This document contains the draft version of the following paper:

M.V. Karnik, S.K. Gupta, D.K. Anand, F.J. Valenta, and I.A. Wexler. Design Navigator system: A case study in improving product development through improved information management. *ASME Computers and Information in Engineering Conference*, Long Beach, CA, September 2005.

Readers are encouraged to get the official version from the conference proceedings or by contacting Dr. S.K. Gupta (skgupta@umd.edu).

DESIGN NAVIGATOR SYSTEM: A CASE STUDY IN IMPROVING PRODUCT DEVELOPMENT THROUGH IMPROVED INFORMATION MANAGEMENT

M.V. Karnik

Iktara and Associates, LLC
Bethesda, MD
mkarnik@iktara.com

S.K. Gupta[1]

Dept. of Mechanical Engineering
University of Maryland
College Park, MD 20742
skgupta@eng.umd.edu

D.K. Anand

Dept. of Mechanical Engineering
University of Maryland
College Park, MD 20742
dkanand@eng.umd.edu

F.J. Valenta

Naval Surface Warfare Center
Indian Head Division
Indian Head, MD 20640
frank.valenta@navy.mil

I.A. Wexler

Naval Surface Warfare Center
Indian Head Division
Indian Head, MD 20640
ian.wexler@navy.mil

ABSTRACT

This paper provides an overview of the Design Navigator system being developed for the Naval Surface Warfare Center, Indian Head. This system addresses the following three information management needs. First, it captures all the relevant information being generated during the product development process in a computer-interpretable form. This eliminates information loss from the design process. Second, it builds fully interconnected information models. Thus ensuring full connectivity between requirements and specifications to various parts and assemblies in the design. Third, it offers multiple modes of searching and retrieving design information in an intuitive and convenient manner. By supporting functionality-based queries, change-based queries, geometry-based queries, and visual navigation of the entire product database, the Design Navigator system is expected to offer maximum flexibility and power to the designers to meet their diverse information retrieval needs.

1. MOTIVATION

Constantly changing requirements, development of new materials and technologies, obsolescence of old technologies, desire to reduce cost, and a changing supplier base are the major factors that drive product improvement in defense sector. At the Naval Surface Warfare Center (NSWC) at Indian Head, which specializes in life cycle support for cartridge actuated devices (CADs) and propellant actuated devices (PADs), product improvement projects often need to be performed. CADs are small devices that contain cartridges to perform work or transmit a signal and do not contain any electronic components. CADs usually contain a mechanism to initiate the cartridge and features to control the output of the cartridge. CADs are widely used in emergency life support, aircrew escape, and weapons systems such as canopy jettison systems, ejection seats, emergency release systems, floatation devices, and fire extinguisher systems. PADs are rocket powered, propulsive devices that use controlled energy to perform work in an aircrew escape system. PADs are used in aircrew escape systems as seat-back rocket motors, capsule escape system rocket motors, and under seat rocket motors.

CADs and PADs are critical explosive components used in aircrew escape/ejection systems of military high performance aircraft. CADs and PADs contain reactive energetic materials such as propellants, explosives, and pyrotechnics, and are relatively inexpensive. They are subjected to relatively severe thermal and dynamic environments and, therefore, have limited installed service life on the aircraft. Because of this, CADs and PADs are replaced at regular intervals between 36 and 120 months depending upon the device and its application in an aircraft that may remain in operational service for 30 years.

Product development teams at NSWC face unique information management challenges. While performing a product improvement, designers often need to utilize information gained during previous projects. A change that has

[1] Corresponding Author

been successfully incorporated in one design can often be incorporated into other designs as well. On the other hand, if a proposed change ultimately is not approved, then this change is often not attempted in other designs. Hence, to efficiently and successfully perform product improvements, one must capture, store, and reuse all the relevant design information.

To make decisions, designers at NSWC often need to gather information about the available alternatives, and evaluate these alternatives using some selection criteria. The quality of a decision depends upon the quality of information used to make that decision. Delay in getting right information results in delays in decision making. The information from previous projects is often useful in future projects. However to use information from previous projects, it should be archived and should be easily accessible at the time of decision making.

Existing software systems for mechanical design only store the final design, and all intermediate information generated during the design process is lost. This intermediate information, however, is quite valuable in understanding the thought process through which the initial requirements are transformed into the final design. The loss of information prevents efficient utilization of previous design information in subsequent design projects. Even if some design information is archived, existing software systems do not provide efficient content-based search tools to search through the archived information.

Currently, Product Lifecycle Management (PLM) systems do not archive all the relevant design information. They are mainly used for access and version control. They store the changes in the design geometry but do not store the rationale or the process by which the changes were brought about. A significant amount of information stored using PLM systems is in the form of text documents so it is difficult to perform content-based search using current PLM tools. Hence, PLM systems do not provide capabilities of archiving and retrieving all the relevant design information. As a result, most product development engineers spend a lot of time searching for information.

This paper provides an overview of the Design Navigator system that addresses the following three information management needs for the designers at Naval Surface Warfare Center (NSWC), Indian Head.

1. Captures all information being generated during the design process in a computer-interpretable form. This capability eliminates information loss from the design process.
2. Builds fully interconnected information models. Thus ensuring full connectivity between requirements and specifications to various parts and assemblies in the design.
3. Offers four intuitive and convenient modes for searching and retrieving design information: (1) functionality-based queries; (2) geometry-based queries; (3) change-based queries; and (4) visual navigation of product database. This capability offers maximum flexibility to the NSWC designers to meet their diverse needs.

2. STATE OF THE ART

In order for the design information and knowledge to be reusable, it must be represented in a computer-interpretable form. Many different efforts have been made to model and represent design information and knowledge [Bilg97, Bens01, Neel03, Panc04, Mock04]. Some of these methods are highly specialized and some of them are very abstract. Due to space restrictions, it is not possible to include a comprehensive literature review of the overall field. We will instead briefly describe the main topics related to this paper and include references to survey papers and representative works. The following are the main categories that are relevant to this paper from information modeling point of view:

- **Design Information Flow Models**: This area typically deals with the modeling of the information being generated as the requirements are mapped into detailed description of products. Information flow models to a large extent depend upon the design methodology being used. Representative works in this area include [Shoo00, Szyk01, Fenv01].
- **Requirements Modeling**: Extensive research has also been performed on modeling of requirements. Research in this field is usually concerned with the modeling of requirements that drive the design process. Usually top-level requirements are decomposed into finer requirements and represented hierarchically. Understanding and documenting requirements is a key step towards ensuring that the design process meets the requirements. Representative works in this area include [Sunn03, Fu03, Beck03, Balm04].
- **Function Modeling**: Determination of functions and performing functional decomposition play an important role during the conceptual design. Due to hierarchical nature of the commonly used design methodologies, functions are often decomposed into sub-functions. Besides the hierarchical relationships between functions and sub-functions, different levels of importance for all functions also need to be identified. Function representation is also referred to as functional modeling. Representative works in this area include [Kirs98, Ston00, Hirt01, Bohm04].
- **Conceptual Design Modeling**: This area mainly deals with modeling of designs during the conceptual design stage. These types of representations mainly try to capture underlying components, their functions, and their behaviors in an integrated framework. In addition, interactions and relationships among different design elements are also captured. Representative works in this area include [Umed96, Varg04, Fisc04, Xu04].
- **Feature-Based Product Modeling**: During the detailed design of mechanical products, most modern design tools usually use feature-based representation [Shah95]. However, commercially available systems often do not maintain connectivity between feature-based representations and other types of representations mentioned above.
- **Rationale Modeling**: This area mainly deals with representing and archiving the information that explains how and why different decisions were made during the design process. Sometimes rationale may include information such as when, where, and who was responsible for making the decisions. Archiving design rationale is critical for better understanding of a design, capturing the design history, and reusing existing designs. Representative works in this area include [Pena95, Lee97, Hu00, Nom04].

Designers should be able to retrieve the archived information by performing content-based search. The problem

of design information retrieval has been studied in the following two contexts:

- **Case Based Reasoning**: This area mainly focuses on automating the synthesis of new solutions based on previous ones [Szyk01]. A key step in accomplishing this is retrieval of appropriate prior solutions. In the engineering design and manufacturing area, several case based design and case based reasoning systems have been developed. For example, CADET [Syca92] and its descendent projects focused on conceptual design, solving problems by the use of relationships that capture function, structure and behavior contained in the system's knowledge base. Case retrieval is typically performed using variations of graph matching, and is supported at varying degrees of abstraction. The reasoning techniques in this field are primarily symbolic and do not exploit all of the available design information. Representative works in the area of case-based reasoning include [Kolo93, Amao94, Mahe97].
- **Shape Based Retrieval**: The problem of finding similar parts has been explored in many different design and manufacturing contexts [Card03, Li04]. There are broadly two different kinds of methods. The first type of methods uses the overall object shape in identifying similar parts. The second type of methods uses shape features in identifying similar parts. Both of these types of approaches have useful attributes and depending upon a particular application, one might be better than the other.

3. OVERVIEW OF DESIGN NAVIGATOR SYSTEM

The main goal of Design Navigator system is to facilitate management of design information. This system enables designers to effectively reuse existing design information. The designers can learn from relevant prior experience such as previously conducted test results. It also serves as a learning tool for new designers enabling them to become productive quickly by providing them with adequate examples of creating and improving designs. By using such a system designers will be able to get answers to queries such as 'Were there any projects in which metal parts were replaced by ceramic parts?'

To manage information one must be able to represent, archive, and retrieve all design-related information. This framework should consist of a design repository, a reasoning procedure, and an easy to use interface. Because the design information contains data, text, geometry, and formulas, the user interface must be able to handle these different representations.

The Design Navigator system utilizes two different types of design information. The first type of information is about the design itself. The second type of information pertains to the design's evolution or its history. The design information is shown in Figure 1 and can be divided into the following six types:

- Requirements and Specifications: This is the highest level of design information. It contains the requirements and specifications that drive the decisions made during the design process. Design requirement information is usually obtained from customers and then given to the design team. This type of information captures the context in which the design is performed and is necessary whenever the design is revisited for modification or as a starting point for the design of a new product.

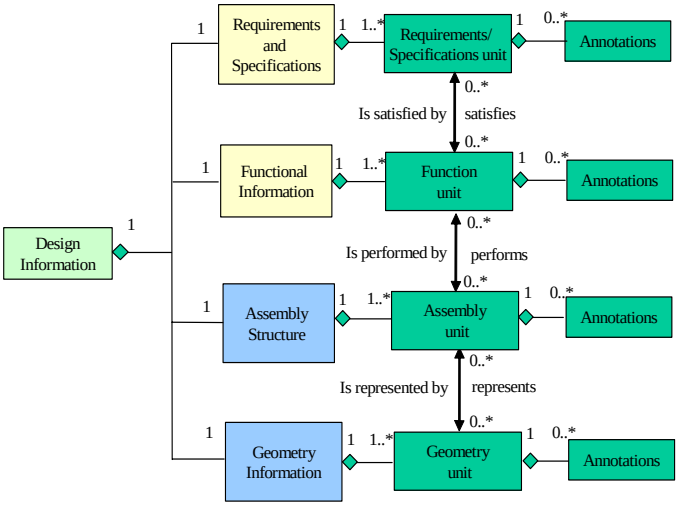

Figure 1: Design information available in the *Design Navigator* system.

- Functional Decomposition: This level provides information behind the functional decomposition. The process of design can be viewed as a mapping from the desired function to a combination of subsystems and components that results in a behavior satisfying that function. To reduce the complexity of this mapping, designers often use a process of decomposition. They decompose the desired high-level function hierarchically into sub-functions, until the functions are simple enough to be mapped to individual physical components, and then compose these components back into complete systems. The decomposition information is critical for the understanding of the original design process and, thus, provides an invaluable guide when any improvement to the original design is made.
- Assembly Structure: This level includes information such as the various assemblies and their relationship with each other. This information is obtained after the detailed design has been performed by the design team.
- Part Geometry: This level contains geometric information about parts including the final design parameters, and the final geometry of the design. This information is also obtained after the detailed design has been performed by the design team.
- Annotations: Annotations can be attached to all the levels of design information to store additional design-related information. Annotations may be in the form of web-links, audio, video, PDF documents or free-form text. The designer can attach comments, test results, supplier information or any such data in any of the formats specified above.
- Interconnections: Interconnections are used to connect different levels of information to represent the mapping between (1) Requirements-Functional Decomposition (2) Functional Decomposition-Assembly Structure, and (3) Assembly Structure-Geometry. The designer can then know the functions that satisfy a particular requirement, the part of the assembly that performs a particular function.

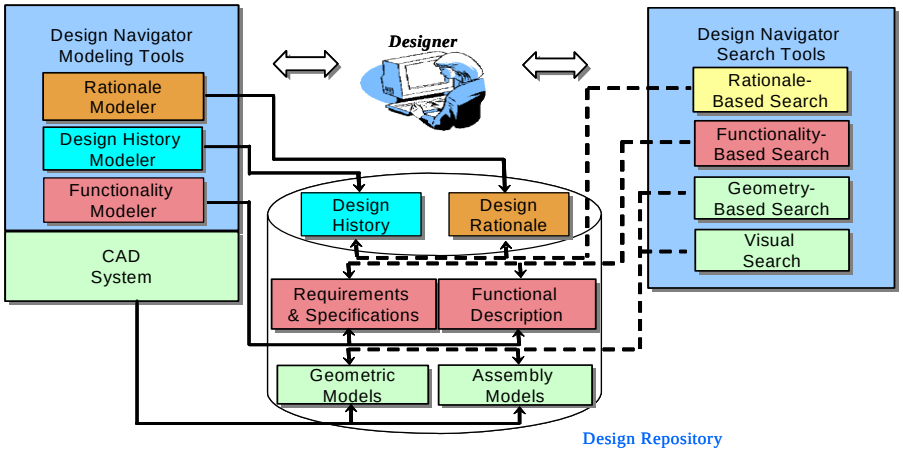

Figure 2: *Design Navigator* system architecture for information management.

Two pieces of information are required for the second type of information, which consists of the design's history:

- Administrative information: This level includes information such as designers' name, the date of initiation, and other administrative information. The designer can attach comments to it.
- Detailed information about the improvement: This information includes the nature of improvement in the design and the reason for carrying out the improvement. Two taxonomies are used to represent this information.

Figure 2 shows our architecture for the information management system. The information management system consists of Archival Tools and Search Tools. The Archival Tools include Design History Modeler, Functionality Modeler, Rationale Modeler, and the Search Tools include Rationale-Based Search, Functionality-Based Search, Geometry-Based Search and Visual Search.

- Archival Tools: The Archival Tools provide capabilities for archiving design information. There are three archival tools that archive information not captured by a typical CAD system. These tools include Design History Modeler, Functionality Modeler, and Rationale Modeler.
- Search Tools: The Search Tools provide searching capabilities for all the design information. Thus, the designer has access to information from previous designs that can be used in designing new products as well as performing Product Improvement Projects (PIPs). There are four search modules namely Functionality-based search, Rationale-based search, Geometry-based search and Visual search.

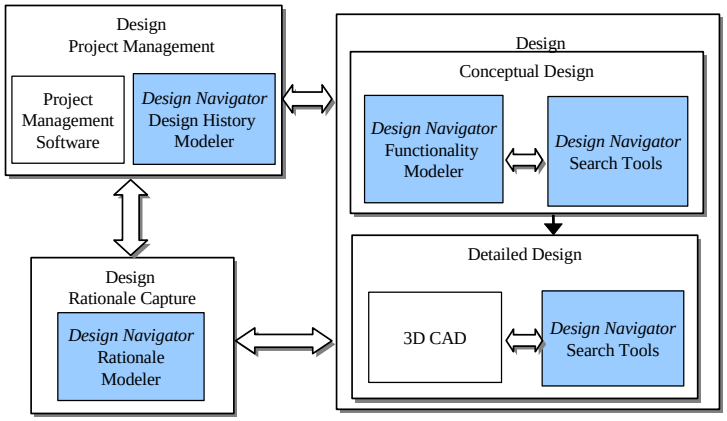

Figure 3: Use of *Design Navigator* in design process.

Figure 3 shows how the Design Navigator system fits into the design process. The Design History Modeler, Functionality Modeler, Rationale Modeler, and Search Tools can be used to create new designs as well as perform product improvement.

4. ARCHIVAL TOOLS

4.1 Design History Modeler

The Design History Modeler module is used to capture the administrative information associated with the design and PIPs performed on the design. Initially, the designer creates a new design project and enters all the administrative information, which includes the project name, a brief description, the type of design, and the designers. In this new project, the designer creates a new design. Initially, this design is empty. As the designer goes through the design process he will add additional information to this design. The user can launch this design in a viewer and view all the information.

Suppose the design now needs to be modified. This can be done by creating an Engineering Change Proposals (ECP). An ECP is a document that describes the nature of the proposed change (change action) and the reason for proposing the change (change rationale).

The change action and change rationale can be attached using a customizable dialog. This dialog is based on a standard taxonomy that is stored in an XML file. A system administrator can change the contents of the dialog by changing the XML file. The code does not need to be changed to alter the taxonomy. Thus, the system administrator can easily add new elements or delete elements from the change action and change rationale taxonomies. Thus, we provide an extensible taxonomy that does not require the system administrator to change the code.

The process for making a change in Design History Modeler (DHM) is explained below.

1. The designer can select the design and propose a change. This will launch a dialog that the designer needs to

complete. The dialog contains one section for entering administrative information such as date of proposed change, the designers etc. and some comments, a second section for entering the nature of proposed change (change action) and a third section for entering the reasons for proposed change (change rationale). After entering the information, the ECP will be represented as a node in the DHM module connected to the old version of the design.

2. The proposal is then screened to check if it should undergo further evaluation. At the end of the screening the proposal may be approved for further evaluation or rejected. The reasons for approving/rejecting the proposal are included in a file that is attached to the 'screening' node.
3. If the proposal is accepted during the screening phase then it is sent for an engineering review of the suggested changes. This generally involves performing physical tests and/or numerical simulations. The test results are attached to the engineering review node.
4. At the end of engineering review, there are two possibilities: (1) the proposed change is accepted (2) the proposed change is rejected. The reasons for accepting/ rejecting the proposal are attached to the 'engineering review' node.
5. The proposal is then sent to the board for a final decision. The decision is based on numerous factors. Again, there are two possibilities: (1) the proposed change is accepted (2) the proposed change is rejected. The reasons for accepting/ rejecting the proposal are attached to the 'board decision' node.
6. If the proposed change is accepted then a new version of the design is created. This new version will be a new design node.

This process of capturing the change rationale is consistent with the current practice at NSWC, Indian Head. We have studied their Engineering Change Proposal documents and have come up with this change management process. However, the change management process varies from organization to organization. The Design History Modeler is flexible to incorporate any changes in the change management process without changing the code. This is because the DHM system reads the change management process at run-time from an XML file and provides the appropriate options. For example suppose the 'screening' process may not be taking place in some organization, then that node could easily be dropped from the system.

The design is actually modified using Functionality Modeler, CAD software, and Rationale Modeler. The Functionality Modeler is used to change the functional information of the design, if required. CAD software is used to change the geometry information of the design. Once these changes have been made, the Rationale Modeler can be used to import this information, and create a modified version of the design. The administrative information, change action and change rationale information are transferred from the ECP to the design automatically. The new design being represented as a node is connected to the previous design node through the ECP. If the proposed change is rejected then the following information is archived: the reason for rejecting the proposed change, any test results or other files supporting the reason for rejecting the change. Thus, the system archives even the failed change proposals. The ECP information can be obtained by clicking on the ECP nodes. The design information can be viewed in a viewer by clicking on the design nodes.

4.2 Functionality Modeler

Functionality Modeler is used to model conceptual design. Conceptual design information consists of requirements and functional decomposition. The requirements and specifications drive the decisions made during the design process. Design requirement information is usually obtained from customers and then given to the design team. Functional decomposition is hierarchical decomposition of the desired high-level function into sub-functions, until the functions are simple enough to be mapped to individual physical components, and then compose these components back into complete systems.

The Functionality Modeler is used to model the requirements, specifications and functional decomposition of a conceptual design. It consists of one requirements window and one functional decomposition window. Both requirements and functional decomposition are represented as trees. In the initial state, the requirements window and the functional decomposition window consist of one node each (top-level node). The user can add or delete nodes in both the windows by selecting 'add node' or 'delete node' on right click. The user can thus build the requirements as well as functional decomposition tree. A requirement/function consists of one action item (function) and one or more entities on which the action is performed. The entities could be any of the following types: energy, material, wave, space etc. The requirements and function are read from a standard taxonomy.

The Functionality Modeler archives different types of requirements. These include functional requirements, manufacturing requirements, testing requirements, handling requirements and any miscellaneous requirements.

A dialog will pop up in which the user can select a requirement or function from a pre-defined taxonomy. The entity/entities on which the function acts can be defined by the user. A standard taxonomy is typically followed by a particular company to represent the requirements and functions of its product(s). The user can also add custom requirements and functions. There are two types of connections to connect requirements and functions. One type of connection connects two nodes within the same window. For example a connection between two requirements nodes. The other type of connection connects two nodes from different windows. This helps in mapping a particular requirement to a particular function in the functional decomposition.

The user can add a list of attributes to each node both in the requirements as well as functional decomposition window. The attributes represent the specifications for that requirement or function that need to be satisfied. A maximum and minimum value for the attribute can be set. The system checks if the given value lies between the minimum and maximum. The user can add following two types of attributes.

- Standard: These attributes are commonly used by the company for a particular variety of products and are included in a standard taxonomy. The dialog box for adding attributes contains a list of these standard attributes from which user can select one. Before the dialog box is launched, the system parses an XML file containing the attributes. The system administrator can add, remove or

modify attributes in the XML file. The minimum and maximum values for the attribute are also set automatically.

- Custom: Sometimes the user may want to attach non-standard attributes to the node. The system provides a dialog-box to add non-standard attributes.

4.3 Rationale Modeler

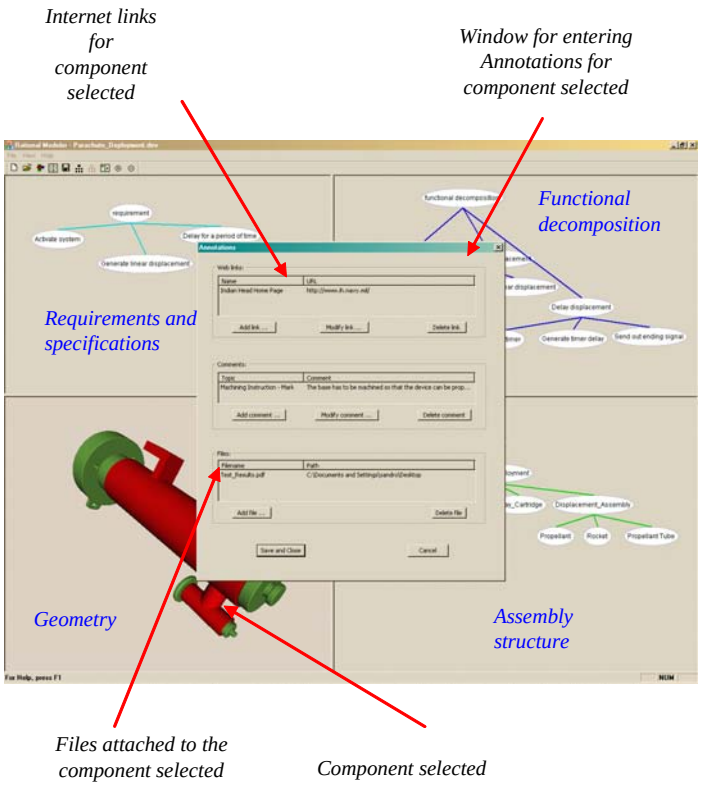

Figure 4: Standard dialog box used to attach information in requirement view, concept view, assembly view and/or geometry view.

Rationale Modeler assists designers in their understanding of how design parameters relate to requirements in existing designs. The ability to do this greatly improves the focus of the designers towards satisfying the device's important functional requirements and provides the framework by which one can record the reasons why the design is the way it is and, when applicable, why the design was modified.

Design Requirement:

1. Functional - Design a parachute displacement system that generates a linear displacement of 0.5 m 0.5 ± 0.01 s after requested.
2. Manufacturing - Conform to requirements in AS 3885, conform to drawing 946AS100 and condition prior to production.
3. Testing - Test in accordance with MIL-A-85097
4. Handling - Place in hermetically sealed can. Mark with following phrase "Explosive Power Device, Class C".
5. Miscellaneous - Prevent propellant exposure to environment. Follow Specifications in MIL-S-10464

Design Specification:

Delay time - 0.5 ± 0.01 s
Linear motion - velocity 20 m/s; displacement 0.5 m
Geometric compatibility with seat and propellant connector
Operating conditions - –55 to 62 °C and 500 to 1000 Pa
Weight – 4.5 kgs
Service life - 7 years

Figure 5: Design requirements and specifications.

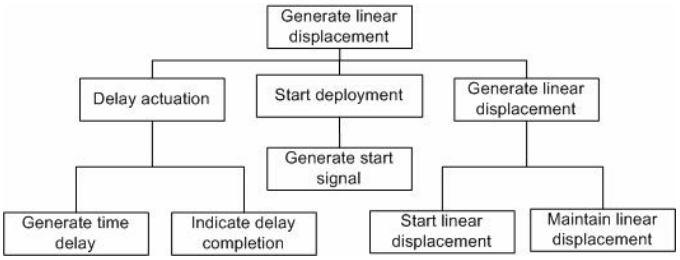

Figure 6: Functional decomposition of parachute displacement system.

Figure 4 shows a design being represented in the Rationale Modeler. The example shown in this figure is for a Parachute Displacement Device (PDD), which generates a linear displacement of 0.5 m in 0.5 ± 0.01 s after requested.

Figure 5 shows the design requirements and specifications of the PDD. After the design requirements and specifications are recorded, designers usually generate a structure representing a functional decomposition of the device. These functions must satisfy the product's requirements. Figure 6 shows the functional decomposition of the PDD. The requirements, specifications, and functional decomposition are recorded using the Functionality Modeler. After completing the functional decomposition, the concept is converted into a detailed design and the various functional subsystems are assembled to form the final product. These latter steps are performed using CAD software. Figure 7 shows the assembly structure for the PDD.

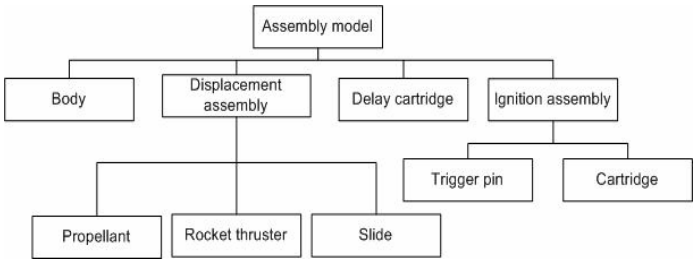

Figure 7: Assembly structure of parachute displacement system.

The four views shown in Figure 4 are linked such that when the user clicks on an item in one view, the corresponding items in the other three views are automatically highlighted. For example, if we click on one requirement item in the requirement view, the corresponding function(s) that satisfies this requirement are highlighted, as are the corresponding assembly components in the assembly view and corresponding parts in the geometry view. These four views represent visually the basic information about a design.

In addition to the information represented in these four major views, designers are allowed to record and subsequently retrieve supplementary information on any aspect of the design. For example, as shown in Figure 4, if we click on the delay cartridge in the assembly view of the PDD, a template appears. By using this template, one can record, and subsequently

retrieve information such as the evaluation results of the different delay cartridges considered, the online vendors who sell it, etc.

5. SEARCH TOOLS

The Search Tools have been designed to search for information stored using the modeling systems described above. Sections 5.1 through 5.5 describe these search tools.

5.1 Visual Search

Visual search tool is capable of loading parts that fit particular size/complexity criteria into the system and then displaying all the parts in a single scene. The user needs to specify only the high level directory. The system searches all the sub-directories recursively, loads the relevant files, and displays them in the scene. Multiple navigation and sorting utilities have been implemented in visual search to aid the user in visually locating the part that he/she is looking for. Some basic size and complexity based filters have also been implemented to reduce the size of the database.

Visual Search tool uses a scene graph to display all the part files in a single scene. A scene graph is a directed acyclic graph consisting of heterogeneous nodes that represent the elements of a scene. The scene graph helps in representing the elements of the scene as objects and promotes object oriented programming. The typical nodes of a scene graph include geometry, material/property, camera, light and some other nodes. The scene graph in Visual Search is used to perform the culling, level-of-detail and rendering operations.

Visual Search provides auto panning so that the user can navigate through the parts without any interaction from the user. There are two different ways of organizing the parts on the screen namely planar and cylindrical. All parts are rescaled to the same size after loading. This prevents overlapping when displaying them and also ensures that large differences in size do not affect the visualization. A sorting function has been implemented to sort the parts based on size i.e. radius of the bounding sphere. For display purposes, all parts have the same size. However, to distinguish between the sizes, larger parts are colored in red while smaller parts are colored in white. A filtering utility has also been implemented to filter out parts based on size and complexity. The size criteria help in filtering the parts based on volume while the complexity criteria help in filtering the parts based on number of faces.

5.2 Geometry-Based Search

The geometry-based search tool locates existing parts similar to the new part based on some geometric attributes. It creates signatures for each of the parts in the database and stores the signatures along with the solid model of the part. A signature is a list of geometric attributes that describe the part and depends on the application. These pre-computed signatures reduce the time required for comparison and, thus, improve the speed of comparison. The search tool then uses the signatures to compare the signature of the query part with each of the signatures of the database parts to determine if the parts are similar.

The geometry-based search tool uses signatures based on either the overall shape or shape features for comparison depending upon the application. The overall shape based method uses four signatures to identify existing parts similar to the query part. The signatures are applied sequentially to improve the efficiency and accuracy of the comparison. The four signatures used to compare the query part with database part to assess similarity are part volume and surface area, basic shape statistic, gross shape complexity, and detailed shape complexity and are explained in [Karn05]. The shape features based method is explained in [Card04].

5.3 Change Rationale-Based Search

Rationale-based search tool is used to search the rationale behind any change in the design. There are two scenarios in which a designer is looking for information. In one case the designer knows the specific detail of the previous project that he/she is looking for. e.g. all the projects in which molybdenum was replaced. However sometimes he/she may just have a vague idea or hint of the information that he is looking e.g. approximate date, designer's name etc. Our system can support both types of designers needs.

5.3.1 Search Definition

The rationale-based search needs to be performed on the following information: change action, change rationale, administrative details, and comments. To define the search, the user needs to specify at least one criterion from the four mentioned above. We have developed search definition methods based on studying designers' typical needs in carrying out these searches.

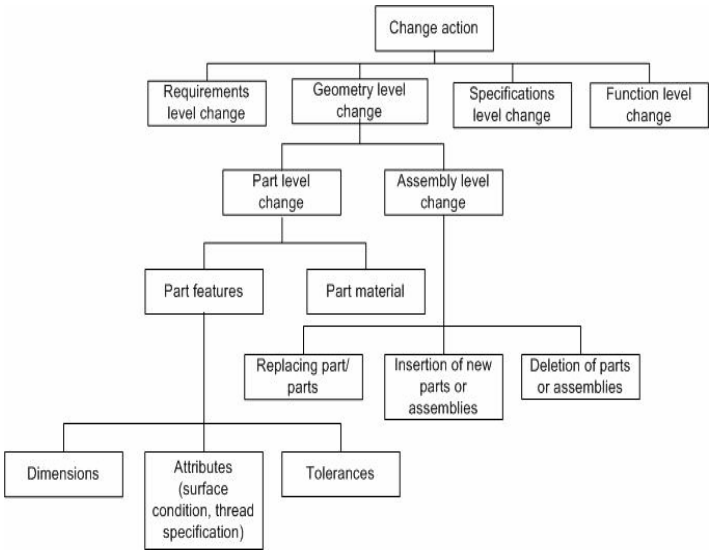

Figure 8: Taxonomy for change actions.

- Change action and change rationale search: The sub-criteria for change action and change rationale are in the form of a tree as shown in Figure 8 and Figure 9, respectively. The user can specify the search criteria at a particular level. The system will search for that criteria and the entire sub-tree. Suppose the user wants to locate all the designs in which a part level change was performed. The user only needs to check the box next to part level change. The entire sub-tree is selected and searched. However, if the user wants to locate all part level changes except the ones in which tolerance was changed, then he can select part level change and deselect tolerance change. The change rationale search is specified in a similar manner.

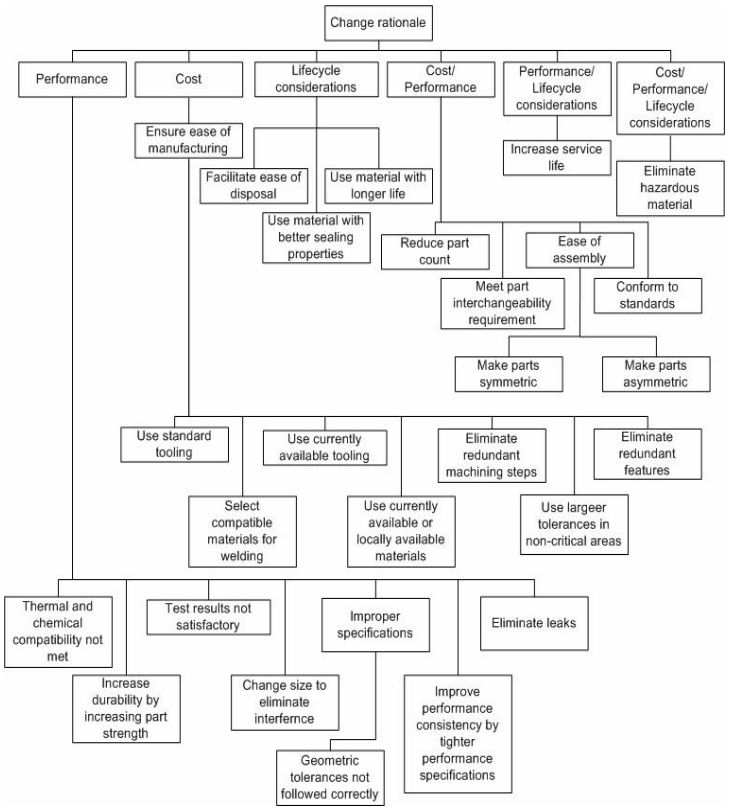

Figure 9: Taxonomy for change rationale for CAD/PAD devices.

- Comments search: The search for comments is specified as free-form text. The user can specify keywords using AND/OR operators.
- Administrative information search: The administrative information includes two inputs. One input is the target date, which is used to specify the approximate date when the change proposal was made. The second input is free-form text such as designer's name, project name, etc. using AND/OR operators.

For each of the criteria and sub-criteria, the user can specify one of the three preference levels (low, medium, and high). The preference level is used to weigh the importance of a particular criterion. Using the preference level, the user can give more importance to particular criteria and reduce the importance of other criteria. As seen in the next sub-section, the preference level influences the scoring function. For example, in a certain situation a user may need to look for change proposals associated with part material changes. In this situation a user may decide to give a high preference to the change action and low preference to the comment-based search that can be used to supply extra information.

The user can also mark some criteria as 'required'; that is, if those criteria do not match, then the system should not return any result. The system does an initial filtering of the designs in the repository based on the criteria that are required. This helps to reduce the number of search results and the user can narrow down the results very quickly. Also, it reduces the time required for searching. If a user is only looking for change proposals in the cost reduction area, then the change rationale should be the required criteria.

5.3.2 Match Distance Used in Search

The string matching algorithm is a combination of two types of string matching algorithms; namely, equivalence method [Hall80] and similarity [Hall80] method. The equivalence methods compare two strings and return true or false depending upon whether the strings match or not. The similarity methods compare a given string to a set of strings and rank those strings in order of similarity. The system uses a combination of equivalence and similarity methods to perform the rationale search. The overall distance function consists of four sub-functions. The similarity method is based on the Levenshtein distance [Hirs97], which is a measure of similarity between two strings. Suppose *s* is the source string and *t* is the target string, then the Levenshtein distance is the number of deletions, insertions, or substitutions required to transform *s* into *t*. This paper uses a modified Levenshtein distance. Suppose n_{avg} is the average length of a word in the vocabulary, then the maximum Levenshtein distance between two words of average length is $2n_{avg}$. The modified Levenshtein distance has value between 0 and 10 such that if two words are the same then the value is 0 and if two words are very dissimilar the value is 10. The modified Levenshtein distance is defined as:

If $LD(s, t) \le 2n_{avg}$, then

$$LD_{mod}(s,t) = \left(\frac{LD(s,t)}{2 \times n_{avg}} \right) \times 10$$

If $LD(s, t) > 2n_{avg}$, then

$$LD_{mod}(s, t) = 10$$

where $LD(s, t)$ is the Levenshtein distance and $LD_{mod}(s, t)$ is the modified Levenshtein distance. If $LD(s, t)$ exceeds $2n_{avg}$, then the words *s* and *t* are very dissimilar and hence $LD(s, t)$ is set to $2n_{avg}$. Then, $LD_{mod}(s, t)$ is 10.

Change action and rationale based distance: The change action and change rationale search use the equivalence method to compare the strings. The system searches the archived change action files to locate those designs that underwent a particular change action. The change action taxonomy is shown in Figure 8. The taxonomy is in the form of a tree. To simplify the search, a tag is stored along with each change action item. The tag is defined based on the depth of the change action item in the tree. For example all the high level nodes have tags CA1, CA2, etc. The nodes immediately under CA1 have tags CA11, CA12, etc. While performing the search, the system searches for all the tags corresponding to the items that have been selected. The tags are searched for an exact match using Backward Oracle Matching [Alla01] and the system returns either true or false. If multiple items are selected, then the distance is computed based on the number of tag matches. The distance function used to rank the change action search is based on the number of matches of search criteria and their preferences that have been defined.

Let *n* be the number of criteria that have been specified. Suppose the *k*th search result matches *m* criteria; that is, *m* true values and *n*-*m* false values. Then distance $d_a(k)$ for *k*th search result for change action based search can be calculated as:

$$d_a(k) = \frac{1}{10^{n-1}} \prod_{i=1}^{n} c_i$$

where $c_i = 0$ if ith criterion matches (returns true) and $c_i = 10$ if ith criterion does not match (returns false).

The nature of this function is such that if any tag matches, then it returns 0. We have designed this distance function based on designers need. Typically, a change proposal only has one change action and designers are interested in viewing all change proposals that meet at least one of their specified change actions.

Similar methodology is used for change rationale taxonomy. The distance for change rationale search can be calculated as follows:

$$d_r(k) = \frac{1}{10^{n-1}} \prod_{i=1}^{n} c_i$$

Comments: The comments field is searched using Backward Oracle Matching [Alla01]. Each word of the input search string is matched with the closest possible word in the comments fields of archived design rationale. The distance between the input word and matched word is computed using the modified Levenshtein distance and is represented as $LD_{mod}(s_i, t)$, where s_i is the ith word in the input string and t is the closest match using the modified Levenshtein distance.

In general, a complex expression can be formed by combining strings using AND/OR operators. An expression is allowed to have an arbitrary level of nesting of operators. More formally, we define an arbitrary expression in the following manner. An expression is one of the following:

- A string
- A set of expressions combined with AND operator
- A set of expressions combined with OR operator

Input expression:
((Steel OR Brass) AND (Compression OR Torsion) AND (Spring))

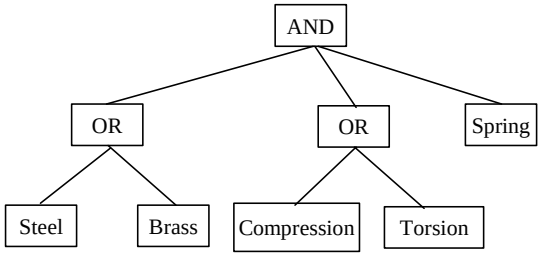

Figure 10: An example of a tree associated with an expression.

Because of the above recursive definition of the expression, it can be modeled as a tree. The leaf nodes in the expression tree are stings. The internal nodes in the tree are either and AND or OR nodes. An example of such a tree is shown in Figure 10. Distance for various nodes in the tree is computed as following.

- For a leaf node the distance is simply computed as $LD_{mod}(s, t)$ where s is the string at leaf node and t is the closest matching string in the comments field of archived design rationale.
- For an AND node the distance is computed as $\left(\frac{1}{n}\sum_{i=1}^{n} d_i\right)$.

  Here we assume that the tree has n children nodes and the distance for ith child node is d_i. So the distance for an AND node is simply an average distance for its children.
- For OR nodes the distance is computed as the minimum over all its children.

The distance for a complex expression is determined by evaluating its expression tree in a bottom up manner. The distance associated with the root node is the distance for the expression.

Administrative Information: The administrative information section has two fields; namely, target date and free-form text field. The target date field lets the user input an approximate date of the change proposal he/she is looking for. The system then computes the distance value depending upon the difference in the actual change proposal date (m_p, y_p) found and target date (m_t, y_t). This distance can be calculated as follows.

$$d_{i,1}(k) = \left(\frac{\left|12(y_t - y_p) + (m_t - m_p)\right|}{60} \right) \times 10$$

where m_p is the month of the proposal date, y_p is the year of the proposal date, m_t is the month of the target date, and y_t is the year of the target date. If the difference in the number of months exceeds 60 months (5 years), then the distance value is set to 10.

The free-form text field is searched using Backward Oracle Matching [Alla01] and the ranking is performed and the distance is similar to the comments field. The overall distance for the kth search result is given as

$$d(k) = \frac{w_a p_a d_a(k) + w_r p_r d_r(k) + w_c p_c d_c(k) + w_{i,1} p_i d_{i,1}(k) + w_{i,2} p_i d_{i,2}(k)}{w_a p_a + w_r p_r + w_c p_c + w_{i,1} p_i + w_{i,2} p_i}$$

where w_a and p_a are the weight and preference associated with change action, w_r and p_r are the weight and preference associated with change rationale, w_c and p_c are the weight and preference associated with comments, and $w_{i,1}$, $w_{i,2}$, $p_{i,1}$ and $p_{i,2}$ are the weights and preferences associated with administrative information.

This overall distance is used to rank the search result. The smaller the value of the distance, the higher is the rank of the result. The overall distance value varies between 0 and 10.

5.4 Functionality-Based Search

Functionality-Based Search tool is used to locate conceptual designs that contain particular requirement or function. The designer may want to search for a particular function or particular requirement. The designer may want to include attributes in the search definition. Our system allows designer to search for requirements and functions that are constrained by attributes.

Functionality-based search definitions and match distances are very similar to the search definitions and distances described in Section 5.3.1. Hence for the sake of brevity, they are not repeated here.

5.5 Integration between Different Search Tools

All the search tools have been integrated into one single framework. The hierarchy of information associated with design is as follows. The design evolution information is at the highest level and contains design evolution information. Thus there may be multiple versions of design within each design project depending upon the number of PIPs that have been performed. Each version of design contains the following information: requirements, specification, functional hierarchy, assembly models and geometric models. The assembly and geometry models are created using CAD system and are definitely available. The other information may or may mot be available. Based on this hierarchy following combinations of search can be performed using the Integrated Search module:

- Rationale-Based-Search followed by Functionality-Based Search followed by Geometry and Visual Search.
- Rationale-Based-Search followed by Geometry and Visual Search.
- Functionality-Based Search followed by Geometry and Visual Search.
- Geometry and Visual Search.

This integration between different search modules provides a powerful tool by means of which the designer can locate the information that he/she is looking for during the design process.

6. IMPLEMENTATION

The Design History Modeler and Functionality Modeler have been implemented using C++ and MFC. The node structure (change management process structure) has been defined in a XML file. The Design History Modeler reads this node structure at run-time and provides appropriate options during the change management process. Similarly, the requirements and function taxonomies in the Functionality Modeler have been defined in XML files. The Functionality Modeler reads the taxonomies at run-time.

The Rationale Modeler has been implemented using C++, MFC, and OpenGL. During the modeling process, it inherits the functional information from Functionality Modeler and the assembly and geometry information from a CAD system. Currently, we have interfaced the Rationale Modeler with Pro/Engineer. The data from Pro/Engineer is converted into STL file format using InterOp translators provided by Spatial Technologies. The assembly structure is maintained in an assembly file.

The Integrated Search system has been built using C++ and MFC. The functionality and rationale-based search use requirements, function, change action and change rationale taxonomies for providing the search options. These taxonomies are read from an XML file. The geometry-based search has been implemented based on the algorithms described above. To locate a set of parts that contains the query part, the designer must supply the part geometry of the query part and the directory of the database to search. The output of the system is the models of all the parts that are similar to the query part.

Visual Search is written in C++ under Microsoft Windows using the MFC-library for drawing the windows. For displaying the parts the OpenSceneGraph-library is used. This is a portable, high-level graphic library on the top of OpenGL, which was developed for high performance applications like simulators, virtual reality, games and scientific visualizations.

All the systems described above use MSXML parser for parsing the XML file.

7. CASE STUDIES

The following two case studies illustrate how the use of the Design Navigator system is expected to improve the product development process at NSWC.

7.1 Case Study 1

Scenario without the use of Design Navigator: Each CAD/PAD device is an assembly of parts. A PIP was initiated to improve the design of an emergency egress initiator. After studying the existing design, the designers proposed to use a different spring in this assembly. They then checked the existing documentation for this device. Surprisingly, they found out that the proposed spring had been considered and rejected during the process of generating the original device. However, from the existing documents, they were unable to determine what the reason behind that decision was. Based on their experiences and knowledge, they thought this new spring would perform better than the currently used one. In order to prove this, they generated a new design that used the new spring, and then performed several tests. Finally, it was determined that this new spring would cause a delay when the handle was pushed. This delay could have disastrous consequences when the emergency egress initiator was activated. Thus, to arrive at this conclusion, the designers had to repeat tests that were previously performed.

Scenario with the use of Design Navigator: Suppose the test data was documented properly during the initial design, using the Rationale Modeler. A search for a PIP was performed using the following criteria:

1. A change action of ‘Replacing one or more than one parts’.
2. A change rationale of ‘Improve performance’.
3. Comments field with ‘Spring’ keyword.
4. Administrative information showing only ‘Rejected proposals’

The search resulted in 4 PIPs for which the distance function is more than 9. The top-match was a PIP in which a similar spring had failed the test. Thus, the designers would have been able to locate the PIP and would not have had to rediscover the reasons for its rejection.

7.2 Case Study 2

Scenario without the use of Design Navigator: A PIP was initiated to reduce the manufacturing cost of bracket of parachute deployment device and the task had been assigned to a new designer. The designer did not know where to begin. He/She examined several generations of designs and consulted senior designers before proceeding with the task. Each design had to be studied to determine their advantages and disadvantages in order to find ways to improve the existing designs. This proved to be a very time consuming and frustrating task. It was also a waste of senior designers’ time.

Scenario with the use of Design Navigator: Suppose the design history and rationale had been properly recorded and archived using Design History Management and Rationale Modeler module. The designer would then be able to search for all those designs in which manufacturing cost was reduced. To

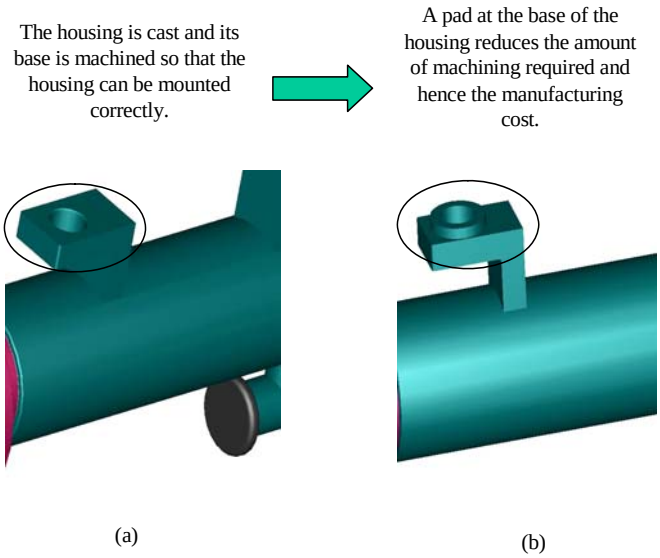

Figure 11: An example of cost reduction
(a) original design of housing, (b) modified design of housing.

test this, a search for PIP was performed using the following criteria:

1. A change action of 'Modifying part(s)'.
2. A change rationale of 'Reducing Manufacturing Cost'.
3. An empty comments field.
4. Administrative information showing PIPs performed around May 1999.

The search resulted in 2 PIPs for which distance function is more than 9. One PIP is shown in Figure 11. In this design, the cost of manufacturing has been reduced by providing a pad on the flange that is used to mount the device. Now, only the pad needs to be machined accurately for mounting and not the entire base of the flange.

The case studies for Visual and Geometry Search have been presented in [Karn05].

8 CONCLUSIONS

The Design Navigator system addresses the following three information management needs. First, it captures all the relevant information being generated during the product development process in a computer-interpretable form. This eliminates information loss from the design process. Second, it builds fully interconnected information models. Thus ensuring full connectivity between requirements and specifications to various parts and assemblies in the design. Third, it offers multiple modes of searching and retrieving design information in an intuitive and convenient manner.

Design Navigator System has the following functional capabilities:

- Can be used to ensure that the products being designed can be generated by minor modification of existing products. This will help reduce the total number of parts for which an inventory is maintained and, hence, reduce life cycle costs.
- Helps perform quick cost estimation by finding similar designs and examining their costs. This capability will help designers make cost and performance trade-off decisions during the design stage, hence creating better and more cost-effective products.
- Can be used to identify an initial list of contractors that can make a new or improved design by finding previous designs that are similar in nature and locating the contractors for those designs. This capability will reduce procurement costs and lead times.
- Records information on product improvement projects and can be used to guide new design teams in learning about the changes performed and the consequences of those changes. It gives ideas for product improvement based on previous examples. This will reduce costs and design time associated with product improvement projects.
- Allow new designers to learn how requirements, specifications, functional decomposition, assembly structures, and detailed part features relate to each other. This can help them to understand how changing one aspect of a design is likely to affect other aspects. This will help in the training of new designers and increase a designer's awareness of the affects that changes have on other components of an assembly.
- Helps in finding previously designed products with appropriate characteristics. Designers can reuse these models directly or after making minor changes to them. This capability will significantly reduce effort required in modeling complex parts and in creating new parts.

Even though the design Navigator system is being developed to meet the specific information management needs for the designers at NSWC, we believe that the underlying concepts and system has a much wider applicability and can be easily tailored to needs of other organizations.

ACKNOWLEDGMENTS

This research is supported in part by the Center for Energetic Concepts Development at the University of Maryland, Naval Surface Warfare Center at Indian Head, and Iktara and Associates.

REFERENCES

[Alla01] C. Allauzen, M. Crochemore, and M. Raffinot. Efficient experimental string matching by weak factor recognition. In Proceedings of 12th Annual Symposium on Combinatorial Pattern Matching, pages 51 – 72, 2001.

[Amao94] A. Aamodt, and E. Plazas. Case-based reasoning: foundational issues, methodological variations, and system approaches. AI Communications, 1994:7(1) 39- 52.

[Balm04] L. Balmelli and A. Moore. Requirements Modeling for System Engineering Using SYSML, The Systems Modeling Language. In CIE Conference, Paper CIE-57751, Salt Lake City, UT, September 2004.

[Beck03] B. Becker and N. Wang. ERMM: An Engineering Requirements Management Method. In CIE Conference. Paper CIE-48238, Chicago, IL, September 2003.

[Bens01] M. Benson and J. Terpenny. A Survey of Methods and Approaches to Knowledge Management in the Product Development Environment. In CIE, Paper CIE-21288, Pittsburgh, PA, September 2001.

[Bilg97] T. Bilgic and D. Rock. Product data management systems: state-of-the-art and the future. In DETC and CIE Conference, September 1997, Sacramento, CA.

[Bohm04] M. R. Bohm and R. B. Stone. Representing Functionality to Support Reuse: Conceptual and Supporting Functions. In CIE Conference, Paper CIE-57693, Salt Lake City, UT, September 2004.

[Card03] A. Cardone, S.K. Gupta, and M.V. Karnik. A survey of shape similarity assessment algorithms for product design & manufacturing applications. Journal of Computing & Information Science in Engineering, 3(2):109-118, June 2003.

[Card04] A. Cardone, S.K. Gupta, and M.V. Karnik. Identifying similar parts for assisting cost estimation of prismatic machined parts. In ASME Design for Manufacturing Conference, Salt Lake City, Utah, September 2004.

[Fenv01] S.J. Fenves. A core product model for representing design information. Technical Report, Number NISTIR6736, National Institute of Standards and Technology, Gaithersburg, MD, USA, 2001.

[Fisc04] X. Fischer, C. Merlo, J. Legardeur, L. Zimmer, and A. Anglada. Knowledge Management and Support Environment in Early Phases of Design Process. In CIE Conference, Paper CIE-57791, Salt Lake City, UT, September-October 2004.

[Fu03] M. W. Fu and W. F. Lu. Modeling and Management of Design Requirements in Product Deveopment Life Cycle. In CIE Conference, Paper CIE-48236, Chicago, IL, September 2003.

[Hall80] P. Hall and G. Dowling. Approximate String Matching. ACM Computing Surveys. 12(4):381-402, 1980.

[Hirs97] D.S. Hirschberg. Serial computations of Levenshtein distances. Pattern Matching Algorithms, 4:123--142, 1997.

[Hirt01] J. Hirtz, R. Stone, D. McAdams, S. Szykman, and K. Wood. A functional basis for engineering design: reconciling and evolving previous efforts. Research In Engineering Design, 13(2):65-82, March 2002.

[Hu00] X. Hu, J. Pang, Y. Pang, M. Atwood, W. Sun, and W.C. Regli. A survey on design rationale: representation, capture and retrieval. In 5th DFM Conference, Paper DFM-14008, Baltimore, September 2000.

[Karn05] M. Karnik, D.K. Anand, E. Eick, S. K. Gupta, and R. Kavetsky. Integrated Visual and Geometric Search Tools for Locating Desired Part in Part Database. Accepted in CAD'05 Conference, Bangkok, Thailand, June 2005.

[Kirs98] C. F. Kirschman and G. M. Fadel. Classifying functions for mechanical design. ASME Journal of Mechanical Design, 120(3) 475-482, 1998.

[Kolo93] J. L. Kolodner. Case-Based Reasoning, Morgan Kaufmann Publishers, San Mateo, California. 1993.

[Lee97] J. Lee. Design Rationale Systems: Understanding the Issues. IEEE Expert, 12(3) 78-85, 1997.

[Li04] Z. Li, M. Liu, and K. Ramani. Review of Product Information Retrieval: Representation & Indexing. In CIE Conference, Paper CIE-57749, Salt Lake City, UT, September 2004.

[Mahe97] M. L. Maher and P. Pu (eds.). Issues and Applications of Case-based Reasoning in Design, Lawrence Erlbaum Associates, Hillsdale, NJ, 1997.

[Mock04] G. Mocko, R. Malak, C. Paredis, and R. Peak. A Knowledge Repository for Behavioral Models in Engineering Design. In CIE Conference, Paper CIE-57746, Salt Lake City, UT, September 2004.

[Neel03] J. Neelamkavil and M. Kernahan. A Framework for Design Knowledge Reuse. In CIE Conference. Paper CIE-48215, Chicago, IL, September 2003.

[Nom04] Y. Nomaguchi, A. Ohnuma, and K. Fujita. Design Rationale Acquisition in Conceptual Design by Hierarchical Integration of Action, Model and Argumentation. In CIE Conference, Paper CIE-57681, Salt Lake City, UT, September 2004.

[Panc04] J. H. Panchal, M. G. Fernandez, C. J. J. Paredis, J. K. Allen, and F. Mistree. Designing Design Processes in Product Lifecycle Management: Research Issues and Strategies. In CIE Conference, Paper CIE-57742, Salt Lake City, UT, September 2004.

[Pena95] F. Pena-Mora, D. Sriram, R. Logcher. Design rationale for computer-supported conflict mitigation. ASCE Journal of Computing in Civil Engineering, 57-72, 1995.

[Shah95] J.J. Shah and M. Mantyla. Parametric and Feature-Based CAD/CAM. John Wiley and Sons, New York, 1995.

[Ston00] R.B. Stone and K.L. Wood. Development of a functional basis for design. Journal of Mechanical Design, 122(4):359-370, December 2000.

[Shoo00] S. Shooter, W. T. Keirouz, S. Szykman and S. Fenves. A model for information flow in design. In ASME Design Theory and Methodology Conference, Baltimore, Maryland, September 2000.

[Sunn03] S. Sunnersjo, I. Rask, and R. Amen. Requirement-Driven Design Process with Integrated Knowledge Structures. In CIE Conference, Paper CIE-48218, Chicago, IL, USA, September 2003.

[Syca92] K. Sycara, D. NavinChandra, R. Guttal, J. Koning, and S. Narasimhan. CADET: a case-based synthesis tool for engineering design. International Journal of Expert Systems, 4(2)157-188, 1992.

[Szyk01] S. Szykman, R.D. Sriram and W. C. Regli. The role of knowledge in next-generation product development systems. Journal of Computation and Information Science in Engineering, 1(1):3-11, 2001.

[Umed96] Y. Umeda, M. Ishii, M. Yoshioka, Y. Shimomura and T. Tomiyama. Supporting conceptual design based on the function-behavior-state modeler. Artificial Intelligence for Engineering Design, Analysis and Manufacturing, 10:275-288, 1996.

[Varg04] N. Vargas-Hernandez and J.J. Shah. 2nd-CAD: a tool for conceptual systems design in electromechanical domain. Journal of Computing and Information Science in Engineering, 4(3):28-36, 2004.

[Xu04] C. Xu, S.K. Gupta, and Z. Yao. A framework for conceptual design of multiple interaction state mechatronic systems. In Tools and Methods of Competitive Engineering Conference, Lausanne, Switzerland, April 2004