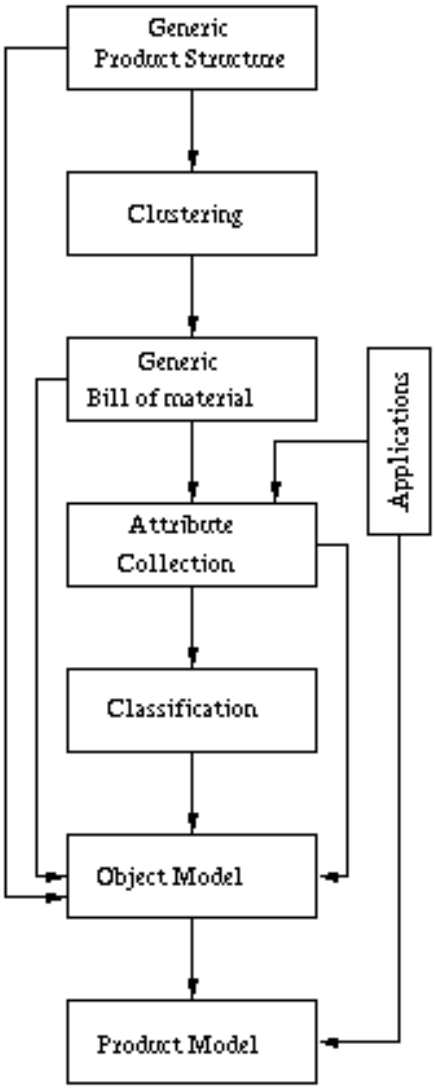

Figure 1: Procedure for developing a product model

3.1 General Product Structure

The purpose of the general product structure is to capture the sub-assemblies and components forming the different critical assembly variants. Using this general structure as a template, new product variants can be designed and customized.

The approach in formulating this template consists of beginning with a small number of existing representative end-product variants from the company's database. The variants that are chosen must be representative of the products that form about 80% of the company's business volume and those that requires major design and manufacturing resources (Pareto analysis can be performed). This selection can often be guided by experts in marketing, forecasting, and design. The sources of these product information could come from the parts list, product structures, drawings, and bills-of-material. This is followed by taking a union of these variants to eliminate repetition, and come up with a compact description.

Figure 2 presents the indented format recommended for documentation. It is important to keep the bill-of-material type of hierarchy in this general product structure. All the individual sub-assemblies and components of the variants, irrespective of their being a '*type of*' should be documented in the product structure and then marked '**x**' against '*where or in which used*'. This serves two purposes: (i) it maintains completeness of information, and (ii) it does not allow non-systematic classification often biased to the needs of a specific application.

Next, the standard parts are identified in this product structure. The standard parts maybe: (i) those that are not designed but are rather specified, such as nuts, bolts, washers, etc., and (ii) sub-assemblies and components whose design changes are infrequent and variations are small. The standard parts (especially the latter) are shown shaded in the format. Keeping standard items in the structure helps during attribute collection and in modeling mutually exclusive relationships with non-standard items during object modeling (see step 6).

Sub – assemblies/components	M 1	M 2	M 3	M 4
1. Engine	x	x	x	x
1.1 Cylinder Block	x	x	x	x
1.1.1 Block	x	x	x	x
1.1.2 Crank Shaft	x	x	x	x
1.1.3 Oil Pump	x	x	x	x
1.1.4 Cam Shaft	x			x
1.1.5 Piston & Connecting Rod Assly	x	x	x	x
1.1.5.1 Piston Body	x	x	x	x
1.1.5.2 Oil rings	x	x	x	x
1.1.5.3 Connecting Rod	x	x	x	x
1.1.5.4 Bearings	x	x	x	x
1.1.5.5 Caps	x	x	x	x
1.1.6 Push Rod	x			x
1.1.7 Follower/Tappet	x			x
1.2 Cylinder Head	x	x	x	x
1.2.1 Combustion Chamber	x	x	x	x
1.2.2 Camshaft		x	x	
1.2.3 Rocker Arms	x			x
1.2.4 Valve guides	x			x
1.2.5 Lash adjustor		x	x	
1.2.6 Valves	x	x	x	x
1.3 Spark Plugs	x			x
1.4 Glow Plugs		x	x	

Figure 2: General Product Structure

While developing a software prototype, we assume the bill-of-material information for every instance that is selected for analysis is available in a relational database with the following attributes:

1. Level in the hierarchy
2. Part Number (and Drawing Number)
3. Product Name (Generic) e.g., Column
4. Product description (Specific) e.g., Auxiliary Column
5. Manufactured / Purchased

The bill-of-material information with the above attributes, is read from legacy storage such as relational tables and stored in the OO database. Standard parts are filtered by comparing with a knowledge base or by comparing the part/drawing numbers prior to storing. Option to store the standard sub-assemblies/components of the second type described earlier is permitted to model mutually exclusive relationships involving them. This is done for each representative sample. All storage in our software are list implementations in order to preserve the bill-of-material information read from legacy sources. The individual lists of different representative samples are then processed to form an unionized bill-of-material called the General Product Structure. The General Product Structure is output into a grid format as shown in the figure 2. It is possible that the same object resides in multiple levels across different sub-assemblies and across different instances. Such issues are partially resolved by prompting the user to define the most suitable location with the option of duplicating the object at another level. One has to be careful while duplicating objects as they may lead to inconsistencies later during attribute collection (refer to section 3.4).

3.2 Clustering

Individual sub-assemblies, starting from the lowest level are classified based on their composition. This grouping could be done similar to forming part families using Single linkage clustering algorithm (SLCA) or Average linkage clustering algorithm (ALCA) (Seifoddini and Wolfe, 1986). Clustering help in refining the general product structure by adding objects that help in describing classification of sub-assemblies in the product structure. They also help in collecting classification specific attributes (refer section 3.4).

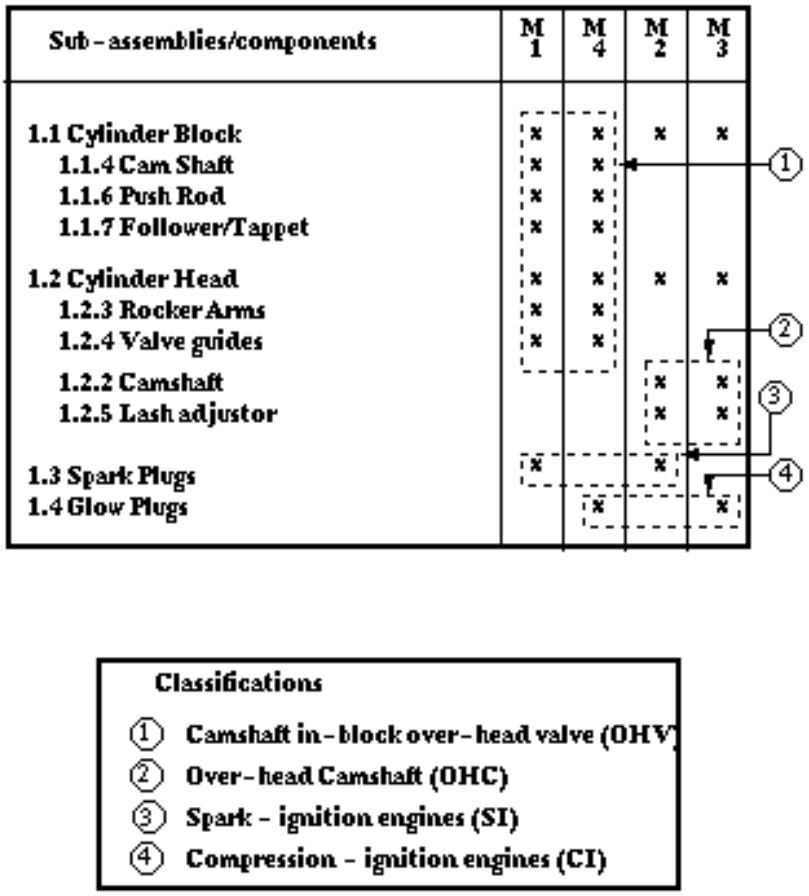

Figure 3: High level classification using clustering

A systematic procedure consists of subjecting all sub-assemblies at the lowest level to the clustering algorithm and then moving up the hierarchy to the next level and repeating the process. This provides a better and clearer description of all sub-assembly variants at all levels. Standard parts, especially the second type, described earlier are helpful in identifying clusters and hence we recommend permitting them while filtering standard parts (refer back to section 3.1). Figure 3 shows the result of clustering.

During implementation, the general product structure in the grid format is subjected to a clustering algorithm which performs two functions: (i) navigates the product hierarchy in the general product structure and fixes the starting and ending objects (both at the same level in the hierarchy) within which the clusters are identified and (ii) identifies clusters. Figure 4 shows how the clustering algorithm works by identifying clusters first in the lower levels and moving towards the top level in a hypothetical product hierarchy. In the figure, O1, O2, etc., represent the objects and the dotted rectangles represents the location where the clustering algorithm is applied.

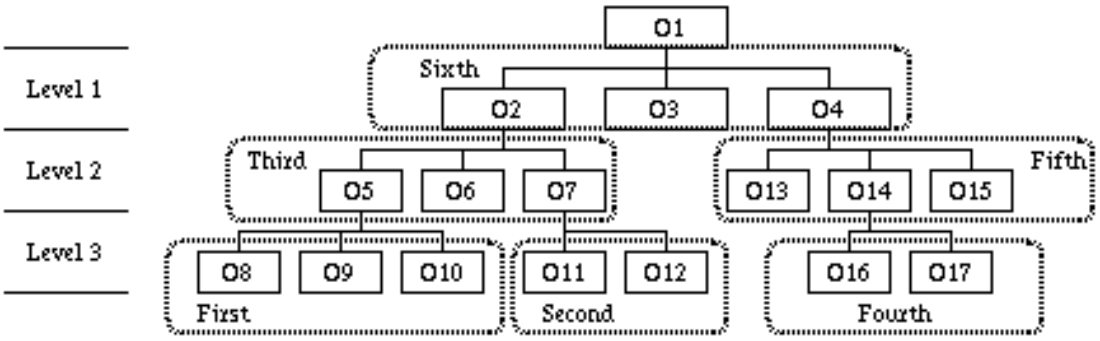

Figure 4: Clustering process in a product hierarchy

When clusters are identified, the user is prompted to give names to the classification. These names permit addition of classification objects at appropriate locations the general product structure. Note that the addition is done to the stored general product structure list, while the grid format is subject to the clustering algorithm.

3.3 Generic Bill of Material

The Generic Bill of Material is a graphical representation of the general product structure, refined with the high level classifications obtained during clustering. It serves two purposes: (i) the refined product structure ensures data consistency and provides a good description of most end-product variants, and (ii) allows easy identification of objects for which attributes have to be collected.

To represent the relationship between objects, we have used symbols from Object Modeling Technique (OMT) wazzu (Rumbaugh *et al*., 1992). The relationships modeled at this stage are aggregations with multiplicity and generalizations. Figure 4 shows the generic bill of material.

In the implementation, a file structure displays all the objects that are contained in the general product structure. At this stage, only aggregation relationships are stored in the general product structure. However, we have generalizations due to clustering and the multiplicity relationship that need to be stored. These are discussed later under attribute collection.

3.4 Attribute collection

In object orientation, all real world objects are represented as classes and information that characterize these objects are in the form of attributes and operations. The generic bill-of-material has in it all the sub-assembly and component objects whose attributes have to be collected. Attributes that directly relate to a particular application should be considered. The types of attributes that typically need to be captured are:

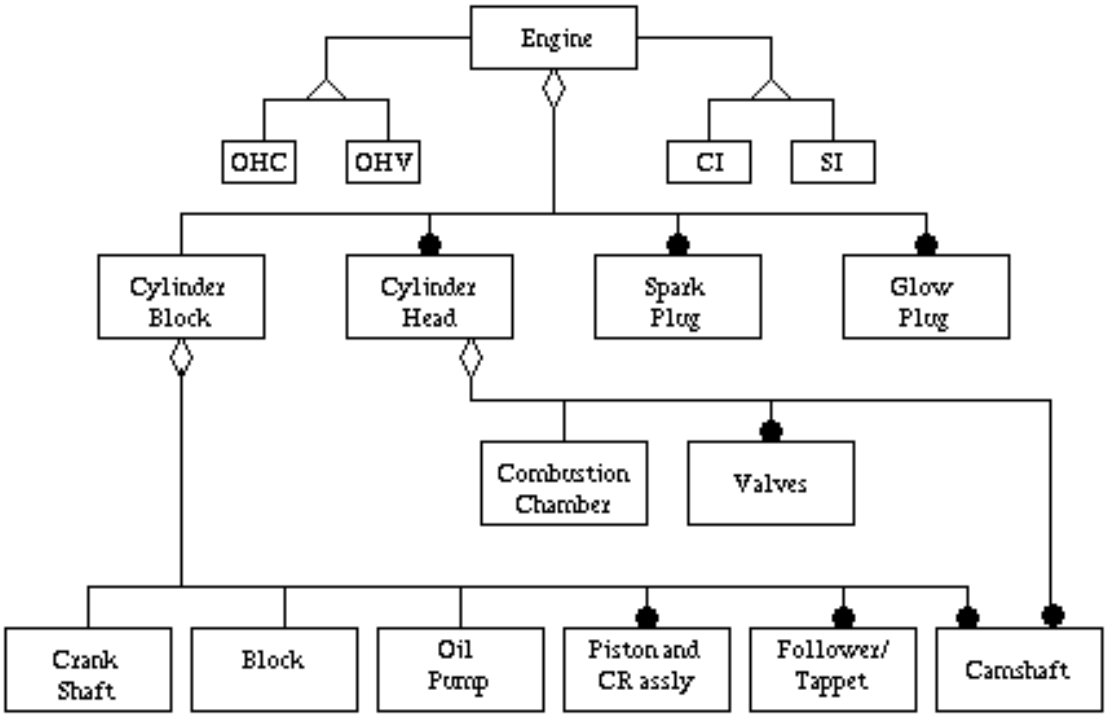

Figure 5: Generic Bill-of-Material

1. Physical Attributes: These are design and process attributes that are required for design activities. They are high level geometric and physical parameters like weight, height, dimension/envelope, and the process parameters like efficiency, working pressure, etc.
2. Material Attributes: These are material specific properties like the density, specific volume, supplier code, etc.
3. Key Quantity Attributes: These are information required for optimizing designs such as number of objects, volume, cost, etc.
4. Specific Attributes like the drawing numbers, and the like that provide links to other work process databases.

5. Application-specific attributes like the composition and number of, where used, etc.

The last three types of attributes are not directly derived from technical design and process oriented sources, but are nevertheless important to the designer while making optimizing decisions such as envelope, cost and manufacturability.

Figure 5 documents attributes for some objects that are identified in the general bill-of-material. While documenting, it is beneficial to represent the data type required during implementation (e.g., integer int or character char). It is also helpful to have a few examples annotated along the attributes as an easy guide to understanding (e.g., displacement 27479 cu in/min).

In the implementation, the collection of attributes for each object in the file structure or the general product structure list is user assisted. The user is presented with a form that collects attribute information. The attribute information is processed and added as an attribute list to an the object. The typical fields in the form are text field for attribute name, and check boxes for data types. When multiple instances of the same object occur at different levels in the product structure, the form for attribute collection should be presented only once to the user. This ensures data consistency. At this stage we rely on the user to use the attribute collection form to key in the generalization and multiplicity relationships for all objects. Classification specific attributes should also be added where ever necessary. This is later discussed in the section on Retrieval engine.

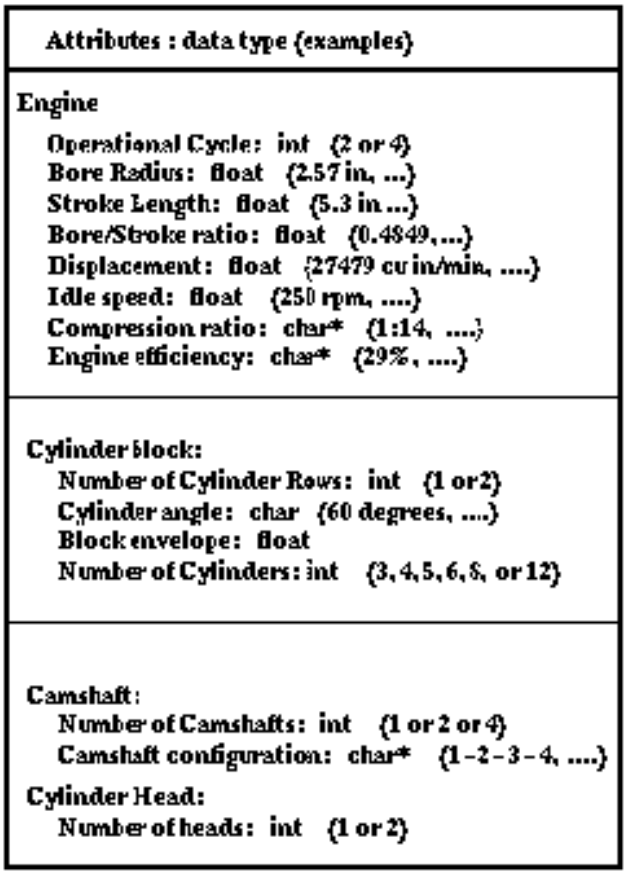

Figure 6: Attribute collection for the objects

3.5 Classification

Often, classifications exist due to the value an attribute takes. An example is the way shells are classified as thick-walled and thin-walled based on the length/diameter ratio. Note that clustering allows identification of high level classification based on an object's composition while classification obtained here is that due to one or more attribute values. With the identification of attribute based variants, it is possible to fully describe more end-product variants from the product model.

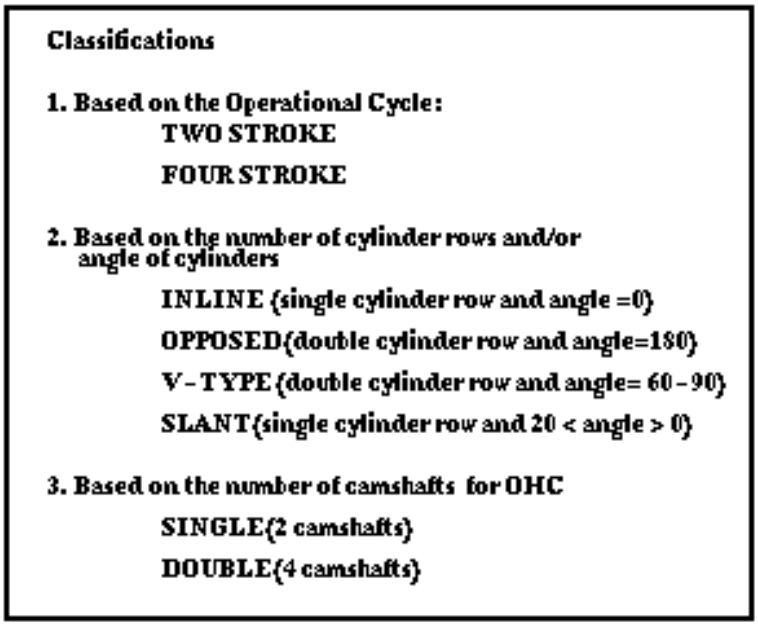

Figure 7: Classification based on attributes

Automating classification during implementation requires getting information about the **IF** *attribute* = *value* **THEN** *classification* at the attribute collection stage. These relationships then have to be processed and then the classification incorporated in the generic bill of material. At the attribute collection stage it is required to add additional attributes that define abstract classes which need to be added into the model to improve the products description. Our implementation at this stage is user assisted. User is prompted to add additional classes like the abstract classes and those that exist because of specific attribute values.

3.6 Object Model

The generic bill-of-material represents sub-assemblies and components as end-product decomposition using aggregation, generalization and multiplicity rules, which now has the potential to fully describe any end-product variant. Further refinement of this graphical structure is done using OMT principles such as inheritance, abstractions, and links and associations (Rumbaugh *et al*., 1992). The following rules, apart from others help refine the model.

To avoid redundancy, an object has to be linked to other objects that have relationships with it. Different objects that are not derived from the same parent, might share a set of similar attributes. In that case, the similar attribute set should be removed and modeled as a separate class object with associations to the objects sharing that set of attributes. Access methods must be determined in a way that all objects can be accessed from all other objects. The decision is dictated by what information will be needed, which object is often queried and how often the access routes will be used. We recommend that the access be bi-directional to allow movement both up and down the hierarchy. Multi-level jumps in accessing objects should not generally be provided as they would violate openness of the data-structure essential to future integration. Care should be taken to not give multiple access to an object from another object, in order to keep the data-structure consistent. In the object-oriented programming language, like C++, this could be achieved by providing forward and backward pointers. The final object model is shown in the Figure 7.

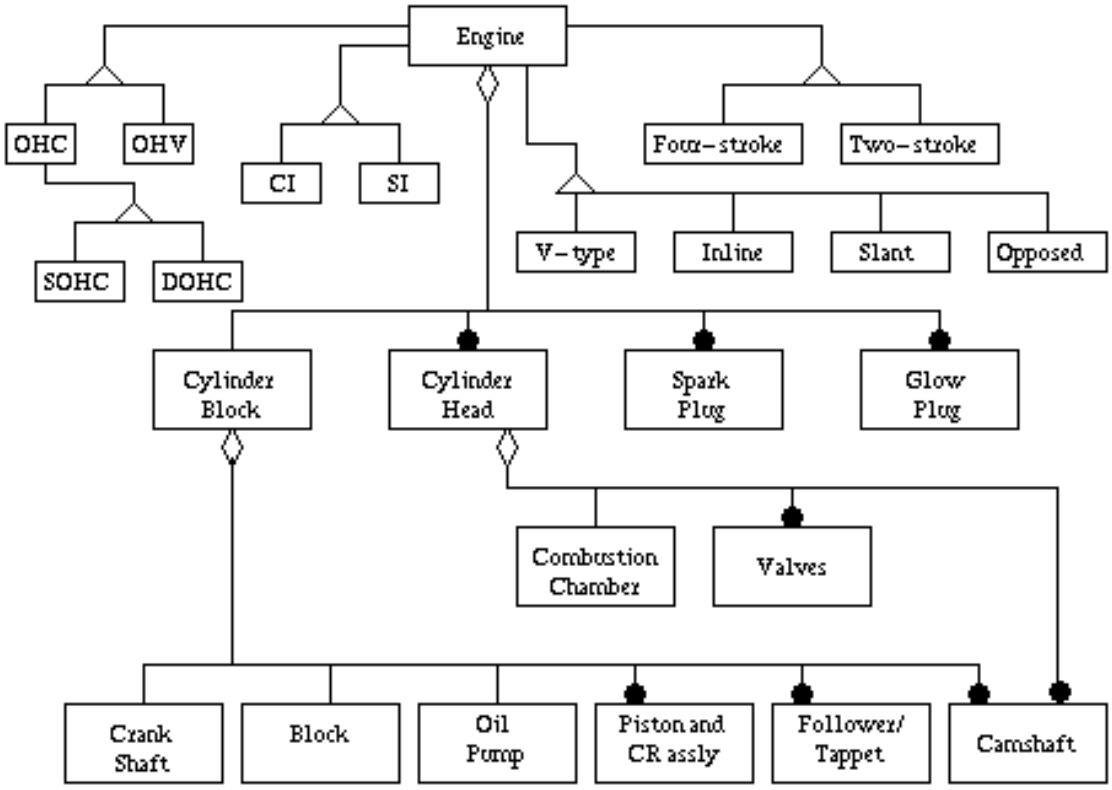

Figure 8: The final object model

The software being developed here gathers the attributes for each of the objects in the file structure and generates C++ class definitions. The class definition will consists of the following:

- Private data members - Attributes
- Public member functions - default operations such as
    - functions that map the product model into the database
    - population functions such as “set functions”
    - retrieval functions such as “get functions” for each of the attribute
    - functions that enable the parents to access attributes of their children and vice versa
    - display functions for viewing
    - some of the search functions that are embedded in database queries

4 APPLICATION - A Retrieval Engine

The retrieval engine is actually an application that is built to access data stored in the underlying GT Product model. In the software prototype being developed, the designer will have two main functional interfaces: (i) to add, edit and delete the persistent object instances in the data repository, and (ii) to perform similarity searches to enable variant design activities. The types of information retrievals that can be performed are based on:

1. A single attribute value: In this case the function call is usually a single database query that retrieves all object instances that have their attribute values equal to the designer specified attribute value.
2. A set of single valued attributes: In this case the function call is usually a single database query that retrieves all objects instances that have a set of attribute values as specified.
3. A single attribute within a range of values: In this case the function call is usually a set of database queries that use some filtering before showing up .
4. A set of attributes each in a range of values: In this case the function call retrieves all object instances that have their attribute values with in the range specified.
5. Mutually exclusive parts: In this case, the function takes a particular object as an argument instead of the attribute value(s) and retrieves information about objects that are mutually exclusive with the specified part. The function is a set of database queries that pick up the mutually exclusive relationship between objects in the model.
6. Multi-level and single level explosions of end-products: In this case the function takes as an argument an object and displays the entire product structure under it in the first instance and displays only the immediate children in the second.
7. Designs that are similar to the candidate's design intent by a specified percentage: This function retrieves objects that have their important or critical attributes with in the percentage

specified. The results are then subjected to the Analytical Hierarchical Process (AHP) to rank object instances. To perform such a query, it is important at the attribute collection stage to indicate the criticality of the attributes in each object.

5 CONCLUSIONS AND RECOMMENDATIONS

One of the efficient ways for a company to rapidly adjust in this dynamic and competitive environment is to participate in virtual company formation. Collaborating with other participating companies and sharing resources, capabilities and information serves the purpose of satisfying the market demand rapidly. The work discussed concerns the need for modeling product related data in a open architecture so as to enable participating companies to rapidly find such data to perform manufacturing activities. Although product development is the main focus here, different types of data suitable to support other manufacturing activities both locally and globally in an agile environment can be accommodated. The paper presents a systematic procedure to construct such an open product model for complex assemblies so as to enable variant design of an end-product. Most often a complex assembly consists of sub-assemblies and components in a multi-level product structure. The systematic procedure is practical as it starts from the available and existing information. Implementation specifics of the software prototype that is being developed to automate the propose procedure are also presented. The characteristics of the retrieval engine with various modes of searches are also discussed.

The ongoing work consists of building a robust software prototype that can simulate the systematic procedure that is outlined. The prototype will be user assisted especially where it is difficult to automate such as attribute collection, giving object names for the clusters when identified, adding abstract classes and editing access paths. A retrieval engine for searching similar designs based on

the designer's search intent for enabling variant design of end-product is also being developed. The use of AHP for ranking the objects for use to variant end-product design in the context of database is also under investigation. Other areas that need to be investigated are (i) an automated analytical tool to identify clustering instead of the present user assisted clustering algorithm, and (ii) an automatic identification of attribute based classifications which can be assisted by a learning algorithm after populating the database. Both these cases are being investigated.

ACKNOWLEDGEMENTS

Rakesh Nagi acknowledges the support of the National Science foundation career grant DMI-962409.

REFERENCES

Allen, D.K. (1994). Group Technology. *The Journal of Applied Manufacturing Systems* 6(2), 37-46.

Allen, D. and P. Smith (1980). Part Classification and coding. Monograph No.3. Brigham Young University, CAM Software Laboratory.

Billo, R.E. , R. Rucker, and D.L. Shunk (1987). Integration of a group technology classification and coding system with an engineering database. *Journal of Manufacturing Systems* 6(1), 37-45.

Billo, R.E., R. Rucker, and D.L. Shunk (1988). Enhancing group technology with database abstractions. *Journal of Manufacturing Systems* 7(2), 95-106.

Billo, R.E. and B. Bidanda (1995). Representing group technology classification and coding techniques with object oriented modeling principles. *IIE Transactions* 27, 542-554.

Bond, A. and R. Jain (1988). The Formal Definition and Automatic Extraction of Group Technology Codes. *Proceedings of the ASME Computers in Engineering Conference*, 537-542.

Chung, Y. and G.W. Fischer (1994). A conceptual structure and issues for an object-oriented bill of materials (BOM). *Computers Ind. Engng* 26(2), 321-339.

Ham, I., D. Marion, and J. Rubinovich (1986). Developing a group technology coding and classification scheme. *Industrial Engineering*, 18(7), 90-97.

Harhalakis, G., A. Kinsey, and I. Minis (1992). Automated group technology code generation using PDES. *Proceedings of the Third International Conference on Computer Integrated Manufacturing*, Rensselaer Institute, Troy, NY.

Hegge, H.M.H, and J.C. Wortmann (1991). Generic bill-of-material: a new product model. *International Journal of Production Economics* 23, 117-128.

Hyer, N.L. and U. Wemmerlov (1985). Group Technology oriented coding systems: Structures, applications, and implementation. *Production and Inventory Management* 26, 55-78.

Iyer, S. and R. Nagi (1994). Identification of Similar Parts in Agile Manufacturing. *Concurrent Product Design*, ASME 74, 87-96.

Kinsey, A. (1992). Automated Generation of Group Technology Codes from a PDES Product Information Model. M.S. 92-7, Master's Thesis, Systems Research Center, University of Maryland, College Park.

Krause, F.L., F. Kimura, T. Kjellberg, and S.C.Y. Lu (1993). Product Modeling. *Annals of the CIRP* 42(2), 695-705.

McAuley, J. (1972). Machine Grouping for Efficient Production. *The Production Engineer* 51(53), 53-57.

Minis, I., A. Candadai, S. Champati, J. W. Herrmann, and V. Ramachandran (1993). Information Needs in Agile Manufacturing. Systems Research Center, University of Maryland, College Park.

OIR and Multi-M (1986). Code Book and Conventions. Organization for Industrial Research, Waltham, MA.

Opitz, A. (1970). A Classification System to Describe Workpieces. Pergamon Press.

Reodecha, M. (1985). A Classification and Coding System for CAD/CAM Applications in the Electronic Industry. Ph.D Dissertation, North Carolina State University, Raleigh, NC.

Rumbaugh, J., M. Blaha, W. Premerlani, F. Eddy, and W. Lorensen (1991). Object-Oriented Modeling and Design. Prentice Hall.

Seifoddini, H.M, and P.M. Wolfe (1986). Application of similarity coefficient method in GT. *IIE Transactions* 18, 271-277.

Shah, J. and A. Bhatnagar (1989). Group Technology Classification from Feature-Based Geometric Models. *Manufacturing Review*, 2(3), 204-213.

Teicholz, E. and J. Orr (1987). Computer Integrated Manufacturing. McGraw-Hill.

Usher, J.M. (1993). An object-oriented approach to product modeling for manufacturing systems. *Computers Ind. Engng* 25(4), 557-560.