Design Repositories: Engineering Design's New Knowledge Base

Simon Szykman, Ram D. Sriram, Christophe Bochenek, Janusz W. Racz, and Jocelyn Senfaute
National Institute of Standards and Technology

DRIVEN BY PRESSURE TO REDUCE PRODUCT DEVELOPMENT TIMES, INDUSTRY HAS STARTED LOOKING FOR NEW WAYS TO EXPLOIT STORES OF ENGINEERING ARTIFACT KNOWLEDGE. ENGINEERS ARE INCREASINGLY TURNING TO DESIGN REPOSITORIES AS KNOWLEDGE BASES TO HELP THEM REPRESENT, CAPTURE, SHARE, AND REUSE CORPORATE DESIGN KNOWLEDGE.

DESIGN OF COMPLEX ENGINEERing systems has increasingly become a collaborative task among teams that are physically, geographically, and temporally separated. The complexity of modern products means that a single designer or design team can no longer manage the complete product development effort. An enterprise that tries to develop products without sufficient expertise in a broad set of disciplines can suffer from long product development cycles, higher development costs, and quality problems. On the other hand, ensuring comprehensive technical proficiency in a world where the trend is toward more multidisciplinary design can become an equally costly undertaking for a company.

Driven by such issues, companies increasingly staff only their core competencies inhouse, depending on other firms to provide the complementary design knowledge and effort needed for a complete product. 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.

Although advances in 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. Industry's need for rapid retrieval and subsequent reuse of knowledge to reduce product development times has resulted in a sharper 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 aims to address these industry needs (see the "Design repositories" sidebar).

The goal of this research is not to provide commercial-quality software for creating design repositories but to offer prototypes that prove 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 NIST Design Repository Project

The National Institute of Standards and Technology Design Repository Project is an ongoing effort at NIST in Gaithersburg, Maryland. The project promotes research toward design repository creation. We have framed this research around industry needs identified at a workshop NIST held in November 1996.[1] The infrastructure we're developing consists of formal representations of design artifact knowledge and Web-based interfaces for creating repositories. The project also includes the development of prototype repositories that we will make available on the Web.

Our work to date has focused on four areas: developing a design modeling language, implementing interfaces for creating, editing, and browsing artifact repositories; identifying taxonomies of functions and associated flows; and

1094-7167/00/$10.00 © 2000 IEEE

Design repositories

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. Just as the word *database* can refer to either a database management system or an individual information store and its content, the term *design repository* can describe either the modeling system (the underlying representation, interfaces, and mechanisms) or a specific design artifact model and its content. Although the term *design repository* has not yet found its way into daily use in industry, many companies are migrating from traditional design databases toward 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 or CAD models, version information, and related documentation. Design repositories attempt to capture a more complete design representation that might include characterization of function, behavior, design rules, simulation models, and so on; however, a fully comprehensive representation of every aspect of a design might not be possible.
- Design databases generally contain only a few kinds of information (such as images, CAD models, and unstructured text or documentation). They tend to be static sources of information, although their contents can grow with time. Design repositories might also include formal data or information models, structured text (specialized languages for representing function, design rules, and logical expressions), mathematical simulation models, and more.
- Design repositories not only store information but also support retrieval and reuse of design knowledge using sophisticated methods that are not possible with traditional database systems. They can explicitly represent physical and functional decompositions and the mappings between them. Given appropriate algorithms, they can also support more sophisticated searches (for example, for components or assemblies that satisfy required functions), simulation of behavior and performance, and automation or partial automation of design reasoning better than traditional design databases. Because engineers have not designed databases specifically for these purposes, the traditional systems are limited in their utility for the design of large-scale engineering systems.

creating a prototype design repository.

The project has raised numerous research issues that will affect the way engineers implement and use design repositories, including

- the development of an information-modeling framework to support the modeling of engineering artifacts that can provide 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 the contribution to the development of long-term standards where they currently do not exist;
- the need for representations that are both human- and machine-interpretable, so that both human designers and knowledge-based design systems can access and use information stored in a repository;
- the 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; and
- the implementation of easy-to-use and effective interfaces for creating, editing, and browsing design repositories.

Besides these research issues, database and Internet access considerations also affect this project. Researchers will have to address other pragmatic issues, such as communication security and intellectual property protection when sharing or exchanging design knowledge, before engineers can share design repositories widely.

Facets of artifact representation

Research in the area of intelligent design systems has typically attempted to integrate three fundamental facets of artifact representation: the physical layout of the artifact (form), an indication of the artifact's overall effect (function), and a causal account of the artifact's operation (behavior). Various researchers have developed models that attempt to capture these facets.[2–8]

Although major differences exist in the implementation of such models, the top-level division into representation of form, function, and behavior is common. The NIST Design Repository Project uses an approach that incorporates these three concepts. Building on previous design representation research, we have developed an object-oriented artifact representation language that provides a high-level division into form, function, and behavior. A significant distinction between the research we present here and previous research is the implementation of Web-based interfaces to support distributed access to knowledge. We also 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 nonphysical concepts such as function and behavior. We use the term *artifact* in the remainder of this article in a generic sense to signify assemblies, subassemblies, or individual components. We use *relationships* to represent relations or interactions between sets of objects, including physical decomposition (of an assembly into subassemblies), functional decomposition (of a function into subfunctions), and other kinds of interactions between objects. The overall artifact representation comprises not only the collection of the objects that represent physical entities but also the other objects and relationships, the interconnections between them, their 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 through inheritance mechanisms. Thus, a priori development of useful generic classes of objects and relationships makes modeling a specific artifact easier, because many of the required attributes are already represented in the class schemata.

Knowledge representation

Traditional CAD systems represent only geometric data and other types of information relating to geometry, such as constraints, parametric information, and features. Because of industry's increasing dependence on other types of design knowledge, new classes of tools to support knowledge-based design, product data management, and concurrent engineering have emerged in the engineering software marketplace. When contrasted with traditional CAD tools, these new systems rep-

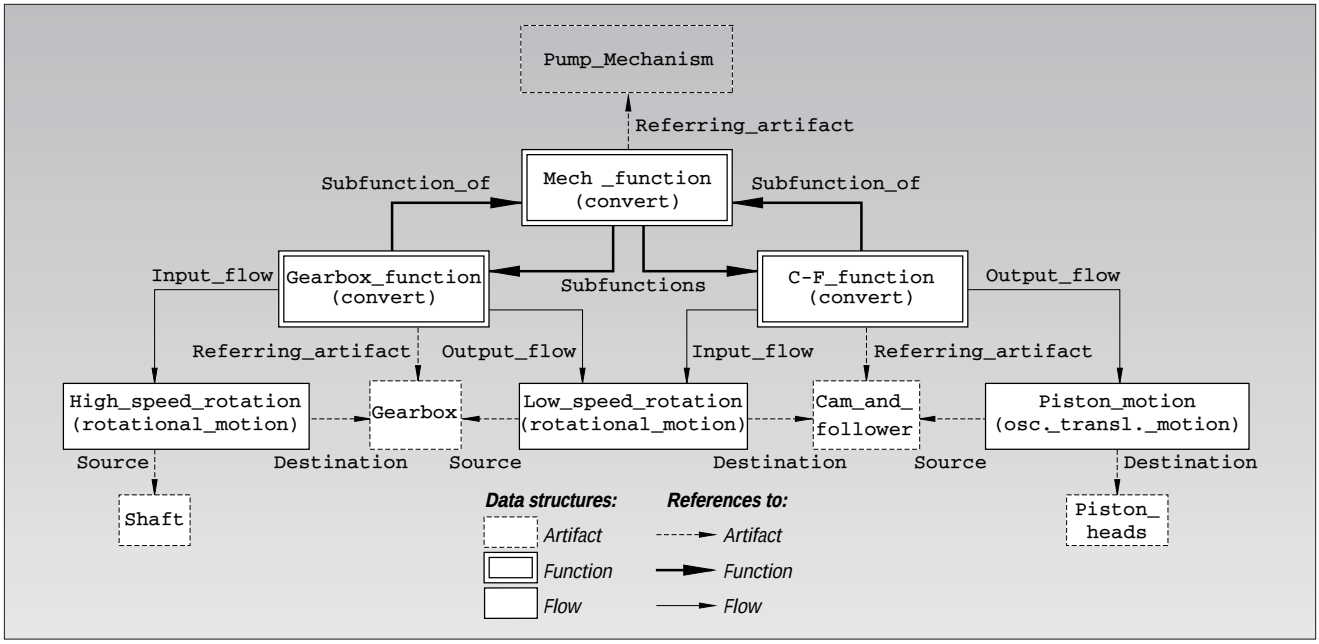

Figure 1. Graphical representation of pump mechanism function. The Referring_function references for the three flows have been omitted to simplify the figure.

resent progress toward the next generation of engineering design support tools. However, they focus primarily on database-related issues and do not emphasize information models for artifact representation. Furthermore, although these systems can represent nongeometric information—for example, about design process, manufacturing process, and bills of materials—representation of the artifact is still generally limited to geometry.

To leverage research efforts and maximize interoperability with existing software used in industry, we make use of standard representations when possible. For geometric representation, we use ISO 10303, more commonly known as the STEP (Standard for the Exchange of Product Model Data)—specifically, application protocol 203. As more CAD companies incorporate the ability to import and export STEP data, the use of the STEP AP 203 as a neutral format will make NIST design repositories more widely accessible than would a proprietary CAD-system-specific format.

In the design repository implementation, a representation of geometry is also maintained in the Virtual Reality Modeling Language. This format facilitates visualization of repositories that users can remotely access from the 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 the STEP AP 203, though—it was designed for graphical display of geometry and not for general geometric representation, so VRML cannot represent all the kinds of geometric information designers need.

The standards-development community has focused primarily on representing geometric data. This is clearly essential, but it is no longer sufficient. As industry's reliance on nongeometric information and knowledge-based design increases, so does the need for standards for design knowledge such as function, behavior, and other kinds of nongeometric data.

In addition to geometric data, this project's information-modeling framework includes representation of function and behavior, physical decompositions, functional decompositions, and mappings between the physical and functional domains. Long-term plans include extensions to the framework to admit other kinds of nongeometric information such as design rationale. Although the standards community is not yet exploring these sorts of formal standards, we expect the NIST project will lay an early foundation for such work.

The generic schema for the function information model here illustrates the structure of existing schemata:

```
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]
```

A word in brackets ([ ]) indicates a reference to another data structure, and one in braces ({ }) indicates a list of references to other data structures.

The `Name` of the function is a string and must be unique. The `Type` is a reference to a generic function class that is part of a function-class taxonomy (to be discussed later). `Documentation` is a string used to describe the function. In cases where a description is long, this string can consist of or include file paths or URLs that lead to more information, images, and so forth.

`Methods` is another string and can also be a file path or URL. This item differs from `Documentation` in that `Methods` is intended to include computer-processable information (such as computer programs, code fragments, rules, and 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`, a list of references to other function data structures. This lets a function be broken into multiple subfunctions, each of which can have its own associated input and output flows. The division enabled by `Subfunctions` lets us map complex functionality to more detailed portions of an artifact model.

The next item in the schema is `Subfunction_of`, which we can think 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 to the artifact that references it.

Figure 1 displays data structures and references for a mechanism that forms part of a fluid pump design. It illustrates how generic schemata can capture knowledge about the physical and function domains and mappings between the domains. This mechanism takes the motion from a motor-driven rotating shaft and converts it to oscillatory translational motion that drives the pump's piston heads.

At one level, we can regard this mechanism as a single artifact with an input flow (rotational motion) whose source is a motor shaft and with an output flow (oscillatory translational motion) whose destination is the pump heads. However, the function of this mechanism is actually more complex—it consists of multiple subfunctions, each of which is satisfied by different portions of the mechanism.

The mechanism accomplishes the conversion of motion as follows:

- A motor drives a shaft, which enters a gearbox.
- The gearbox reduces the speed of rotation and the output drives a camshaft.
- The cam followers have links to the pump piston heads, resulting in an output at the piston heads of oscillatory translational motion.

The structure of the schemata for function and flow represents these multiple subfunctions individually and maps them back to the appropriate physical artifact domain.

Design repositories aim to serve as rich stores of corporate design knowledge. They are not, however, intended to be sophisticated parts catalogs in which a designer can search for final parts or subassemblies to drop into a new design. In most cases, artifact knowledge retrieved from previous designs will not completely apply to a similar problem. A design might require further modification before suiting a new design. In many instances, even a modified design might not work. Even so, a designer can still benefit by retrieving knowledge about previous designs and applying it to a new design or by gaining insight into how an earlier related artifact was designed.

Another way in which designers can use stored design knowledge 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. In some cases, they can devise 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, it is a poor choice for storage, retrieval, and computer-based reasoning; for those, formal representations are preferable. Conversely, a formal representation is of little use to a human designer if extracting information from it is too difficult. Thus, a key objective of this research is to provide knowledge representations that are both human- and machine-interpretable, so that both human designers and knowledge-based design systems can access and use information stored in a design repository.

EXCHANGE OF KNOWLEDGE BASES OR DATABASES CAN BECOME PROBLEMATIC WHEN MULTIPLE COLLABORATORS USE DIFFERENT SYSTEMS.

The information-modeling framework we developed uses a formal knowledge representation but one wherein a human designer can use suitable interfaces to comprehend and browse data structures. A current effort in this project involves mapping from the generic schemata and data structures into XML, which is similar to HTML but lets users develop their own tags, various kinds of references, and other mechanisms. XML is both human- and machine-interpretable and provides a better representation to help implement software systems based on this information-modeling framework.

XML's advantages over the initially developed generic schemata (as well as over other alternatives for formal information-modeling languages) stem from its widespread adoption in information technology. XML support is expected in upcoming versions of several commercial Web browsers and word-processing applications. Numerous XML authoring and development tools are already available. For example, generic XML parsers already exist; these liberate software developers from the burden of writing parsers to read XML data.

Exchange of knowledge bases or databases can become problematic when multiple collaborators use different systems. Because XML is an ASCII text-based language, XML-based representations will facilitate interoperability and knowledge exchange among distributed designers, design teams, and companies. Throughout the course of our work, we have created XML mappings for the function and flow schemata as well as for the taxonomies of function and flow, and we are currently developing similar mappings for other aspects of artifact representation.[9]

Taxonomies and terminology

Workers often overlook the need for standardized terminology in the literature of function-based design; however, it is an issue of critical importance for a number of reasons. The first is to reduce ambiguity at the modeling level due to the use of multiple terms that mean the same thing or the use of one term with multiple meanings. Distillation of a design-storage lexicon into concise taxonomies does not eliminate this problem entirely, but it significantly lessens it.

A related issue is that of uniqueness—not at the level of individual terms, as with synonyms, but at the conceptual level. The larger the number of terms that constitute a vocabulary, the more ways there are to model or describe a given concept. This makes the processing of previously represented information more difficult, whether by a human trying to interpret information someone else has modeled or by an algorithm developed for design automation or reasoning about a design.

We can take a minimalist approach to terminology to mitigate this problem. In practice, it is impossible to devise a vocabulary that allows all concepts to be modeled in one unique way because the flexibility required to represent a broad set of concepts results in multiple ways of expressing the same concept. However, to the extent that we can identify design concepts as unique, interpreting information becomes 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

```
Function
    Usage-function[...]
        Sink[...]
        Source[...]
        Storage[...]
    Combination/distribution-function[...]
    Transformation-function[...]
    Conveyance-function[...]
    Signal/Control-function[...]
        Mathematical/Logical[...]
        Signal-processing[...]
    Assembly-function[...]
```

(a)

```
Flow
    Material[...]
        Solid[...]
            Object[...]
        Liquid[...]
        Gas[...]
        Multi-phase-mixture[...]
    Energy[...]
        Generic[...]
        Mechanical-domain[...]
            Translational-domain[...]
            Rotational-domain[...]
        Electrical-domain[...]
        Thermal-domain[...]
        Hydraulic-domain[...]
    Signal[...]
```

(b)

Figure 2. Top-level subdivisions for the (a) function and (b) flow taxonomies.

within and across design repositories will facilitate indexing, search, and retrieval of information from them.

The first phase of taxonomy development has focused on the area of function representation. Besides developing schemata for the representation of engineering functions and associated flows, we have designed a pair of generic taxonomies of function and flow that are concise yet comprehensive enough to allow the modeling of function for a broad variety of engineering artifacts.

Figure 2 shows the top-level divisions of the two taxonomies. Bracketed ellipses ([...]) indicate that each listed type has additional terms as subtypes that are not listed in the abbreviated taxonomies. We have previously published the extended taxonomies containing over 130 functions and over 100 flows.[10] We intend to further the evolution of both taxonomies to achieve more comprehensive coverage of engineering functions.

Implementation

Besides 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 the implementation of a suite of tools for distributed development of, and access to, design repositories. The system we implemented lets multiple distributed clients access Web-based interfaces to design repositories using common Web browsers.

The Design Repository tool suite includes several components besides the Web-based interfaces:

- ObjectStore, a commercial, object-oriented database-management system developed by Object Design.
- A Web-based design repository editor to create, modify, and update design repositories.
- A Web-based design repository browser, the user interface to a design repository that lets the designer 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 for local (non-Web-based) visualization of STEP-based geometry—desirable in some cases because VRML provides a less-comprehensive representation of geometry.

The Web-based design repository editor, which users can access over most common Web browsers, greatly simplifies the design

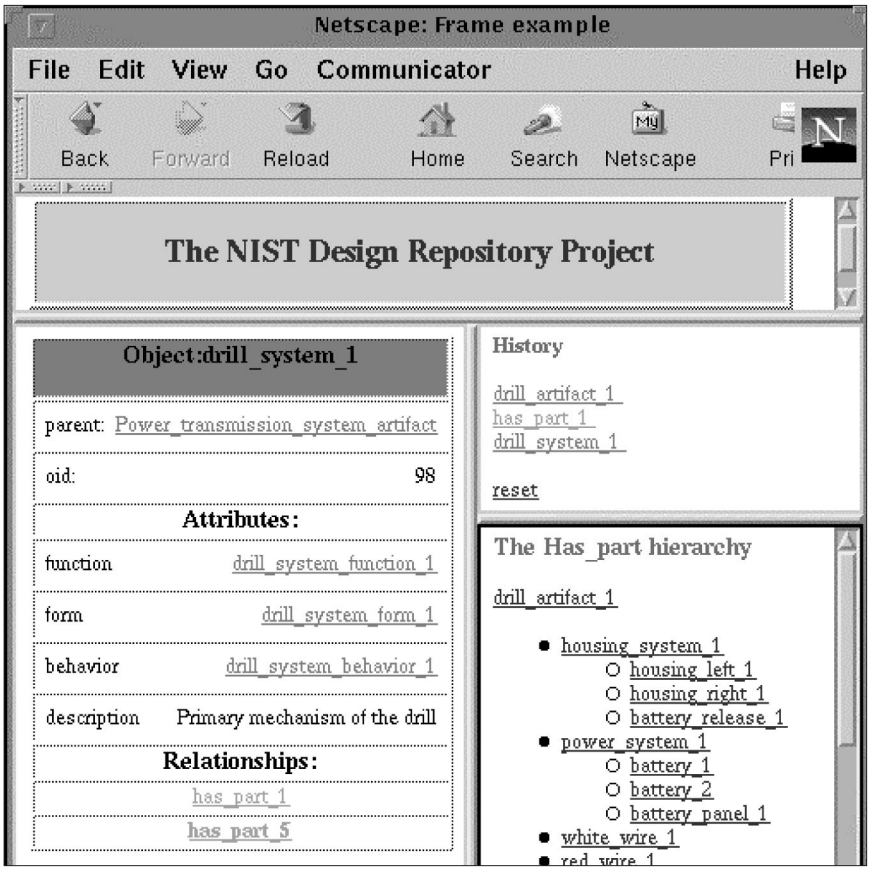

Figure 3. Web-based artifact browser interface.

repository development process. This interface provides a designer with remote access to a design repository. It lets the user create or edit a design artifact model by creating objects and relationships from existing classes, add new classes of artifacts, and 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, can automate creation, naming, and linking of data structures. It can also maintain a to-do list of "empty" data structures that were created but not yet fully described.

The other available interface is the Web-based design repository browser. Figure 3 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 lets the designer browse an existing design repository. The figure presents object-oriented data structures (objects, relationships, and their classes) wherein 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 on another to view the relationships for that artifact. (The first indicates the artifact of which this object is a subassembly; the second specifies the further division of this artifact into subassemblies and components.) The different types of data structures as well as the hyperlinks are color-coded to allow easy identification. The user can follow the artifact's form hyperlink to an object that has a link to a 3D model of that artifact—either in STEP AP 203 format or in VRML format for Web-based viewing. Figure 4 shows a screenshot demonstrating the visualization of the power drill's geometry in VRML format.

The bottom-right portion of the interface provides the overall hierarchical physical subdivision of an artifact into assemblies and subassemblies down to individual components. The user can quickly jump to any artifact object in the hierarchy by following one of the hyperlinks. As a user visits different parts of the design, the interface caches these locations in the history link in the top-right portion of the interface, letting the user quickly return to any previously viewed part of the artifact model. Figure 5 displays the architecture of the data interactions in the design repository browser.

Because we intend the interfaces described to provide access to large bodies of knowledge, usability is paramount. Creating prototype repositories has let us understand how users access these repositories and how the system presents information to them. We have started work on new versions of both the editor and browser interfaces to address these issues.

Information processing and knowledge retrieval

Now that a formal specification for representation of artifact function information and a prototype artifact modeling system exist, the next step is to generate algorithms for information processing and knowledge retrieval. We have identified and devised a few algorithms for information processing and reasoning but have not yet implemented them.

For example, we have designed an algorithm to search an artifact representation to identify input and output flows for functions that reduce to subfunctions. For the mechanism exemplified in Figure 1, 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 are organized into more than one level and that have subfunctions with multiple input and output flows that may reference different source and destination artifacts.

Once we've modeled sufficient design artifacts, the set of models can serve as a database for knowledge retrieval and reuse. Using the above algorithm, a designer can search for particular function structures; a search for something that converts rotational motion to oscillatory translational motion, for instance, would find the pump mechanism. Because the schemata that has been developed also captures various kinds of properties, we can include them as search criteria. We can search not only for a function structure that converts electrical energy to rotary motion but for one whose input is electrical energy in the form of direct current at, say, no more than 12 volts.

Furthermore, this formal representation can support even more sophisticated queries. As illustrated in Figure 1, a user can map the artifact representation to a graph on which the various data structures (functions, flows, and artifacts) are nodes, and references among data structures are arcs.

Figure 4. Visualization of the power drill geometry.

We could use sophisticated graph-based pattern-matching algorithms for even more intelligent kinds of searches. A search could attempt not only to match a function with certain inputs and outputs but also to 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 in which 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 we describe provides a means for the creation of archives and repositories of corporate design knowledge that support more effective retrieval and knowledge reuse.

THE NIST DESIGN REPOSITORY PROJECT is developing an information-modeling framework to support the creation of design repositories. This project is driven by industry needs—needs for representation, capture, sharing, and reuse of corporate design knowledge.

Besides continuing the present aspects of the project and populating the design repositories with new design artifacts, several important areas of research remain to be addressed in future work.

The main aspects of the design modeling

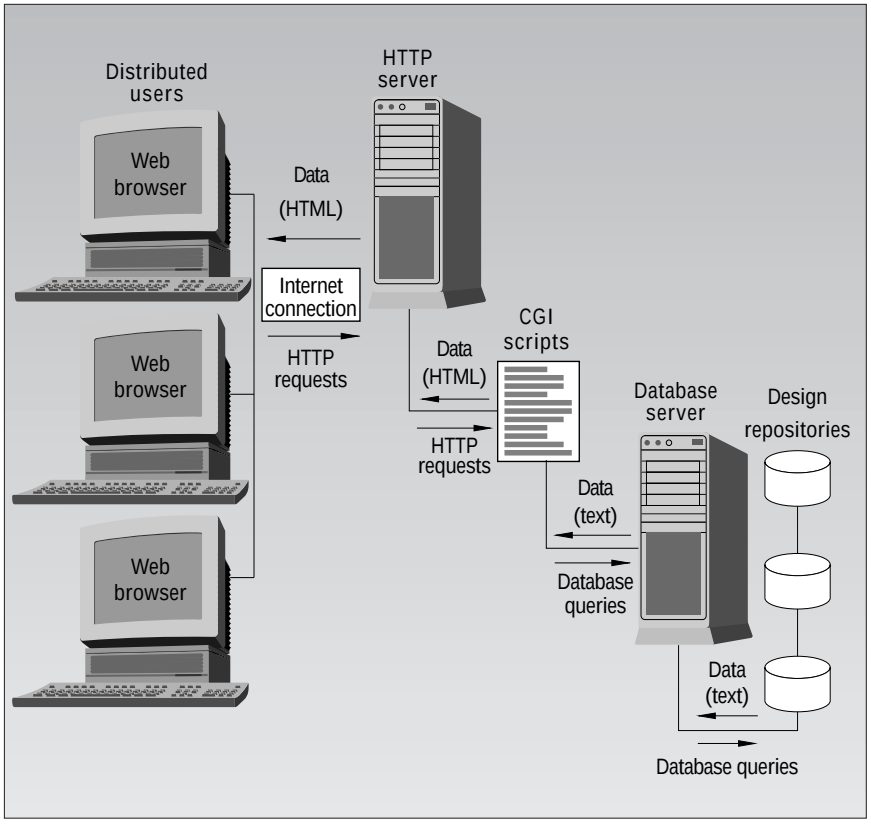

Figure 5. Architecture of design repository browser data interactions.

language are the representation of form, function, and behavior (as well as relationships between their associated objects). Although artifact representation currently captures information from the physical (form) and functional domains and mappings between them, representation of behavior within the existing framework is still relatively simple, descriptive, and text-based.

Another objective for future work is to incorporate a more formal representation of behavior to allow composable simulations. We will explore the possibility of incorporating an existing behavior-modeling language into the NIST Design Repository Project before we attempt to create a new one.

The Defense Advanced Research Projects Agency's Rapid Design Exploration and Optimization Program (for which NIST was a funding agent) has funded work in the area of behavior modeling through two projects: the Model-Based Support of Distributed Collaborative Design Project (previously How Things Work) at the Stanford University Knowledge Systems Laboratory[11] and the Active Catalog Project at the University of Southern California Information Sciences Institute.[12]

A need also exists 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. We will expand the current artifact representation to capture assembly-related information (such as assembly constraints and mating information).

In addition, design rationale representation is very important for 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 functions and mapping them 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. We have a new architecture 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 (that is, RAM) in a server-side Java application called a *servlet*. When navigating an artifact repository, the current implementation requires repeated time-consuming queries to a database back end. Second, the Web browser will process information for presentation purposes on the client side using JavaScript. This allows the sending of a larger amount of information to the client with each query, which very slightly increases the time associated with a single query but decreases overall lag time by substantially reducing the total number of server queries.

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 for the architecture currently under development.

Industry needs have motivated the NIST Design Repository Project. As such, we hope that industry will benefit from this technical foundation for the development of design repositories. Another potential use for this work is enhancement of the database content at the US Patent and Trademark Office. Currently, patent information consists of text and 2D images. Providing a formal artifact representation that extends to function and behavior not only leads to a more comprehensive description of a device but also provides additional types of information for meaningful indexing, which could lead to more efficient search and retrieval of patent information. ❐

Acknowledgments

Use of any commercial product or company names in this article 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.

References

1. S. Szykman, R.D. Sriram, and S.J. Smith, eds., *Proc. NIST Design Repository Workshop*, NISTIR 6159, National Institute of Standards and Technology, Gaithersburg, Md., 1998.

2. A. Goel et al., "Explanatory Interface in Interactive Design Environments," *Artificial Intelligence in Design '96*, J.S. Gero, ed., Kluwer Academic Publishers, Boston, 1996, pp. 387–404.

3. S.R. Gorti et al., "An Object-Oriented Representation for Product and Design Processes," *Computer-Aided Design*, Vol. 30, No. 7, June 1998, pp. 489–501.

4. L. Qian and J.S. Gero, "Function-Behavior-Structure Paths and Their Role in Analogy-Based Design," *Artificial Intelligence for Eng. Design, Analysis and Manufacturing*, Vol. 10, No. 4, Sept. 1996, pp. 289–312.

5. S. Szykman et al., "A Web-Based System for

Design Artifact Modeling," *Design Studies*, Vol. 21, No. 2, 2000, pp. 145–165.

6. Y. Umeda and T. Tomiyama, "Functional Reasoning in Design," *IEEE Expert*, Vol. 12, No. 2, Mar./Apr. 1997, pp. 42–48.
7. Y. Iwasaki and B. Chandrasekaran, "Design Verification through Function and Behavior-Oriented Representations: Bridging the Gap between Function and Behavior," *Artificial Intelligence in Design '92*, J.S. Gero, ed., Kluwer Academic Publishers, Boston, 1992, pp. 597–616.
8. J. de Kleer and J.S. Brown, "Assumptions and Ambiguities in Mechanistic Mental Models," *Mental Models*, D. Gentner and A.L. Stevens, eds., Lawrence Erlbaum Associates, Hillsdale, N.J., 1983, pp. 155–190.
9. S. Szykman, J. Senfaute, and R.D. Sriram, "Using XML to Describe Functions and Taxonomies in Computer-Based Design," *Proc. 1999 ASME Design Eng. Technical Conferences (Computers and Information in Eng. Conf.)*, Paper DETC99/CIE-9025, ASME, New York, 1999.
10. S. Szykman, J.W. Racz, and R.D. Sriram, "The Representation of Function in Computer-Based Design," *Proc. 1999 ASME Design Eng. Technical Conferences (International Conf. Design Theory and Methodology)*, Paper DETC99/DTM-8742, ASME, New York, 1999.
11. B.Y. Chouiery et al., "Thoughts on a Practical Reformation for Reasoning about Physical Systems," *Proc. Symp. Abstraction, Reformation, and Approximation '98*, AAAI, 1998, pp. 25–36.
12. M. Coutinhi et al., "Active Catalogs: Integrated Support for Component Engineering," *Proc. 1998 ASME Design Eng. Technical Conf.*, ASME, New York, 1998.

Simon Szykman is a mechanical engineer in the Engineering Design Technologies Group in the Manufacturing Systems Integration Division at the National Institute of Standards and Technology. He currently leads an effort in intelligent and distributed design that includes the NIST Design Repository Project. His research activities have included the use and improvement of stochastic optimization methods for product layout, the application of optimization to assembly design, Internet-based computer-aided design and manufacturing, and the development of tools and representations for the modeling of design artifact information. He received a BS in mechanical engineering at the University of Rochester and an MS and PhD in mechanical engineering from Carnegie Mellon University. Contact him at NIST, Manufacturing Systems Integration Division, Bldg. 304, Rm. 6, Gaithersburg, MD 20899; szykman@cme.nist.gov.

Ram D. Sriram currently leads the Engineering Design Technologies Group in the Manufacturing Systems Integration Division at NIST. He has authored or coauthored more than 100 publications in computer-aided engineering, including 12 books, and he was a founding coeditor of the *International Journal for AI in Engineering*. He has a BS from the Indian Institute of Technology in Madras, India, and an MS and PhD from Carnegie Mellon University. He has also received a Presidential Young Investigator's Award from the National Science Foundation. Contact him at NIST, Manufacturing Systems Integration Division, Bldg. 304, Rm. 6, Gaithersburg, MD 20899; sriram@cme.nist.gov.

Christophe Bochenek serves on the NIST Design Repository Project staff and develops Web-based tools for design artifact modeling. He received an MS in computer integrated manufacturing from Ecole Supérieure d'Informatique et Applications de Lorraine (ESIAL) in Nancy, France.

Janusz W. Racz is a member of the NIST Design Repository Project staff and deals with artifact representation development and the implementation of a Web-based prototype tool suite. He's also served as an assistant professor at the Institute of Fundamental Technological Research, Polish Academy of Sciences, where he received a PhD related to the use of AI in mobile robotics. He has authored nearly 30 papers on AI, robotics, civil engineering, and mechanics. Contact him at NIST, Manufacturing Systems Integration Division, Bldg. 304, Rm. 6, Gaithersburg, MD 20899; jwracz@cme.nist.gov.

Jocelyn Senfaute is a computer engineer who joined NIST as a guest researcher. He is currently a member of the NIST Design Repository Project staff and is working on system architecture and representation development. His main research interests are in XML and Web technologies, object-oriented development, and database systems. He received his MSc in computer science from Ecole Supérieure d'Informatique et Applications de Lorraine (ESIAL) in Nancy, France. Contact him at NIST, Manufacturing Systems Integration Division, Bldg. 304, Rm. 6, Gaithersburg, MD 20899; senfaute@cme.nist.gov.