Design Repositories: Next-Generation Engineering Design Databases

Simon Szykman, Ram D. Sriram, Christophe Bochenek, Janusz W. Racz and Jocelyn Senfaute
National Institute of Standards and Technology (NIST)
Manufacturing Systems Integration Division
Building 304, Room 6
Gaithersburg, MD 20899
szykman@cme.nist.gov, sriram@cme.nist.gov,
bchrist@cme.nist.gov, jwracz@cme.nist.gov, and senfaute@cme.nist.gov

INTRODUCTION

Design of complex engineering systems is increasingly becoming a collaborative task among designers or design teams that are physically, geographically, and temporally distributed. The complexity of modern products means that a single designer or design team can no longer manage the complete product development effort. Developing products without sufficient expertise in a broad set of disciplines can result in extended product development cycles, higher development costs, and quality problems. On the other hand, ensuring comprehensive technical proficiency in a world where trends are toward more multidisciplinary design can become a costly undertaking for a company.

Driven by such issues, companies are increasingly staffing only their core competencies in-house and depending on other firms to provide the complementary design knowledge and design effort needed for a complete product. Designers are no longer merely exchanging geometric data, but more general knowledge about design and design process, including specifications, design rules, constraints, rationale, etc. As design becomes increasingly knowledge-intensive and collaborative, the need for computational design frameworks to support the representation and use of knowledge among distributed designers becomes more critical. Due to the explosive growth of the Internet and associated information infrastructure, as well as the ubiquity of World Wide Web browsers, the use of the Internet and the World Wide Web as a medium for communications and information transfer is increasing.

While advances in the area of Internet computing have improved the means for sharing and exchanging information, the more significant barrier to product development is not the problem of providing distributed access to distributed information, but of finding the information that's needed. The need for rapid retrieval and subsequent reuse of knowledge, driven by pressure to reduce product development times in

industry, has resulted in an increased focus on methods for representing and storing engineering artifact knowledge. Traditional design databases, which merely provide access to schematics, computer-aided design (CAD) models, and documentation, are inadequate for this purpose. The emerging research area of design repositories is aimed at addressing these industry needs.

A design repository is an intelligent knowledge-based design artifact modeling system used to facilitate the representation, capture, sharing, and reuse of corporate design knowledge. It should be noted that although the term *design repositories* has not yet found its way into daily usage in industry, many companies are migrating from traditional design databases in the direction of design repositories. Design repositories are distinguished from traditional design databases in several significant ways:[†]

- Traditional design databases are typically more data-centric than knowledge-centric, and contain only a limited representation of an artifact such as drawings and/or CAD models, version information, and often related documentation. Design repositories attempt to capture a more complete design representation that may include characterization of function, behavior, design rules, simulation models, and so on. It should be noted, however, that a fully comprehensive representation of every aspect of a design may simply not be possible.
- Design databases are generally more homogeneous in the kinds of information they contain (e.g. images, CAD models, and unstructured text/documentation). Design repositories can include these, but may also include formal data/information models, structured text (specialized languages for representing function, design rules, logical expressions), mathematical simulation models and more. Design databases tend to be static sources of information though their contents may grow with time.
- Design repositories are designed not only for storage of information, but to support retrieval and reuse of design knowledge using sophisticated methods that are not incorporated into traditional database systems. Such capabilities might include search for components/assemblies that satisfy required functions, explicit representation of physical and functional decompositions and the mappings between them, simulation of behavior and performance, (partially) automated reasoning about a design, and more. Since design databases have not been designed specifically for these purposes, they are limited in their ability to meet needs for design of large-scale engineering systems.

THE NIST DESIGN REPOSITORY PROJECT

The NIST Design Repository Project is an ongoing project at the National Institute of Standards and Technology (NIST) that involves research toward providing a technical foundation for the creation of design repositories—repositories of heterogeneous knowledge and data that are designed to support repre-

[†] Just as the word "database" can refer to either a database management system or an individual information store and its content, in this paper, the term design repository will be used to describe both the modeling system (underlying representation, interfaces and mechanisms) as well as an specific design artifact model and its content. The intended meaning in a particular instance should be clear from the surrounding context.

sentation, capture, sharing, and reuse of corporate design knowledge. This research is driven by industry needs that were identified at a workshop held at NIST in November, 1996.[1] The infrastructure being developed consists of formal representations for design artifact knowledge and web-based interfaces for creating repositories. The scope of this project also includes the development of prototype repositories that will be made available online via the World Wide Web.

Through the course of this project, a variety of research issues have arisen that will in the long term affect the way in which design repositories are implemented and used. These issues include:

- Development of an information modeling framework to support modeling of engineering artifacts that provides a more comprehensive knowledge representation than traditional CAD systems.
- The use of standard representations, when possible, to leverage research efforts and maximize interoperability with existing software used in industry, and contribution to long-term standards development where standards currently do not exist. The latter consists of the development of information models for representation of knowledge that is generally not found in traditional CAD systems, such as function, behavior, etc.
- A need for representations that are both human interpretable and machine interpretable so that information stored in a repository is accessible and usable by both human designers and knowledge-based design systems that might be used for (partially) automating part of a design process.
- Development of taxonomies of standardized terminology to help provide consistency in, and across, design repositories, as well as to facilitate indexing, search, and retrieval of information from them.
- Implementation of interfaces for creating, editing, and browsing design repositories that are easy to use and effective in conveying information that is desired.

The degree to which these issues have been addressed, to date, within the NIST Design Repository Project varies. However, these issues are all important to the role of design repositories in industry, and ultimately all will have to be resolved by the research community before design repositories can successfully transition into engineering industry. In addition to these research issues, technological issues such as database and Internet access issues also impact this project. Other pragmatic issues such as security of communications and protection of intellectual property when sharing or exchanging design knowledge will also have to be addressed before design repositories become widely used. These issues are beyond the scope of this paper.

The goal of this research is not to provide commercial-quality software for creating design repositories, but to provide prototypes that show the utility of the design repository concept. By highlighting and starting to address these issues, this work will speed up the transition and adoption of these kinds of technologies into industry.

THE INFORMATION MODELING FRAMEWORK

Traditional CAD systems are limited to representation of geometric data and other types of information relating to geometry that may include constraints, parametric information, features, and so on. Because of industry's increasing dependence on other types of knowledge in the design process, new classes of tools to support knowledge-based design, product data management (PDM), and concurrent engineering have begun emerging in the engineering software marketplace. When contrasted with traditional CAD tools, these new systems are making progress toward the next generation of engineering design support tools. However, these systems have focused primarily on database-related issues and do not place a primary emphasis on information models for artifact representation. Furthermore, although these systems can often represent non-geometric knowledge (e.g. information about the design process, manufacturing process, bills of materials, etc.), representation of the *artifact* itself is still generally limited to geometry.

Research in the area of intelligent design systems has typically taken an approach to modeling that attempts to integrate three fundamental facets of an artifact representation: the physical layout of the artifact (form), an indication of the overall effect that the artifact creates (function), and a causal account of the operation of the artifact (behavior). Different models of this type have been developed by various researchers, including [2, 3, 4, 5, 6, 7] and [8] among others.

While there are major differences in the implementations of such models, the top level division into representation of form, function, and behavior is a common one. The NIST Design Repository Project utilizes an approach which incorporates these three concepts. This project, building on previous design representation research, has resulted in an object-oriented artifact representation language that provides a high level division into *form*, *function*, and *behavior*. A significant distinction between the research presented in this paper and previous research is the implementation of web-based interfaces as part of this work to support distributed access to knowledge. Another focus of the NIST Design Repository Project is the attempt to address terminological and taxonomic issues in artifact modeling. The need for standardized terminology in design artifact modeling is often overlooked in the literature; however, it is an issue of critical importance.

This modeling language represents artifacts as sets of *objects*, and *relationships*. *Objects* represent physical entities such as assemblies, subassemblies, and components, as well as non-physical concepts such as function and behavior. In the remainder of this paper, the term *artifact* is used in a generic sense and may refer to assemblies, subassemblies or individual components. *Relationships* are used to represent the relationship between sets of objects, including a physical decomposition (of an assembly into subassemblies), a functional decomposition (of a function into subfunctions), and other kinds of relationships between objects. The overall artifact representation is comprised not only of the collection of the objects that represent physical entities, but also the other objects, relationships, the interconnections between them, as well as various attributes and their values.

As is true in general for object-oriented representations, objects and relationships are instantiated from classes that contain attributes and other information that are transferred via inheritance mechanisms. Thus, *a priori* development of useful generic classes of objects and relationships makes modeling of a specific artifact easier since much of the required attributes are already represented in the class schemata. The artifact modeling language and associated object-oriented representation are described in greater detail in [5], as are .

KNOWLEDGE REPRESENTATION

In order to leverage research efforts and maximize interoperability with existing software used in industry, this work makes use of standard representations when possible. For representation of geometry, this work uses ISO 10303 which is more commonly known as STEP (Standard for the Exchange of Product Model Data; see <http://www.nist.gov/sc4/>), specifically AP (application protocol) 203. As more and more CAD companies incorporate the ability to import and export STEP data, the use of STEP AP 203 as a neutral format will make the NIST design repositories more widely accessible than would a proprietary, CAD system-specific format.

In addition to the STEP AP 203 format, geometry is also maintained in the Virtual Reality Modeling Language (VRML; see <http://www.vrml.org/VRML2.0/FINAL/>) format to facilitate visualization of repositories that may be accessed remotely via the World Wide Web. Although STEP AP 203 viewers are available, they are neither as pervasive nor as well integrated with web browsers as are VRML viewers. VRML is not suitable as a complete replacement for STEP AP 203; however, because it was designed for graphical display of geometry and not as a general geometric representation, VRML is not able to represent all kinds of geometric information needed by designers.

Within the standards development community, the focus to date has been primarily on representation of geometric data. This is clearly essential, but it is no longer sufficient. As industry's reliance on non-geometric information and knowledge-based design increases, so does the need for standards for design knowledge such as function, behavior, and other kinds of non-geometric data. In addition to geometry, the information modeling framework that has been developed within this project presently includes representation of function and behavior, physical decompositions, functional decompositions, and the mappings between the physical and functional domain, etc. Long term plans include extensions to the framework to include other kinds of non-geometric information, such as design rationale. Although formal standards for this kind of information are not yet being explored in the standards community, it is hoped that work done within the National Institute of Standards and Technology will lay a foundation for long-term standards development.

As an example to illustrate the structure of the schemata that have been developed, the generic schema for the function information model is shown in Figure 1, where a word in brackets ("[ ]") indicates a reference to another data structure, and braces ("{ }") indicate a list of references to other data structures.

```
Function
   Name               string
   Type               [Generic_function_class]
   Documentation      string        (or NULL)
   Methods            string        (or NULL)
   Input_flow         {[Flow]}      (or NULL)
   Output_flow        {[Flow]}      (or NULL)
   Subfunctions       {[Function]}  (or NULL)
   Subfunction_of     [Function]    (or NULL)
   Referring_artifact [Artifact]
```

Figure 1. Generic Schema for Representation of Function

The `Name` of the function is a string, and is required to be unique. The `Type` is a reference to a generic function class that is part of a function class taxonomy (to be discussed below). `Documentation` is a string used to describe the function. In cases where a description is somewhat long, this string can consist of or include file paths or web universal resource locators (URLs) that lead to more information, images, etc. `Methods` is also a string and can also be a file path or web URL. This item differs from `Documentation` in that `Methods` is intended to include computer-processable information (such as a computer program, code fragment, rules, constraints) to support computer-based reasoning about a design. The next two items in the schema are `Input_flow` and `Output_flow`. These are references to lists of input and output flows for the function. To illustrate the relationship between functions and flows, if an artifact converts direct current to rotational motion, the function is to *transform* and the input and output flows are *direct current* and *rotational motion*, respectively. The next item in the schema is `Subfunctions`. This item is a list of references to other function data structures, allowing a function to be decomposed into multiple subfunctions each of which may have its own associated input and output flows. The decomposition enabled by `Subfunctions` provides the means to map complex functionality to more detailed portions of an artifact model. The next item in the schema is `Subfunction_of`, which can be thought of as the inverse of a reference indicated by `Subfunctions`. In other words, if function A has functions B and C as subfunctions, then B and C are subfunctions of A and will list function A under `Subfunction_of`. The last item in the function schema is `Referring_artifact`. This is a reference from a function back to the artifact that references it.

Figure 2 shows a schematic representation of some of the data structures and references for a mechanism that is part of a fluid pump design, illustrating how generic schemata can be used to capture knowledge about the physical and function domain, and mappings between the domains. This mechanism takes

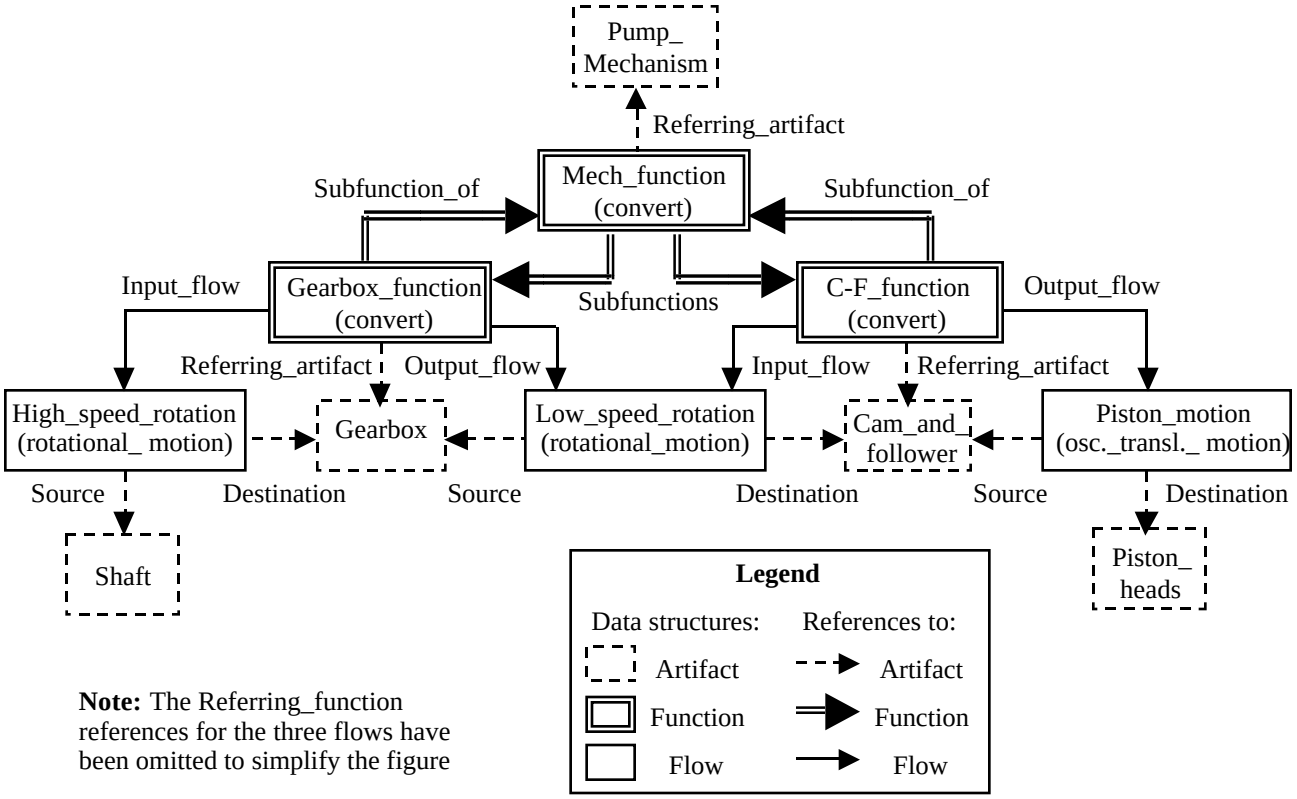


Figure 2. Graphical Illustration of Pump Mechanism Function Representation

the rotational motion from a rotating shaft (which is driven by a motor) and converts it to oscillatory translational motion used to drive the pump piston heads.

At one level, this mechanism can be considered as a single artifact, having an input flow (rotational motion) whose source is a motor shaft, and an output flow (oscillatory translational motion) whose destination is the pump heads. However, the function of this mechanism is actually more complex, consisting of multiple subfunctions each satisfied by different portions of the mechanism. The mechanism accomplishes the conversion of motion described above as follows: a motor drives a shaft, which enters a gearbox; the gearbox reduces the speed of rotation, and the output motion drives a camshaft; the cam followers have links to the pump piston heads, resulting in an output at the piston heads that is a oscillatory translational motion. These multiple subfunctions are represented individually and mapped appropriately back to the physical artifact domain, as shown in the figure.

Design repositories are intended to serve as information-rich stores of corporate design knowledge. They are not, however, intended simply to be used as sophisticated part catalogs where a designer can search for final parts or subassemblies that can be dropped into a new design. In most cases, artifact knowledge retrieved from previous designs will not be completely applicable to a similar problem. A design may require further modification before being usable in a new design. In many instances, even a

modified design may not be applicable. A designer can still benefit from retrieval of knowledge about previous designs by abstracting information and applying it to a new design, or by gaining insight into how an earlier related artifact was designed. Another way in which stored design knowledge can be used is to support design automation. Knowledge-based design systems are becoming more prevalent in industry because of their ability to automate (or partially automate) design reasoning and in some cases to take on the role of designer for selected portions of the design process.

From the knowledge representation standpoint, these two uses produce conflicting requirements. Although natural language is the most expressive and easily comprehended format for humans, for the purposes of storage and retrieval, as well as computer-based reasoning, natural language is a poor choice and formal representations are preferable. Conversely, a formal representation may be of little use to a human designer if extracting information from it is too difficult. Thus a key objective of this research has been to provide knowledge representations that are both human and machine interpretable, so that information stored in a design repository is accessible and usable by both human designers and knowledge-based design systems.

The information modeling framework that has been developed makes use of a formal knowledge representation, but one that uses data structures that are comprehensible by a human designer, and which lends itself to browsing an artifact representation in a meaningful manner given suitable interfaces. A current effort in this project involves the mapping from the generic schemata and data structures into the Extensible Markup Language (XML; see <http://www.w3.org/TR/REC-xml>), a language similar in appearance to the Hypertext Markup Language (HTML; see <http://www.w3.org/MarkUp/html-spec/html-spec_toc.html>) but which allows the development of user-defined tags, various kinds of references, and other mechanisms. XML is also both human- and machine-interpretable, but provides a better representation to support the implementation of software systems based on this information modeling framework.

The advantages of XML over the generic schemata that were initially developed (as well as advantages over other alternatives for formal information modeling languages) stem from its widespread adoption in the information technology world. More specifically, XML support is expected in upcoming versions of several commercial web browsers and word processing applications, in addition to a number of XML authoring and development tools that are currently available. For example, generic XML parsers already exist, freeing software developers from the burden of writing parsers to read XML data.

In cases where multiple collaborators are using different systems, exchange of knowledge bases or databases can become problematic. Since XML is an ASCII text-based language, the XML-based representations will also facilitate interoperability and knowledge exchange among distributed designers, design teams, and companies. The mappings into XML schemata for the representation of function and flows, as well as function and flow taxonomies have already been created.[9] Similar mappings for other aspects of the artifact representation are under development.

TAXONOMIES AND TERMINOLOGY

The need for standardized terminology in function-based design is often overlooked in the literature; however, it is an issue of critical importance for a number of reasons. The first reason is to reduce ambiguity at the modeling level. Ambiguities can occur when multiple terms are used to mean the same things, when the same term is used with multiple meanings. The distillation of a large body of terms into concise taxonomies does not eliminate this problem entirely, but it significantly lessens its occurrence.

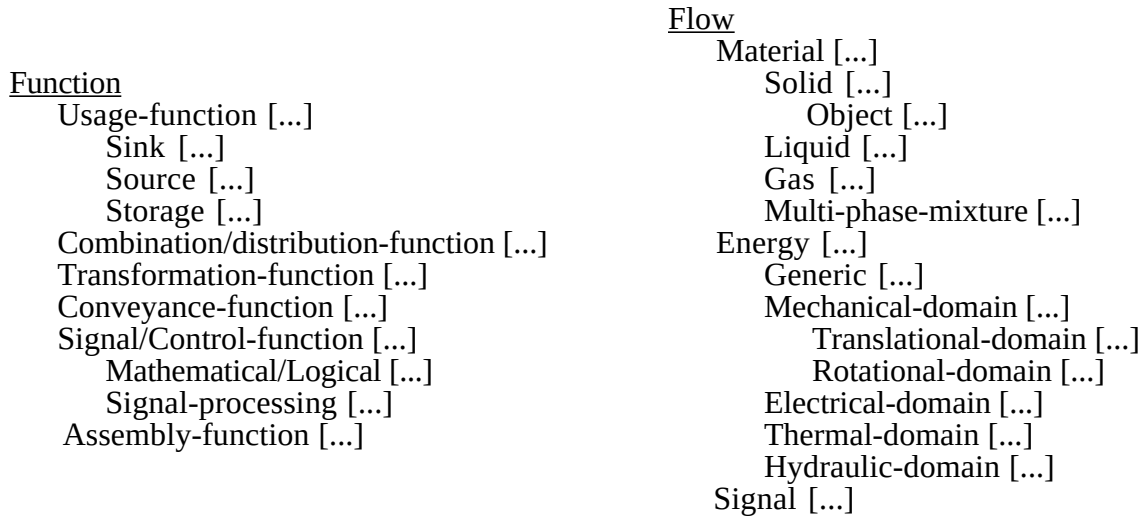
A related issue is that of uniqueness, not at the level of individual terms as with synonyms, but at the concept level. The larger the number of terms there are in a vocabulary, the more different ways there are to model or describe a given concept. This makes processing of information that has been represented more difficult, whether it be a human trying to interpret information modeled by somebody else, or whether it be algorithms developed for design automation or reasoning about a design. This problem is mitigated by taking a minimalistic approach regarding terminology. In practice, it is impossible to have a vocabulary that allows all concepts to be modeled, but only in one unique way, as the flexibility required for representation of a broad set of concepts is what results in multiple ways of expressing the same concept. However, to whatever extent uniqueness problems at the concept level can be reduced, interpreting information that is represented can be made easier.

A third reason for developing a standardized terminology is that it increases the uniformity of information within artifact models. Providing a greater degree of consistency within and across design repositories will facilitate indexing, search, and retrieval of information from them.

The first phase of taxonomy development to support the creation of design repositories has focused on the area of function representation. In addition to the development of schemata for the representation of engineering functions and associated flows, a pair of generic taxonomies of function and flow have been developed which are concise, yet comprehensive enough to allow the modeling of function for a broad variety of engineering artifacts. The top-level divisions of the two taxonomies are shown in Figure 3. The indentation of terms identifies functions or flows that are subtypes of a more generic type; the bracketed ellipsis "[...]" indicate that each of the types listed actually has additional terms as subtypes that are not listed in the abbreviated taxonomies shown in the figure. The extended taxonomies of function and flow appear in [10]. The taxonomies contain over 130 functions and over 100 flows. The evolution of both taxonomies to achieve more comprehensive coverage of engineering functions will be an ongoing part of this research.

IMPLEMENTATION

Along with creating a representational infrastructure for design artifact modeling, a second objective of the NIST Design Repository Project is to develop a computational framework for the creation of design repositories, and a proof-of concept prototype to demonstrate their benefits. This research has resulted in



(a) Function Taxonomy **(b) Flow Taxonomy**

Figure 3. Top-level Subdivisions for the Function and Flow Taxonomies

the implementation of a suite of tools for distributed development of, and access to, design repositories. The system that has been implemented includes web-based interfaces that allow design repositories to be accessed via the Internet by multiple distributed clients using common web browsers.

The Design Repository tool suite includes several components in addition to the web-based interfaces:

- ObjectStore™[‡], a commercial object-oriented database management system, developed by Object Design, Inc.
- A web-based Design Repository Editor used to create, modify and update design repositories.
- A web-based Design Repository Browser, the user interface to a design repository that allows the designer to navigate through an artifact representation
- A Design Repository Compiler which takes a formatted text file created by the Design Repository Editor and transfers the contents to an ObjectStore™ database.
- An information extractor or "decompiler" which takes the contents of a database and replicates them in a formatted text file for further editing.
- STEP/Works, a STEP AP 203 viewer developed by International TechneGroup, Inc. for local (non web-based) visualization of STEP-based geometry, desirable in some cases since VRML provides a less comprehensive representation of geometry.

[‡] Use of any commercial product or company names in this paper is intended to provide readers with information regarding the implementation of the research described, and does not imply recommendation or endorsement by the National Institute of Standards and Technology.

The web-based Design Repository Editor, which can be used via most common web browsers, greatly simplifies the design repository development process. This interface provides a designer with remote access to a design repository, allowing the user to create or edit a design artifact model by creating objects and relationships from existing classes, to add new classes of artifacts, to describe physical, functional and behavioral decompositions as well as the mappings between them. This editor is a combination of a form-based and point-and-click interface that automatically generates forms for the designer to fill out based on the artifact representation schemata. The editor handles information management issues and depending on circumstances, is able to automate creation, naming, and linking of data structures, as well as maintaining a to-do list of "empty" data structures that have been created but not yet filled out with information.

The other available interface is a web-based Design Repository Browser. Figure 4 shows a screenshot of a browsing session for a design repository containing the artifact model of a Black & Decker cordless power drill. The browser provides a point-and-click interface that allows the designer to browse an existing design repository. The object-oriented data structures (objects, relationships, and their classes) are presented in table form, where hyperlinks allow navigation to connected data structures. From the artifact object shown in the figure, the user can click on one hyperlink to examine the artifact's function, or another to view the relationships for that artifact (one indicating the artifact of which this object is a subassembly, and the other specifying the further decomposition of this artifact into subassemblies and components). The different types of data structures as well as the hyperlinks are color-coded to easily distinguish between them. The user can follow the artifact's form hyperlink to an object that has a link to a 3D model of that artifact, in either STEP AP 203 format or in VRML format for web-based viewing. A screenshot showing the visualization of the geometry of the power drill, stored in VRML format, is shown in Figure 5.

In addition to navigating from object to object as described, the bottom right portion of the interface provides the overall hierarchical physical decomposition of an artifact into assemblies, subassemblies, down to individual components. The user can quickly jump to any artifact object in the hierarchy by following one of those hyperlinks. As different parts of the design are visited, they are added to the history in the top right portion of the interface so that the user can quickly return to any part of the artifact model that was previously viewed during a browsing session. Figure 6 illustrates the architecture of the data interactions involved with the Design Repository Browser.

As the interfaces described are intended to provide access to large bodies of knowledge, usability issues are of primary importance. Through the creation of the existing prototype repositories, substantial input has been obtained from users regarding the modes of access to and presentation of information. The implementation of new versions of both the editor and browser interfaces is being undertaken to address these issues.

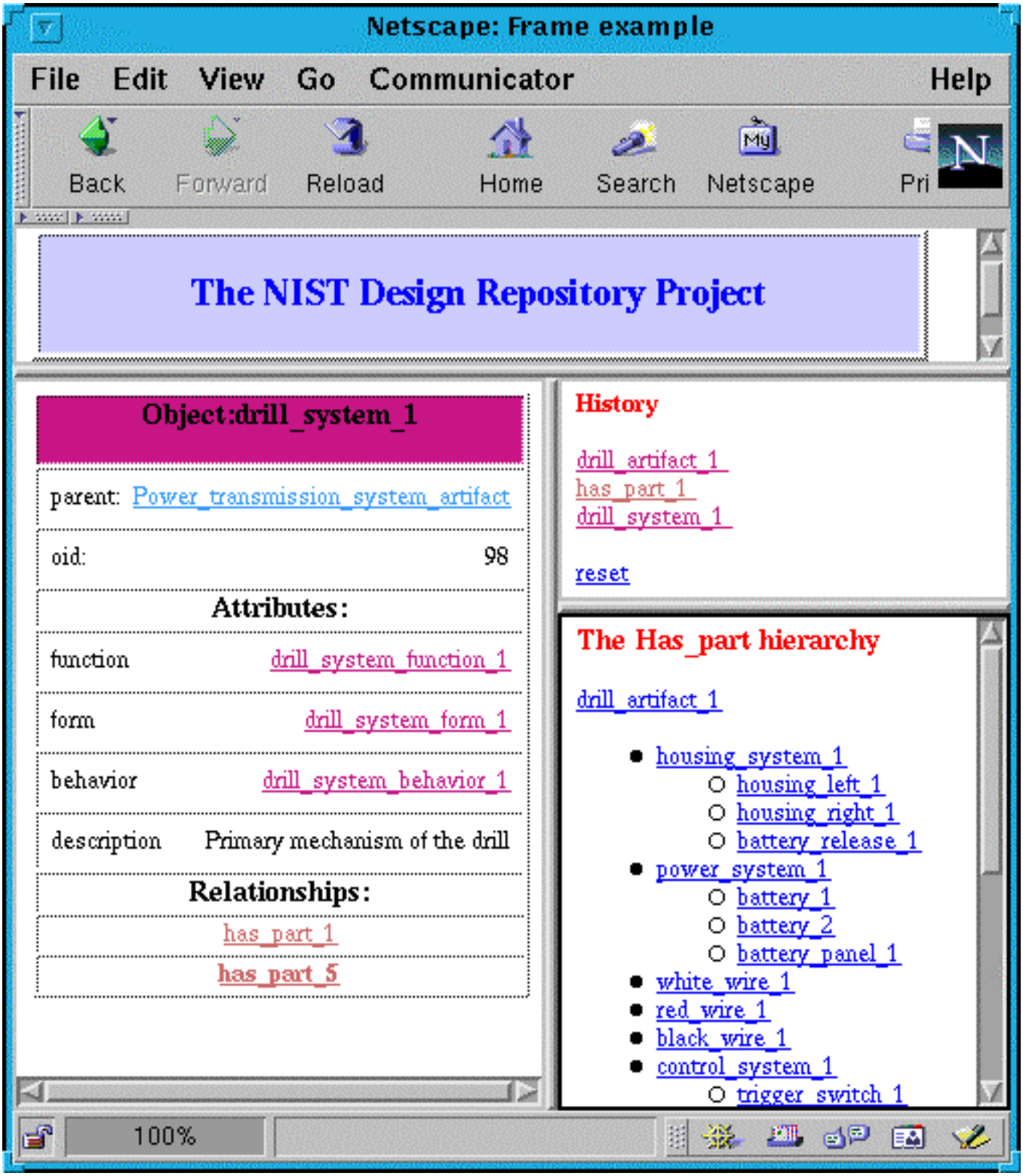


Figure 4. Web-based artifact browser interface.

Figure 5. Visualization of the Power Drill Geometry.

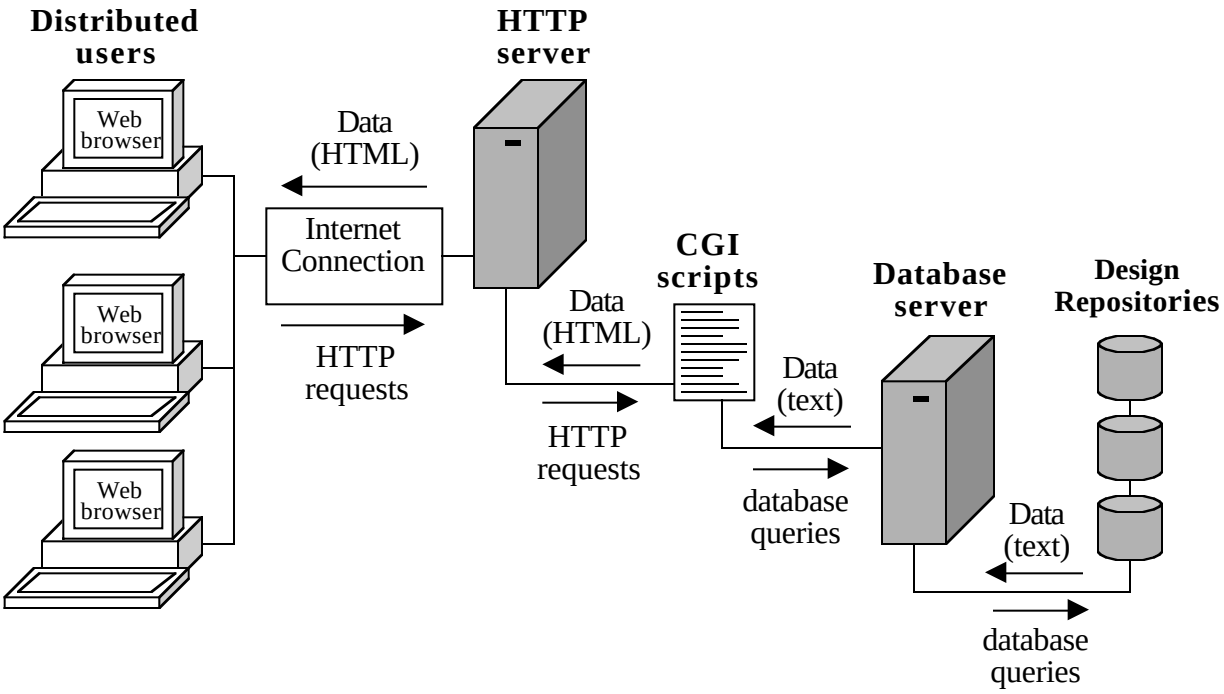


Figure 6. Architecture of Design Repository Browser Data Interactions.

INFORMATION PROCESSING AND KNOWLEDGE RETRIEVAL

Now that a formal specification for representation of artifact function information a prototype artifact modeling system have been created, the next step is to begin generating algorithms for information processing and knowledge retrieval. A small number of algorithms for information processing and reasoning have been identified and devised, though they have not yet been implemented. For example, an algorithm has been designed to search an artifact representation to identify input and output flows for functions that are decomposed into subfunctions. For the mechanism example shown in Figure 2, this algorithm would determine that the input and output flows for the pump mechanism function are rotational motion and oscillatory translational motion, even though these flows are associated not with the mechanism's function, but with its subfunctions. This algorithm can process complex functions that may be decomposed into more than one level in depth, and with subfunctions having multiple input and output flows that may reference different source and destination artifacts.

Once a large enough number of design artifacts has been modeled, the set of function models can serve as a database for knowledge retrieval and reuse. Using the above algorithm, queries can be made to search for particular function structures; a search for something that converts rotational motion to oscillatory translational motion, for instance, would turn up the pump mechanism. Since the schemata that have been developed also capture various kinds of properties, these can be included as search criteria. A search

could be made not only for a function structure that converts electrical energy to rotational motion, but one whose input is electrical energy in the form of direct current at no more than twelve Volts.

Furthermore, the formal representation that has been developed can support even more sophisticated type of queries. As illustrated by the graphical interpretation of the pump mechanism shown in Figure 2, the artifact representation can be mapped to a graph where the various data structures (functions, flows, artifacts) are nodes in the graph and references among data structures are arcs. Sophisticated graph-based pattern matching algorithms have been developed in the field of computer science that could be used for even more intelligent kinds of searches. A search could therefore attempt to match not only a function with certain inputs and outputs, but could also target or avoid certain kinds of subfunctions or internal flows. Once such search algorithms are implemented, searches could target or avoid certain kinds of artifacts, or could attempt to find function structures that satisfy certain functions and that have fewer than *X* parts, and so on.

Searches of these kinds are virtually impossible using traditional design databases where function is either described in some kind of natural language documentation, or even more often not explicitly captured at all. Thus, in addition to providing a neutral language for capturing and exchanging artifact information, the representation described in this work provides a means for the creation of corporate design knowledge archives and repositories that support more effective retrieval and reuse of knowledge.

CONCLUSIONS AND FUTURE DIRECTIONS

The NIST Design Repository Project is working toward the development of an information modeling framework to support the creation of design repositories. This project is driven by industry needs for technology to support the increasing role of knowledge-based design, including the representation, capture, sharing, and reuse of corporate design knowledge. This paper has presented the project objectives, associated research issues, and the current status of the project in the context of those issues. The prototype implementation was also described.

Focus of work to date has been in four areas: (1) the development of a design modeling language for representing design artifact knowledge, (2) the implementation of interfaces for creating, editing, and browsing artifact repositories, (3) work toward identifying taxonomies of functions and associated flows that are small, yet generic enough to model a broad variety of engineering artifacts, and (4) the creation of a prototype design repository. In addition to continuing these aspects of the project and populating the design repositories with new design artifacts, there are several important areas of research that will be addressed by future project milestones.

The main aspects of the design modeling language are the representation of form, function, and behavior (as well as relationships between their associated objects). While the artifact representation currently captures information from the physical (form) and functional domains, and mappings between them, the representation of behavior within the existing framework is a relatively simple descriptive text-

based approach. Another objective for future work is to incorporate more a formal representation of behavior to allow composable simulations.

Work in this area has been done as part of two projects that are funded by the Defense Advanced Research Projects Agency (DARPA) Rapid Design Exploration and Optimization (RaDEO) program, for which NIST was a funding agent: the Model-Based Support of Distributed Collaborative Design (previously How Things Work) project at the Stanford University Knowledge Systems Laboratory[11] and the Active Catalog project at the University of Southern California Information Sciences Institute.[12] The possibility of incorporating an existing behavior modeling language into the NIST Design Repository Project will initially be explored before attempting to create a new one.

The mapping of the generic schemas into XML-based schemata has been presented as a means for better supporting software implementations based on the representations that have been developed. The XML mappings for the schemata for function and flow have already been completed. Short term efforts will be directed toward achieving similar mappings for other schemata within the artifact representation language. In addition to simplifying software implementations, the XML-based schemata will also facilitate communication of artifact information among distributed designers, design teams, and companies by providing a neutral format for exchange of data which might otherwise be stored in, and therefore tied to, one particular vendor's database management system. The formalization of more easily implemented representations for artifact modeling, as well as the other taxonomic issues discussed in this paper will facilitate the tasks of indexing design knowledge for efficient search and retrieval. The next step will be to develop the indexing and search mechanisms themselves, as described in the previous section.

There is also a need to expand the representational capabilities of the current system beyond the kinds of information that have already been incorporated into the existing modeling language. Most notably, although the representation presently captures a physical decomposition of a design artifact, it lacks a detailed assembly model that describes the various components of this decomposition. The current artifact representation will be expanded to capture assembly-related information (e.g. assembly constraints, mating information, and so on). In addition, the issue of representation of design rationale is very important to effective reuse of engineering design knowledge. At one level, some portion of design rationale is captured in the current approach, by virtue of explicitly representing an artifact's function, and mapping the functions to flows, which are then mapped back to the physical domain. But design rationale is a more complex issue that extends beyond artifact function.

The main drawback of the current implementation is speed of information access. A new architecture is under development that addresses this issue in a number of ways. First, queries will be served more quickly because larger amounts of information will be available in active memory (i.e., RAM) in a server-side JAVA™ application called a servlet, rather than having to make repeated time-consuming queries to a commercial database back-end as with the current implementation. Second, some of the processing of information for presentation purposes will be done on the client side by the web browser using JavaScript.

This allows a larger amount of information to be sent to the client with each query, which very slightly increases the time associated with a single query but reduces overall lag time by substantially reducing the number of queries to the server that need to be made. Furthermore, users of our initial implementation have provided substantial input regarding the presentation of information through the interfaces. This feedback has resulted in a significant redesign of the interfaces intended for the new architecture currently under development.

The NIST Design Repository Project was motivated by industry needs and as such, it is expected that industry will benefit from the technical foundation for the development of design repositories provided by this research. Another potential use for this work is to enhance the content of patent databases residing at the U.S. Patent and Trademark Office. Currently, patent information consists of text and two-dimensional images. Providing a formal artifact representation that extends to function and behavior not only leads to a more comprehensive description of a device, it also provides additional types of information for meaningful indexing, which can lead to more efficient search and retrieval of patent information.

REFERENCES

1. Szykman, S., R. D. Sriram, and S. J. Smith (eds.), Proceedings of the NIST Design Repository Workshop, NISTIR 6159, National Institute of Standards and Technology, Gaithersburg, MD, 1998.
2. Goel, A., A. Gomez, N. Grue, J. W. Murdock, M. Recker and T. Govindaraj, "Explanatory Interface in Interactive Design Environments," Artificial Intelligence in Design '96, J. S. Gero (ed.), Kluwer Academic Publishers, Boston, 1996.
3. Gorti, S. R., A. Gupta, G. J. Kim, R. D. Sriram, and A. Wong, "An Object-Oriented Representation for Product and Design Processes," *Computer-Aided Design*, Vol. 30, No. 7, 1998, pp. 489-501.
4. Qian L. and J. S. Gero, "Function-Behavior-Structure Paths and Their Role in Analogy-Based Design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, Vol. 10, No. 4, 1996, pp. 289-312.
5. Szykman, S., J. W. Racz, C. Bochenek and R. D. Sriram, "A Web-based System for Design Artifact Modeling," *Design Studies*, 1999 (accepted for publication).
6. Umeda, Y. and T. Tomiyama, "Functional Reasoning in Design," *IEEE Expert Intelligent Systems and Their Applications*, Vol. 12, No. 2, 1997, pp. 42-48.
7. Iwasaki Y. and B. Chandrasekaran, "Design Verification through Function and Behavior-Oriented Representations: Bringing the Gap between Function and Behavior," Artificial Intelligence in Design '92, J.S. Gero (Ed.), Kluwer Academic Publishers, Boston, 1992, pp. 597-616.
8. de Kleer, J. and J. S. Brown, "Assumptions and Ambiguities in Mechanistic Mental Models," Mental Models, D. Gentner and A. L. Stevens (Eds.), Lawrence Erlbaum Associates, New Jersey, 1983, pp. 155-190.

9. Szykman, S., J. Senfaute, and R. D. Sriram, “Using XML to Describe Functions and Taxonomies in Computer-based Design,” Proceedings of the 1999 ASME Design Engineering Technical Conferences (Computers and Information in Engineering Conference), Paper No. DETC99/CIE-9025, ASME, New York, 1999.
10. Szykman, S., J. W. Racz, and R. D. Sriram, “The Representation of Function in Computer-based Design,” Proceedings of the 1999 ASME Design Engineering Technical Conferences (International Conference on Design Theory and Methodology), Paper No. DETC99/DTM-8742, ASME, New York, 1999.
11. Chouiery, B. Y., S. McIlraith, Y. Iwasaki et al., “Thoughts on a Practical Theory of Reformulation for Reasoning about Physical Systems,” Proceedings of the Symposium on Abstraction, Reformulation, and Approximation (SARA '98), 1998, pp. 25-36.
12. Coutinhi, M., R. Eleish, J. Kim, V. Kumar, S. R. Ling, R. Neches and P. Will, “Active Catalogs: Integrated Support for Component Engineering,” Proceedings of the 1998 ASME Design Engineering Technical Conferences (Computers and Information in Engineering Conference), Paper No. DETC98/CIE-5521, ASME, New York, 1998.