Proceedings of the
1999 ASME Design Engineering Technical Conferences
September 12-15, 1999, Las Vegas, Nevada

DETC99/DTM-8742

THE REPRESENTATION OF FUNCTION IN COMPUTER-BASED DESIGN

Simon Szykman, Janusz W. Racz and Ram D. Sriram
Manufacturing Systems Integration Division
National Institute of Standards and Technology
Building 304, Room 6
100 Bureau Drive, Stop 8262
Gaithersburg, MD 20899-8262

Keywords: Design Repositories, Function, Information Modeling, Representation, Taxonomy

ABSTRACT

This paper proposes a standardized representation of function for use by the research community, industry, and eventually commercial software vendors. This includes schemata (information models) for representation of function and associated flows, as well as an initial attempt at developing taxonomies of functions and flows. The objective of the latter effort is to generate taxonomies that are as small as possible, yet generic enough to allow modeling of a broad variety of engineering artifacts. This representation is intended to provide a generic infrastructure that will facilitate the capture and exchange of function information among researchers at present, and eventually in industry by contributing to interoperability between design systems, be they commercial or developed internally within a company.

1 INTRODUCTION

Engineering function has been the subject of investigation in several research communities. A majority of the work in this area has been done in the artificial intelligence (AI) domain, where the representation of function has been studied in both the design modeling and the function-based reasoning communities. More recently, function has increasingly been the subject of research within the engineering domain, much of which has been conducted within the design theory and methodology community.

A review of the literature in area of engineering function shows that the bulk of the research falls into a small number of categories though individual research projects do, in some cases, span across more than one of these categories:

- Studies of the use of function by designers in the design process.
- The development of qualitative or rule-based models that attempt to codify knowledge about the use of function in the design process with the objective of using function-based reasoning as an aid to engineers and designers.
- The development of design artifact models that extend beyond traditional geometric representations to capture some representation of function.
- The development of quantitative models that map function to the physical domain, i.e., the mapping of function to physical or behavioral models associated with assemblies, subassemblies, and components within a design artifact, with the goal of reasoning about a design artifact or simulating its behavior.

Ultimately, in order for any of this work to have an impact on the engineering industry, it must be used by engineers. There are three ways in which knowledge can transition into industry. First, engineers can read papers and/or books that describe the use of function by designers and use that knowledge to improve their own design processes. This is the most natural transition for research in the first category listed above. Second, engineers in industry can develop computer-based design tools that incorporate the kinds of models described in the other three categories. With an emphasis on more effective use and reuse of knowledge, information about artifact function is increasingly being used in industry though typically this information is part of textual documentation and is not represented in any formal way. Third, CAD/CAM/CAE (computer-aided design/manufacturing/engineering) software tool vendors can begin to incorporate function representation into their commercial systems.

With either of the two latter technology transition paths, the use of representations and models of function in individual

systems (be they developed internally within a company or commercial) is only one step towards realizing an impact on product development. Equally important is the ability to share and exchange knowledge with other individuals, design teams, suppliers, corporate partners, etc., who in practice will often not be using the same software systems. Capture of information is only of limited use if the information cannot be effectively communicated to others. Indeed, a major barrier to information exchange in industry today is the proliferation of incompatible proprietary formats for representation of artifact geometry. An analogous problem could easily inhibit the effective use of function-based representations and models in product development.

To address this issue in a proactive, rather than reactive, manner, this paper proposes a standardized representation of function for use by the research community, industry, and eventually commercial software vendors. In the near term, the primary users of this representation will be in the research community as function representation and function-based reasoning have not yet transitioned into widespread use in industry. A standardized format for representation of functional information will provide a starting point for new researchers, and will also enable the exchange of information among researchers who are currently active in this area.

One option is for researchers to modify ongoing work to adopt this representation. Alternatively, for efforts that have significant investment in existing representations, those representations could be maintained while a standardized representation would serve as a neutral format for exchange of functional information through the use of a mapping to translate between an existing representation and a standardized one. This mode of use is similar to the use of STEP (ISO 10303, Standard for the Exchange of Product Model Data) (ISO, 1994a) for representation of geometry. STEP is not used as an internal representation for any major CAD system, but is supported via import and export capabilities by numerous CAD vendors, enabling large segments of industry to exchange geometric information between otherwise incompatible systems.

A standardized format for representation of functional information will facilitate access to existing bodies of work. This will allow researchers and software developers to leverage previous research without having to reinvent the proverbial wheel, and possibly to tie together existing efforts in order to provide systems with a greater level of capability and functionality. By serving as a method of bridging currently-isolated islands of research, this representation can serve to speed up progress in the research community. Most importantly, the early establishment of a standardized representation, if broadly adopted as a *de facto* standard or if formally incorporated as part of an international standard, will help to avoid the emergence of competing proprietary formats. In the realm of commercial geometric CAD systems, the problem of incompatible representations has cost engineering industry quite literally billions of dollars to deal with, not to mention the unquantifiable revenues that might have been realized had interoperability between CAD systems not been a barrier to productivity and collaboration.

With the problem of geometric CAD interoperability as a lesson, industry has repeatedly expressed a desire for standardization in anticipation of new capabilities within CAD systems, and not in reaction to these capabilities after their commercial implementation, as has been demonstrated by industry participation in over a half dozen design-related industry workshops held at the National Institute of Standards and Technology (NIST). One of these workshops, the NIST Design Repository Workshop, is particularly relevant to this research. Discussion of the needs associated with representation of engineering function arose in three different breakout sessions. Specific statements indicated (1) a need for representation of function in CAD, in addition to geometry, (2) a need for a fixed representation scheme for modeling function, and (3) a need for a commonly agreed-upon set of functions performed by mechanical systems (Szykman et al., 1998).

The work described in this paper addresses all of these needs. This paper sets forth an initial specification for a standardized representation of engineering artifact function. This includes schemata (information models) for representation of function and associated flows, as well as an initial attempt at developing taxonomies of functions and flows. The objective of the latter effort is to generate taxonomies that are as small as possible, yet generic enough to allow modeling of a broad variety of engineering artifacts. This representation is intended to provide a generic infrastructure that will facilitate the capture and exchange of function information among researchers at present, and eventually in industry by contributing to interoperability between design systems, be they commercial or developed internally within a company.

The next section provides an overview of related work done in this area. Section 3 describes the schemata for the data structures used to represent artifact function and associated flows. Section 4 discusses the development of taxonomies of generic functions and flows to be used in conjunction with the function representation. Section 5 provides a more general discussion of additional function representation issues. Areas for future research are discussed in Section 6.

2 RELATED WORK

The use of function has long-since been recognized as an important part of the design process. Formalization of approaches to representing and reasoning about function, and using this knowledge to drive design, are in comparison relatively new in the engineering field. Much of the early research in the area function representation was performed in the artificial intelligence (AI) field. Even definitions of function have varied, indicating that the concept is a complex one. Common definitions include any of a number of variations on one proposed by Rodenacker (1971), which defines function as a relation between the input and output of energy, material, and information. Pahl and Beitz (1988) retain this characterization but generalize the concept, defining function as an abstract formulation of a task, independent of any particular solution. In the AI field, definitions of function have often involved the concept of behavior (de Kleer, 1984; Chandrasekaran, 1994; Iwasaki et al., 1995; Kannapan, 1995; Qian and Gero, 1996, Umeda and Tomiyama, 1997; Prabhakar and Goel, 1998; and others).

Functional Representation (FR) takes a top-down approach to representing a device in the sense that the overall function is described first, and then the behavior of each component is de-

scribed in the context of this function (Chandrasekaran, 1994). As an extension of this approach Iwasaki et al. (1995, 1997) propose the Casual Functional Representation Language (CFRL) for representing device function with well-defined semantics in terms of behavior. This formalism allows the specification of conditions that a behavior must satisfy, such as occurrence of temporal sequences of events and casual relations among them. Sasajima et al. (1996) propose a method and a vocabulary for representing components captured from the viewpoints of behavior and function. Their Function and Behavior Representation Language (FBRL) does not rely on domain-specific terms and enables model builders to describe a component at various levels of abstraction. Other approaches to generic artifact modeling attempt to integrate representations of structure, behavior, and function. Models of this type have been developed by Goel et al. (1996), Qian and Gero (1996) Umeda and Tomiyama (1997), Gorti et al.(1998), Prabhakar and Goel (1998), and Szykman et al. (1999a).

The view taken in this paper is that function and behavior are two different facets of a design artifact, but are themselves unrelated. A generic function can be satisfied by more than one physical embodiment, each of which may achieve that function with different behaviors. Function may drive design, and therefore an artifact behavior may exist in order to satisfy required functions. Both are necessary to characterize a design artifact, but the concept of behavior is not necessary for the generic representation of engineering functions.

The variety in definitions of function have led to a variety of uses and representations of function. Baxter (1994) distinguishes two types of representations: models based on inputs and outputs of flows, and syntactic languages. The first type generally follows the Pahl and Beitz paradigm of the flow of materials, energy and signals through a hierarchy of functions. The overall function is determined for a system and then broken down into a set of subfunctions. This type of decomposition yields a functional graph that roughly approximates subassembly boundaries (Shapiro and Voelcker, 1989). Other approaches including (Kirschman and Fadel, 1998), (Umeda and Tomiyama, 1997), and the one taken in this paper, view the functional description of a system as being described by an abstract functional decomposition that may, but need not, have a direct mapping onto an isomorphic physical decomposition of assemblies and subassemblies.

Syntactic languages describe a design artifact using a grammatical approach where a grammar is used to capture information about function. In general, these grammars consist of combinations of verbs (functions) and nouns (parts of a design artifact, or flows) such as "hold liquid" (Lai and Wilson, 1989), "crush material" (Hundal, 1990), "create lateral motion" (Sturges et al., 1996), "transmit linear motion" (Kirschman and Fadel, 1998), and "convert electricity to thermal energy" (Stone and Wood, 1999). Approaches such as these can capture the essence of many artifact functions; however, they do not fully address the needs of a formal representation of function because they do not include information models that capture other types of information relevant to function, or the explicit mappings between functions to flows, and between the function domain and the physical domain. The representation of function described in this paper seeks to address these limitations by providing formal schemata and taxonomies of terms used to describe artifact functions and associated flows.

3 REPRESENTATION OF FUNCTION

This section describes generic representations for function and associated flows (e.g., material, energy, and signal flows). Within the context of this representation, function and flow are represented separately using different schemata, or information models. There are several motivations behind the decision to decouple the representation of function and flow. The first reason is that by separating the representations, changes can be made to one aspect of the representation without changing the other. For instance, if the function of an artifact is to *convert* one kind of energy (a flow) into another, the source, destination, or other properties and parameters of a flow can be modified without making any changes to the representation of the function itself.

The second reason for the separation of function and flow is to avoid a proliferation of concepts required for modeling artifacts. If flows were considered as a part of a function, six different functions would be required to represent the following: *convert direct current to translational motion*, *convert direct current to rotational motion*, *convert translational motion to direct current*, *convert translational motion to rotational motion*, *convert rotational motion to direct current*, *convert rotational motion to translational motion*. By considering functions and flows separately, only a single *convert* function, is needed; this function can have as inputs and outputs many different flows (including direct current, translational motion, rotational motion, as well as others). As another example, Collins et al. (1976) present a list of 105 functions taken from the context of helicopter failures. Of those, eleven have to do with transmitting (of force, torque, motion, etc.) and five have to do with limiting (of force, pressure, etc.). In contrast, as viewed in the context of the representation described in this research, *transmit* and *limit* are functions, and force, torque, motion, pressure are represented separately as flows.

The final reason for this separation is to facilitate the representation of functions that are not associated with any particular flows. While the focus of this paper and the examples given in this section are on representation of flow-based functions, the representation of functions that do not act on flows can be accomplished by defining functions that simply have no input or output flows. In contrast, were the representation of flows to be included as part of the function information model, to represent non-flow-based functions either a separate schema would have to be developed or the flow portion of the function representation would have to be carried around, unused, creating unnecessary baggage.

3.1 The Function Schema

The 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. The `Name` of the function is a string, and is required to be unique. The `Type` is a reference

```
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. Schema for Representation of Function

to a generic function class that is part of a function class taxonomy (to be discussed in Section 4). `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. Neither `Documentation` nor `Methods` are required and either field may be left empty (i.e., `NULL`).

The next two items in the schema are `Input_flow` and `Output_flow`. These are references to data structures representing the input and output flows for the function. As described above, braces indicate a list of references; thus the function schema can represent functions having multiple inputs and outputs. Since not every function has both input and output flows, these can also take the value of `NULL`. To illustrate, a motor has a transformation function "Convert" that transforms an electrical energy flow into a rotational motion flow, and therefore has both input and output flows. In contrast, the function of a battery may be to supply electrical energy, in which case the battery would have an output flow without having an input flow.

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. These flows may in turn each be associated with different physical parts of a design artifact. In cases where it is not necessary to further decompose a function into subfunctions, this item would be left empty or `NULL`. This use of `Subfunctions` to decompose a function would, for example, be used to model the functionality of an assembly. At one level an assembly is in individual entity that has a function, while at another level it consists of multiple subassemblies or components each of which contribute to the overall functionality of the assembly. The decomposition enabled by `Subfunctions` provides the means to map complex functionality to more detailed portions of an artifact model. This idea will be illustrated in an example later in this section.

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`. A function will only have a `Subfunction_of` if it is a subfunction of another function; otherwise, `Subfunction_of` will have a value of `NULL`. The last item in the function schema is `Referring_artifact`. This is a reference from a function back to the artifact that references it.

In reality, `Subfunction_of` and `Referring_artifact` are both redundant information. As will be illustrated in an example below, one can view references among data structures as forming a graph with links from an artifact to its functions, and from a function to its flows. The use of these two references effectively make these links bi-directional. These items are present for convenience at the software development level. Some developers of a design artifact modeling system may find it convenient to use this link to get from a function to the artifact that references that function, while others could find that same artifact by searching all the artifact data structures for the one that references the function of interest. Put another way, a graph can be traversed or searched regardless of whether links are uni-directional or bi-directional. Both approaches have pros and cons, depending on how the artifact modeling system is implemented, and on how data is stored and retrieved. Although a discussion of these implementational issues is beyond the scope of this paper, these references have been included in the function schema to provide flexibility for multiple implementation approaches even though, strictly speaking, only links in one direction are truly necessary.

3.2 The Flow Schema

Figure 2 shows the schema for the flow information model. Like the function schema, the flow schema has a `Name` that is required to be unique, a `Type` (that references the generic flow class to which a given flow belongs, taken from a flow taxonomy to be discussed in Section 4), and a `Documentation` string. In addition to these items, the flow schema also has a `Source` and a `Destination`, which reference the physical artifacts corresponding to the sources and destinations of the flows for a given function. These two items are shown with braces, indicating a list of references, to allow the representation of a flow having multiple sources or destinations. This would not be uncommon in electromechanical devices where a device might have a power input from batteries arranged in parallel (multiple flow sources), or where the output flow from a power source might drive several devices (multiple flow destinations). Either one (but not both) of these can also take the value of `NULL`.[1]

There are a great many properties or parameters that can be associated with flows, many of which depend on the kind of flow, the nature of design artifact under consideration, the

[1] In reality, a flow can neither appear from nowhere, nor vanish into nothingness. However, in most cases, a design artifact will have one or more flows having a source or destination that is not part of the artifact representation itself. In cases where the source or destination resides outside of the modeled artifact, the source and destination in the flow representation would have NULL values. The modeling of flows that cross these boundaries will be discussed further in Section 4.

```
Flow
    Name                string
    Type                [Generic_flow_class]
    Documentation       string       (or NULL)
    Source              {[Artifact]} (or NULL)
    Destination         {[Artifact]} (or NULL)
    Properties          {string}     (or NULL)
    Referring_functions {[Function]}
```

Figure 2. Schema for Representation of Flow

domain of interest, etc. Rather than attempt to itemize *a priori* all the potential properties that could be relevant to a given flow and capture them in a monolithic data structure, the `Properties` item is provided as part of the flow schema. The `Properties` item is a list of strings that specify flow properties or parameters, and can vary based on the user's requirements. For example, an electrical flow may have a property that specifies its voltage in a string such as "v = 5 Volts". Other kinds of properties could also be included, such as range limits, tolerance information, etc. While various types of information can be represented in this manner, the schema itself does not include a description of the meaning, or semantics, associated with `Properties`; they simply appear as a list of strings. Thus, an electrical flow that varies sinusoidally with time might be specified with a string such as "v = 3 sin (t/2)", but parsing that string to elicit its meaning in order to use that information for function-based reasoning becomes a requirement at the implementation level.

The last item in the schema is `Referring_functions`, which is a list of references to functions that reference that flow. As with the `Referring_artifact` reference in the function schema, this information is redundant but is provided, nevertheless, for implementational convenience.

3.3 Examples

The relationship among the representations of the various information types for the motor described above is illustrated graphically in Figure 3, with boxes representing data structures and arrows representing references from one data structure to another. The functions and flows are labeled with the `Name` at the top and the `Type` (generic function or flow class) in parentheses below. Figure 3 shows the mappings between function/flow/artifact for the motor. As the figure illustrates, the motor function is to convert electrical energy that goes from a wire to the motor, into rotational motion that goes from the motor to a shaft. Note that the directions of the arrows represent the references in the data structures presented above and not necessarily the direction of a flow. Although the general direction of flow in this example is from left to right in the figure (wire to motor to shaft), the input flow arrow for the motor function points out from the motor function—not into it—because the function data structure references the flow.

Clearly, any design artifact modeling tool that uses this representation of function would require some kind of artifact representation as well. A comprehensive discussion of artifact representation is outside the scope of this paper, as such a representation would include not only representation of function, but also form, behavior, relationships among artifacts, etc. An artifact schema would also include (1) a method of representing the physical decomposition of an artifact into assemblies, subassemblies and components (much as the function schema allows the decomposition of a function into subfunctions), and (2) a method of mapping the physical domain to the function domain (just as the function and flow schemata map the func-

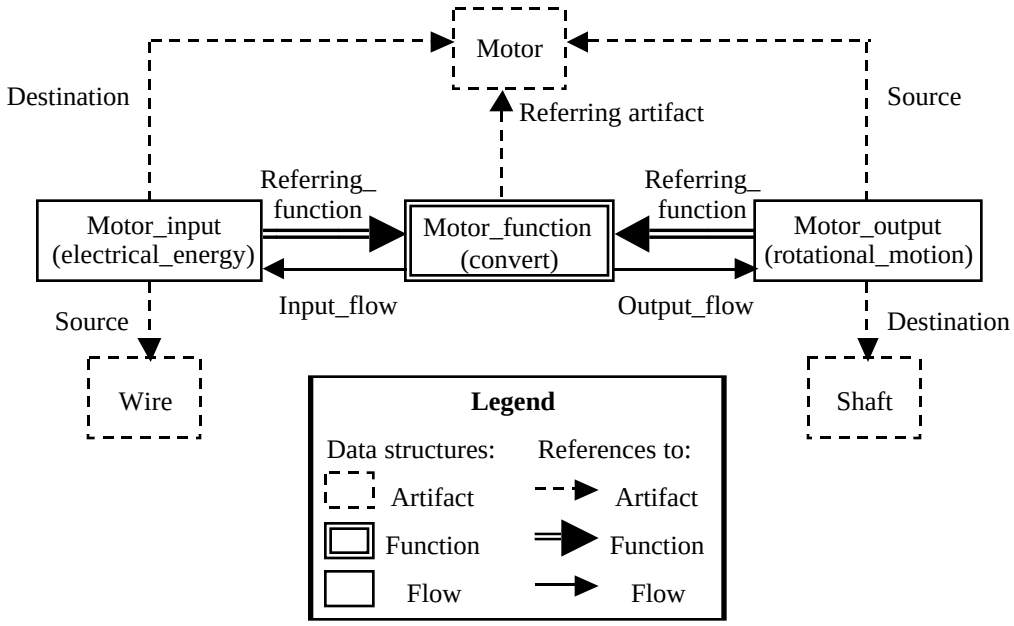


Figure 3. Graphical Illustration of Motor Function Representation

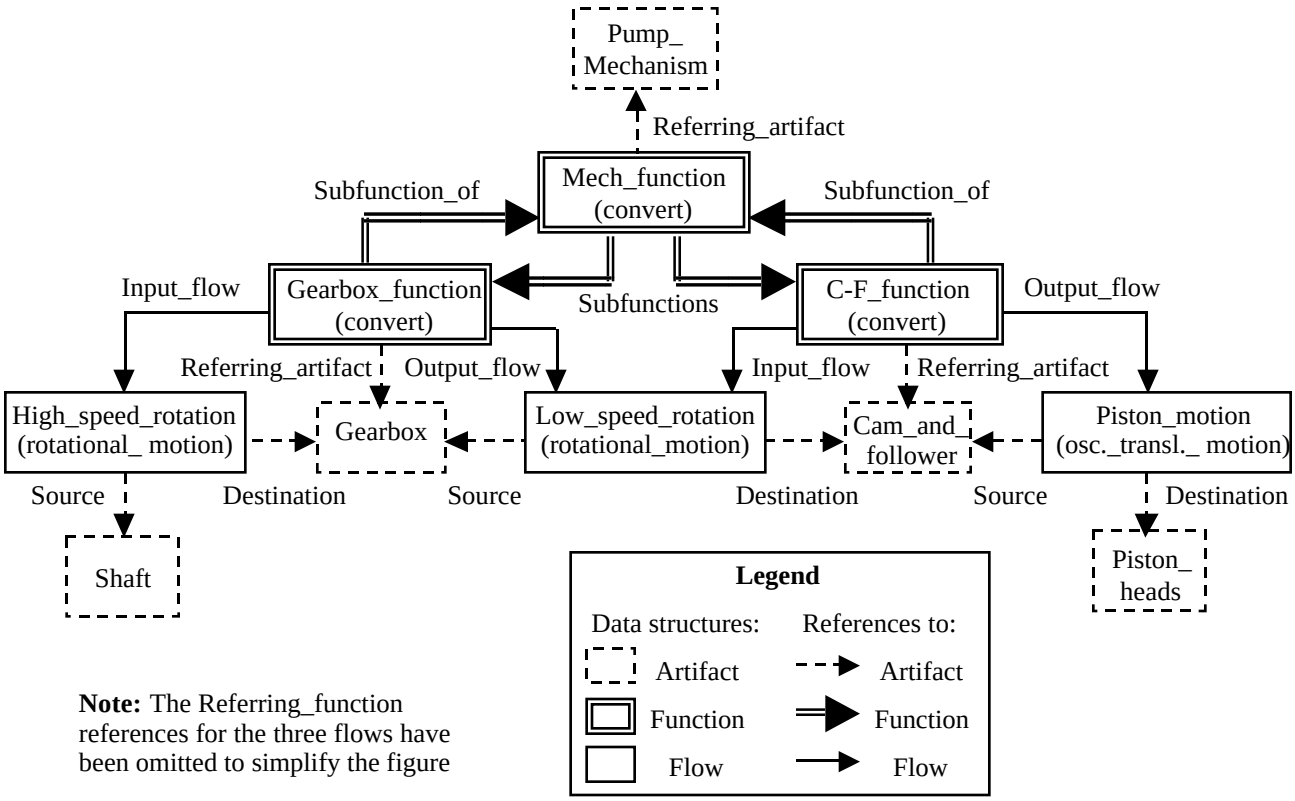


Figure 4. Graphical Illustration of Function Representation for Fluid Pump Mechanism

tion domain to the physical domain). Thus, although this paper is not concerned with representation of the artifact itself, artifacts are shown in the figure to illustrate this mapping between domains. Since no schema for the artifact itself is given, arrows indicating references from artifacts to functions are not explicitly shown in the figures.

A second example, the mechanism from a fluid pump design, can be used to demonstrate the use of `Subfunctions` to decompose a function into multiple subfunctions. Within this pump is a mechanism that takes 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. The function of this mechanism could therefore be represented graphically using a structure that looks the same as that shown in Figure 3 but with different labels in the boxes since the function/flows/artifacts are different.

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 an oscillatory translational motion. These multiple functions can be represented individually and mapped appropriately back to the physical artifact domain. The graphical representation of the mechanism function is shown in Figure 4.[2]

It should be noted that the decomposed subfunctions are represented in conjunction with, not in place of, the composite function, allowing artifacts and their function to be represented at multiple levels of abstraction simultaneously. In this example, the mechanism can be interpreted at one level as a single artifact with a single function (to convert high-speed rotation to oscillatory translational motion), and at another level as a set of subfunctions which are mapped to different physical components of the mechanism. Since the schemata are invariant and do not depend on the level of abstraction, the function and physical decompositions, as well as the mappings between the function and physical domains, can extend upwards and downwards numerous levels. Just as the mechanism is decomposed as shown in Figure 4, the pump itself may at one level be represented as a single artifact, and then be physically and functionally decomposed into multiple components, subassemblies, and subfunctions, which would include the mechanism as well as other parts of the pump.

[2] As before, what is not shown in the figure is the actual artifact representation that would capture the fact that this mechanism is a subassembly within the pump, and that the gearbox, cam and follower are parts of the mechanism assembly.

4 GENERIC TAXONOMIES OF FUNCTION AND FLOW

The function and flow schemata described in Section 3 both include a Type, used to reference the generic class to which that function or flow belongs. In addition to the development of these schemata, a second objective of this work is the development of generic taxonomies of function and flow which are concise, yet comprehensive enough to allow the modeling of a broad variety of engineering artifacts.

In this context, a taxonomy is a hierarchical classification of terms. The organization of the flow taxonomy follows a traditional approach set forth by Pahl and Beitz (1988) whereby flows are divided into material, energy and signal flows.[3] It is important to note that the categorizations used in the taxonomies are not unique, but are rather a matter of convenience. The organization of the taxonomy is a particular instance of a view of the terminology it contains. For example, the flow taxonomy is broken down by domain (mechanical, electrical, thermal, etc.), each of which have various terms below them. However, an alternative categorization could have organized them by the mapping of variable types across domains.[4] The importance should be placed on the content of the taxonomy rather than the specific approach to organizing the terms.

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, or 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 function-based reasoning or design automation. 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, in only one unique way, because it is the flexibility required for representation of a broad set of concepts that 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 function models. This will facilitate the exchange of function information among distributed researchers and developers, and will greatly simplify the task of indexing and retrieval of information for the purposes of function-based searches and query capabilities.

An extensive review of the literature yielded a large body of function- and flow-based terminology within the context of engineering function (most notably Collins et al., 1976; Altshuller, 1984; Hundal, 1990; Keuneke, 1991; Chandrasekaran, 1994; Malmqvist et al., 1996; Sasajima et al., 1996; McAdams et al., 1998; Kirschman and Fadel, 1998; Modarres, 1998). While these terms were in some cases split into a few categories, they generally were not organized into multi-level taxonomies. From these bodies of terminology, an extensive list of functions and related flows was extracted. The lists of functions and flows were then distilled into considerably smaller ones in the following ways:

By removing synonyms (e.g., “change” and “modify”).

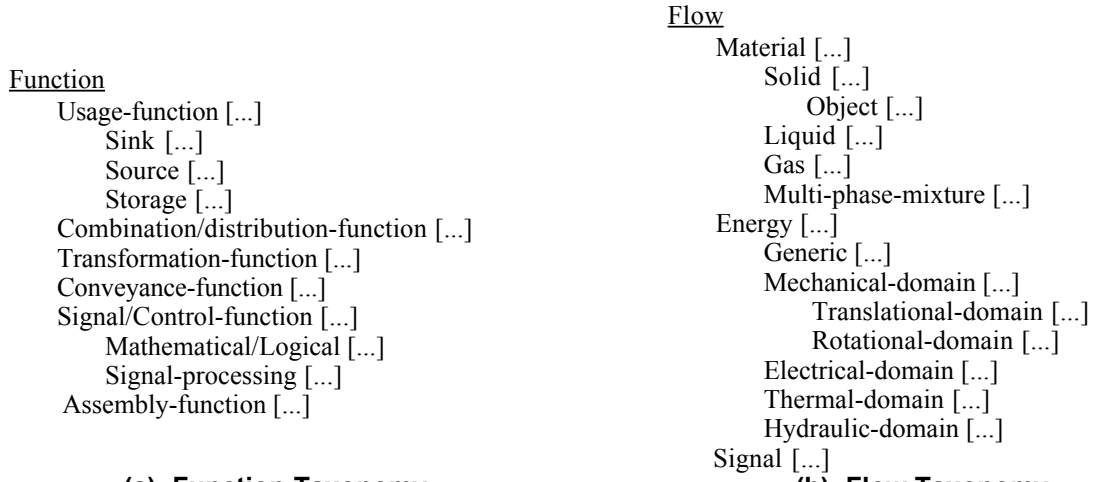
- By eliminating functions that were specializations of more generic functions. For example, boil, melt, freeze, condense, evaporate, liquefy, atomize, are all examples of a more generic convert function that transforms matter from one state into another.
- By eliminating flows that were specializations of more generic types of flows. For instance, the concept of smoke does not appear in the list of flows because smoke is an example of an aerosol, a more generic term that does appear in the list.

The lists of functions and flows were then categorized hierarchically and organized into taxonomies. The top-level divisions of the two taxonomies are shown in Figure 5. 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 Appendices A and B, respectively. 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.

While the content of the two taxonomies is self-explanatory for the most part, there are a couple of items that merit further discussion. The first is the use of the *import* and *export* functions that appear in the *usage-function* portion of the taxonomy. As described in Section 3, in some cases a flow may cross the boundary between the artifact being modeled and the external world. For example, a machine may use electricity whose source is an electrical wall outlet, but the outlet is not actually part of the artifact representation. The crossing of the artifact boundary by such flows is modeled using the *import* and *export* functions. Flows referenced by import and export functions have NULL values for source and destination, respe-

[3] These terms are used somewhat loosely. Many of the terms under the energy flow category relate to the transfer of energy, but do not actually have units of energy. Others such as position and angle, strictly speaking do not necessarily involve transfer of energy, though *changes* in position and angle (i.e., displacement and rotation) can.

[4] In the systems engineering field, there are mappings between certain types of variables. For example, position in the mechanical translational domain is mapped to angle in the rotational domain, to charge in the electrical domain. Force is mapped to torque, and to voltage. Velocity (time rate of change of position) is mapped to angular velocity (time rate of change of angle), and to current (time rate of change of charge), and so on. In most cases, these mappings extend into other domains as well (thermal, hydraulic, etc.).



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

Figure 5. Top-level Subdivisions for the Function and Flow Taxonomies

tively, since the source or destination resides outside of the modeled artifact. In the machine example the function of the power cord, which is part of the artifact representation, is to import electrical energy; the representation of the electrical energy flow would have a NULL value for its source, and some part of the machine as its destination.

The other issue of note is the presence of what appear to be flow parameters or properties in the flow taxonomy. In the approach presented in this paper, flows are inputs and outputs to functions. These flows may have properties associated with them, as can be seen from the flow schema shown in Figure 2, but properties of flows are not themselves flows. One might, for instance, question the term *temperature* in the flow taxonomy. Devices may have functions (and associated flows) such as supply (function) heat (flow), distribute heat, transmit heat, while functions such as supply/distribute/transmit temperature do not seem sensible. Rather, temperature might instead be a property of a fluid flow used to supply heat.

From this perspective, the term temperature would not belong in the flow taxonomy. However, within the scope of control functions, temperature may indeed be treated as a flow by virtue of being an input or an output of a function. Valid control functions might be to decrease (function) temperature (flow), measure temperature, limit temperature, etc. What is a property of a flow in some contexts, may be viewed as a flow in others. The flow taxonomy contains many flows, each of which potentially has a broad set of different parameters that could conceivably be used as flows for control purposes. Thus, some common examples of these kinds of terms have been included in the flow taxonomy but no attempt has been made to identify an exhaustive list.

5 DISCUSSION

The specification for function representation presented in this paper provides a simple, generic language for representing function information. The intent is not to provide a system for function modeling, but to provide a framework onto which such systems can be built. Consequently, the contribution of this work is solely at the representational level. This work does not attempt to specify *how* such information should be used. How information about function should be used in the design process, how reasoning about function should be done given this information (via constraints, rule-based expert systems, automated or not, etc.) are all beyond the scope of this work.

Representation of function is only one facet of engineering artifact representation; another important one is behavior. The work presented in this paper is complementary to ongoing work in the area of behavior modeling, as a comprehensive artifact model would account for both an artifact's function and its behavior. What a behavior model could add to a generic function representation is a formal description of how a given artifact physically accomplishes a function, as well as constraints or limitations on such behavior.

As an example, the Casual Functional Representation Language (CFRL) mentioned briefly in the related work section (Iwasaki et al., 1995, 1997) provides semantics that relate function to behavior. This allows the specification of conditions that a behavior must satisfy, such as occurrence of temporal sequences of events and casual relations among them. What the research presented in this paper could contribute to that work is a formal language and generic terminology for the representation of function and for the exchange of function information with others—something not presently addressed in CFRL and related work in behavior modeling.

Thus far, this paper has focused on the formal aspects of representing information related to engineering artifact function—the issues of information modeling (schemata) and terminology (taxonomies). There are several higher-level issues that have influenced this work but which have not been discussed in previous sections. Section 3 discussed the representation of functions that may have multiple input and output flows, as well as the decomposition of a single artifact function into multiple subfunctions. Neither of these is the same as function

sharing, a common occurrence (and in some cases even a design strategy) in engineering design.

While subfunctions decompose a single artifact function into parts that may be mapped onto different subassemblies or components of that artifact, this is distinctly different from function sharing wherein a system/assembly/component accomplishes multiple distinct functions. This work can support the representation of multiple functions, but does not explicitly account for this concept because the fact that a single artifact performs more than one function is something that must be represented at the artifact representation level, which is not addressed in this paper. One potential approach is to develop an artifact representation that includes a list of functions, just as the function schema includes a list of flows. Once the multiplicity of functions is captured at the artifact level, however this is achieved, each of these multiple functions can modeled using the representation presented in this paper.

Other issues arise at the modeling level when attempting to determine which functions and flows should and shouldn't be modeled. Some functions and flows are associated with an artifact's intended use, while others are effects produced as a consequence of the particular physical embodiment. At the modeling level, any functions and flows that are part of the design by intent should be represented. In contrast, those that arise as effects rather than by intent may or may not be represented, depending on whether or not any part of the artifact exists to interact with them.

Consider a motor whose function is to convert electrical energy into rotational motion. As with most mechanical devices, this motor makes noise, in this case because the components have an effect of converting mechanical energy into acoustical energy. In general, the acoustical energy flow does not need to be modeled as part of the artifact function. However, if the motor includes some form of sound insulation to absorb the sound, then that flow does need to be modeled. Even though the generation of sound is not part of the intended function of the motor, part of the physical design exists to interact with that flow. Since the sound insulation exists to absorb (function) acoustical energy (flow), the acoustical energy should be included when doing the function modeling. Similarly, some amount of heat is almost always generated as an effect of transforming energy into motion or vice-versa. If part of an artifact exists to carry away that heat, it should be represented; otherwise, modeling that flow may not be necessary.

As a related comment, it should be noted that it is often difficult to completely decouple different types of flows (Pahl and Beitz, 1988, Ullman, 1992). It takes energy to create or change motion; motion of matter involves kinetic energy; and in many cases a signal used for control is an electrical signal, which is actually an energy flow. In general, the modeling should be done at whichever level is most convenient for the designer. A designer will typically think of a gearbox as something that modifies motion, and think of a control device as having an input signal. Physically, both the motion and the signal exist as a result of energy flows, but conceptually it is more comfortable to think of motion and a control signal as being distinct types of flows.

6 SUMMARY AND AREAS FOR FUTURE RESEARCH

This paper proposes a standardized representation of function consisting of schemata for functions and associated flows, along with taxonomies of generic functions and flows. The formal representation provides the means for representing functions that have multiple input and output flows, properties and parameters associated with flows, and the decomposition of functions into subfunctions each potentially having its own distinct flows. The representation also provides a mapping from the function domain to the physical domain (via references to artifacts in the flow schema) and supports the representation of function sharing provided that it is used with an artifact representation that permits artifacts to have multiple functions.

Future work relating to the research presented in this paper is continuing along several avenues. To date, the focus of the development of the function taxonomy has focused on classes of functions that have flows associated with them. These are, however, not the only types of functions. Another important class of functions are assembly functions, such as *locate*, *constrain*, and *fasten*. A list of about twenty assembly-related functions has been included as part of the function taxonomy. This list is only representative, as many such functions exist. Work relating specifically to representation of assembly information and function has been done by various researchers, including Gui and Mäntylä (1994), Baxter et al. (1994), Brady and Juster (1995), Roy and Bharadwaj (1998), Shooter et al. (1999), and others. Interactions with the latter group, a team of researchers also at NIST, will be used to expand the scope of the taxonomy into the area of assembly functions. In addition to assembly functions, there are still other non-flow-based functions relating to more abstract types of issues such as to *shelter*, to *provide safety*, or to *provide access*. While the taxonomy does not currently extend to cover these other functions, the function representation itself is still capable of characterizing these concepts simply by using the function schema with the input and output flows having `NULL` values.

The schemata for function and flow presented in Section 3 are generic data structures, independent of implementation. A computer-based implementation of a function modeling tool that incorporates this representation is currently in progress. In this implementation, the schemata are represented using the Extensible Markup Language (XML) (World Wide Web Consortium, 1998), a language similar in appearance to the Hypertext Markup Language (HTML) (World Wide Web Consortium, 1995) but which allows the development of user-defined tags, various kinds of references, and other mechanisms.

XML is not the only language that can be utilized for information modeling and knowledge exchange; other such languages include EXPRESS (ISO, 1994b), Knowledge Interchange Format (KIF) (Genesereth and Fikes, 1992), and Open Knowledge Base Connectivity (OKBC) (Chahudri et al., 1998). However, XML has the main advantage of 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.

This research direction is being explored in order to provide a more broad-based solution to an industry which is increasingly looking towards purchase of off-the-shelf software over in-house development when possible. Preliminary development of mappings of the schemata presented in this paper into XML is described in (Szykman et al., 1999b).

The implementation of these schemata will provide an example of how a modeling tool may be built using this work as a representational layer. While this development will provide a useful learning experience, for the work to be more broadly applicable the function and flow taxonomies need to be expanded. The function taxonomy has currently focused on functions that have input and output flows associated with them. As described previously, these are not the only types of functions used by designers. The taxonomy lists several representative assembly functions, but should be extended to include additional assembly functions and other types of non-flow-based functions. Within the flow taxonomy there will also be an attempt to obtain more comprehensive coverage in the domains that are currently included, as well as to fill out the domains that presently only have a top-level "placeholder" or are not present at all. The development of both these taxonomies will be an ongoing process performed in conjunction with other technical development.

A limitation of this work is that, alone, it is of relatively limited utility. As engineering function is generally not of interest in the absence of design artifact information, the representation of function proposed in this paper is intended to be incorporated as a layer within the context of a larger artifact modeling system that includes a more comprehensive representation of a design artifact. Other important aspects of design knowledge include geometry, behavior, physical (often hierarchical) decompositions, and other kinds of relationships. Combining all of these kinds of knowledge into a comprehensive representation, and building a useful system from it is an area of longer-term research.

One such prototype system is being developed as part of the NIST Design Repository Project (Szykman et al., 1999a; Szykman et al., 1999c). The implementation of a new architecture for this system is in progress; this second-generation prototype will incorporate the schemata and taxonomies presented in this paper, as well as interfaces for modeling design artifacts and browsing repositories of artifact information. Part of this effort will include the modeling of various consumer products. This exercise will serve to validate the function and flow taxonomies by helping to identify gaps between the kinds of functions and flows that appear in real engineering systems and those that have been captured in the taxonomies. Important issues such as that of mechanisms for knowledge indexing and retrieval have not been addressed to a significant extent, though it is expected that the taxonomies that have been developed as part of the work presented in this paper will have an impact on future development.

Beyond its adoption within the NIST Design Repository Project, it is hoped that the specification for a standardized representation of function presented in this paper will propagate into other parts of the academic and industrial research communities. In the near term, such a standardization will facilitate information exchange among researchers. In the longer term, this work is intended to provide a foundation for developers of the next generation of CAD systems and design tools. Aside from supporting ongoing research at NIST, helping to avert the undesirable emergence of multiple competing, proprietary formats—a problem that has adversely impacted industry in the area of geometric modeling and representation—has been a strong motivation for undertaking this work.

REFERENCES

Altshuller, G. S. (1984), *Creativity as an Exact Science: The Theory of the Solution of Inventive Problems*, Gordon and Breach Science, New York.

Baxter, J. E., N. P. Juster, and A. de Pennington (1994), "A Functional Data Model For Assemblies Used To Verify Product Design Specifications," *Proceedings of the IMechE, Part B, Journal of Engineering Manufacture*, **208**:235-244.

Bracewell, R.H. and J.E.E. Sharpe (1996), "Functional Description Used in Computer Support for Qualitative Scheme Generation - Schemebuilder," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **10**(4):333-346.

Brady, D. and N. P. Juster (1995), "A Computerised Tool to Create Concept Variants from Function Structures," AI System Support for Conceptual Design, *Proceedings of the 1995 Lancaster International Workshop on Engineering Design*, Sharpe, J. (Ed.), Springer-Verlag, pp. 206-226.

Chaudhri, V. K., A. Farquhar, R. Fikes, P. D. Karp, and J. P. Rice (1998), *Open Knowledge Base Connectivity 2.0*, KSL-98-06 (technical report), Stanford Knowledge Systems Laboratory, Stanford University, Stanford, CA.

Chandrasekaran B. (1994), "Functional Representation: A Brief Historical Perspective," *Applied Artificial Intelligence*, **8**:163-197.

Collins, J. A., B. T. Hagan, and H. M. Bratt (1976), "The Failure-Experience Matrix—A Useful Design Tool," *Transactions of the ASME Series B, Journal of Engineering in Industry*, **98**(3):1074-1079.

de Kleer, J. (1984), "How Circuits Work," Artificial Intelligence, **24**:205-280.

Genesereth, M. R. and R. E. Fikes (1992), *Knowledge Interchange Format, Version 3.0 Reference Manual*, KSL-92-86 (technical report), Stanford Knowledge Systems Laboratory, Stanford University, Stanford, CA.

Goel, A., A. Gomez, N. Grue, J. W. Murdock, M. Recker and T. Govindaraj (1996), "Explanatory Interface in Interactive Design Environments," *Artificial Intelligence in Design '96*, J. S. Gero (ed.), Kluwer Academic Publishers, Boston.

Gorti, S. R., A. Gupta, G. J. Kim, R. D. Sriram, and A. Wong (1998), "An Object-Oriented Representation for Product and Design Processes," *Computer-Aided Design*, **30**(7):489-501.

Gui, J.-K. and M. Mäntylä (1994), "Functional Understanding of Assembly Modelling," *Computer-Aided Design*, **26**(6):435-451.

Hundal, M. S. (1990), "A Systematic Method for Developing Function Structures, Solutions and Concept Variants," *Mechanism and Machine Theory*, **25**(3):243-256.

ISO 10303-1:1994 (1994a), *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 1: Overview and Fundamental Principles*.

ISO 10303-11:1994 (1994b), *Industrial Automation Systems and Integration – Product Data Representation and Exchange – Part 11: The EXPRESS Language Reference Manual*.

Iwasaki, Y., A. Farquhar, R. Fikes, and J. Rice (1997), "A Web-Based Compositional Modeling System for Sharing of Physical Knowledge," *Proceedings of the 15th International Conference on Artificial Intelligence*, AAAI Press/MIT Press, August.

Iwasaki, Y., M. Vescovi, R. Fikes, and B. Chandrasekaran (1995), "Casual Functional Representation Language with Behavior-Based Semantics," *Applied Artificial Intelligence*, **9**:5-31.

Kannapan, S. M. (1995), "Function Metrics for Engineered Devices," *Applied Artificial Intelligence*, **9**:45-64.

Keuneke, A. (1991), "Device Representation: The Significance of Functional Knowledge," *IEEE Expert*, **6**(2):22-25.

Kirschman C. F. and G. M. Fadel (1998), "Classifying Functions for Mechanical Design," *ASME Journal of Mechanical Design*, **120**(3):475-482.

Lai, K., and W. R. D. Wilson (1989), "FDL - A Language for Function Description and Rationalization in Mechanical Design," *Journal of Mechanics, Transmissions, and Automation in Design*, **111**:117-123.

Lind, M. (1994) "Modeling Goals and Functions of Complex Industrial Plants," *Applied Artificial Intelligence*, **8**, pp. 259-284.

Malmqvist, J., R. Axelsson, and M. Johansson (1996), "A Comparative Analysis of the Theory of Inventive Problem Solving and the Systematic Approach of Pahl and Beitz," *Proceedings of the 1996 ASME Design Engineering Technical Conferences and Computers in Engineering Conference*, Paper No. 96-DETC/DTM-1529, Irvine, CA, August.

McAdams, D. A., R. B. Stone, and K. L. Wood (1998), "Understanding Product Similarity Using Customer Needs," *Proceedings of the 1998 ASME Design Engineering Technical Conferences*, Paper No. DETC98/DTM-5660, Atlanta, GA, September.

Modarres, M. (1998), "Functional Modeling of Physical Systems Using the Goal Tree Framework," *AAAI-98 Workshop on Functional Modeling and Teleological Reasoning*, Madison, WI, July.

Pahl, G. and W. Beitz (1988), *Engineering Design: A Systematic Approach*, Springer-Verlag, New York.

Prabhakar, S. and A. K. Goel, (1998), "Functional Modeling for Enabling Adaptive Design of Devices for New Environments," *Artificial Intelligence in Engineering*, **12**:417-444.

Qian L. and J. S. Gero (1996), "Function-Behavior-Structure Paths and Their Role in Analogy-Based Design," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **10**(4):289-312.

Rodenacker, W. (1971) *Methodishes Konstruieren*, Springer, Berlin.

Roy, U. and B. Bharadwaj (1999), "Design with Part Behaviors: Behavior Model, Representation and Applications," Submitted to *Computer-Aided Design*.

Sasajima, M., Y. Kitamura, M. Ikeda, and M. Mizoguchi (1996), "Representation Language for Behavior and Function: FBRL," *Expert Systems With Applications*, **10**(3/4):471-479.

Shapiro, V. and H. Voelcker (1989), "On the Role of Geometry in Mechanical Design," *Research in Engineering Design*, **1**:69-73.

Shooter, S., W. Keirouz, P. Hart, and K. Lyons (1999), "The Open Assembly Design Environment Project: An Architecture for Design Agent Interoperability," *Proceedings of the 1999 ASME Design Engineering Technical Conferences (4th Design for Manufacturing Conference)*, Paper No. DETC99/DFM-8945, Las Vegas, NV, September.

Stone, R. B. and K. L. Wood (1999), "Development of a Functional Basis for Design," *Proceedings of the 1999 ASME Design Engineering Technical Conferences (11th International Conference on Design Theory and Methodology)*, Paper No. DETC99/DTM-8765, Las Vegas, NV, September.

Sturges, R. H., K. O'Shaughnessy and M. I. Kilani (1996), "Computational Model for Conceptual Design Based on Extended Function Logic," *Artificial Intelligence for Engineering Design, Analysis and Manufacturing*, **10**:255-274.

Szykman, S., J. W. Racz, C. Bochenek and R. D. Sriram (1999a), "A Web-based System for Design Artifact Modeling," *Design Studies* (accepted for publication).

Szykman, S., J. W. Racz, and R. D. Sriram (1999b), "The Use of XML for Representing Functions and Taxonomies in Computer-based Design," *Proceedings of the 1999 ASME Design Engineering Technical Conferences (19th Computers and Information in Engineering Conference)*, Paper No. DETC99/CIE-9025, Las Vegas, NV, September.

Szykman, S., R. D. Sriram, C. Bochenek and J. W. Racz (1999c), "The NIST Design Repository Project," *Advances in Soft Computing – Engineering Design and Manufacturing*, Roy, R., T. Furuhashi, and P. K. Chawdhry (Eds.), Springer-Verlag, London, pp 5-19.

Szykman, S., R. D. Sriram and S. J. Smith (Eds.) (1998), *Proceedings of the NIST Design Repository Workshop*, Gaithersburg, MD, November 1996.

Ullman, D. G. (1992), *The Mechanical Design Process*, McGraw-Hill, New York, NY.

Umeda, Y. and T. Tomiyama (1997), "Functional Reasoning in Design," *IEEE Expert Intelligent Systems and Their Applications*, **12**(2):42-48.

World Wide Web Consortium (1995), *Hypertext Markup Language - 2.0*, World Wide Web Consortium (W3C) Standard, September, <http://www.w3.org/MarkUp/html-spec/html-spec_toc.html>.

World Wide Web Consortium (1998), *Extensible Markup Language (XML) 1.0*, World Wide Web Consortium (W3C) Recommendation, February, <http://www.w3.org/TR/REC-xml>.

APPENDIX A: FUNCTION TAXONOMY

(Top-level categories are shown in bold to improve readability)

Function

- **Usage-function**
  - Sink
    - Absorb
    - Consume
    - Destroy
    - Dissipate
    - Eliminate
    - Empty
    - Export
    - Remove
  - Source
    - Add
    - Create
    - Emit
    - Extract
    - Generate
    - Import
    - Supply
  - Storage
    - Accumulate
    - Collect
    - Store
- **Combination/distribution-function**
  - Branch
  - Combine
  - Connect
  - Couple
  - Distribute
  - Divide-flow
  - Link
  - Mix
  - Separate
  - Sort
- **Transformation-function**
  - Amplify
  - Attenuate
  - Convert
  - Decrease
  - Filter
  - Increase
  - Modify
  - Modify-form
  - Modify-property-magnitude
  - Refine
- **Conveyance-function**
  - Advance
  - Channel
  - Conduct
  - Convey
  - Direct
  - Divert
  - Guide
  - Generic-move
  - Rotate
  - Transfer
  - Translate
  - Transmit
  - Transport
- **Signal/Control-function**
  - Actuate
  - Adjust
  - Close
  - Decrease
  - Delay
  - Detect
  - Display
  - Equalize
  - Enhance
  - Generic-control
  - Identify
  - Increase
  - Indicate
  - Inhibit
  - Limit
  - Maintain
  - Mathematical/Logical
    - Add
    - AND
    - Decrement
    - Differentiate
    - Divide
    - Increment
    - Integrate
    - Invert
    - Multiply
    - NOT
    - OR
    - Shift
    - Sort
    - Subtract
    - XOR
  - Measure
  - Open
  - Resist
  - Retrieve-value
  - Select
  - Sense
  - Signal-processing
    - Amplify
    - Attenuate
    - Clear
    - Compare
    - Decode
    - Decrypt
    - Demodulate
    - Digitize
    - Encode
    - Encrypt
    - Filter

- Interrupt
- Isolate
- Modulate
- Reset
- Split-signal
- Store-value
- Switch
- Time
- Toggle
- Track
- Turn-on
- Turn-off
- Vary

Assembly-function

[Representative list; not comprehensive]

- Assemble
- Constrain
- Cover
- Disassemble
- Enclose
- Extract
- Fasten
- Fix
- Guide
- Join
- Link
- Locate
- Orient
- Position
- Release
- Remove
- Secure
- Separate
- Stabilize
- Support
- Unfasten

APPENDIX B: FLOW TAXONOMY

(Top-level categories are shown in bold to improve readability)

Flow

- **Material**
  - Generic-matter
  - Mass
  - Solid
    - Object
      - Generic-object
      - Rigid-body
      - Elastic-body
      - Widget
    - Powder
    - Particulate
    - Granular-matter
    - Composite-material
    - Aggregate-material
    - Non-homogeneous-solid-mixture
  - Liquid
    - Incompressible-liquid
    - Compressible-liquid
    - Homogeneous-liquid
    - Liquid-mixture
  - Gas
    - Homogeneous-gas
    - Gas-mixture
  - Plasma
  - Multi-phase-mixture
    - Solid-liquid-mixture
    - Colloidal-suspension
    - Liquid-gas-mixture
    - Liquid-particle-aerosol
    - Solid-liquid-gas-mixture
    - Solid-particle-aerosol
    - Solid-gas-mixture
- **Energy**
  - Generic
    - Generic-force
    - Generic-energy
    - Generic-power
    - Generic-impedance
  - Mechanical-domain
    - Compliance
    - Friction-force
    - Weight-force
    - Generic
      - Generic-motion
      - Generic-irregular-motion
      - Generic-oscillatory-motion
      - Generic-inertia
      - Generic-curvilinear-motion
      - Generic-relative-motion
      - Mechanical-kinetic-energy
      - Mechanical-potential-energy
        - Gravitational-potential-energy
        - Spring-potential-energy
    - Translational-domain
      - Translational-motion
      - Oscillatory-translational-motion
      - Relative-translational-motion
      - Position
      - Displacement
      - Velocity
      - Acceleration
      - Jerk
      - Impulse
      - Force
      - Inertia
      - Momentum
      - Translational-impedance
    - Rotational-domain
      - Rotational-motion

- Oscillatory-rotational-motion
- Relative-rotational-motion
- Orientation/angular-position
- Angular-displacement
- Angular-velocity
- Angular-acceleration
- Angular-jerk
- Angular-impulse
- Torque
- Rotational-inertia
- Angular-momentum
- Rotational-impedance
- Rotational-frequency

Electrical-domain
- Electricity/charge-motion
- Charge
- Voltage/electromotive-force
- Current
  - Generic-current
  - Direct-current
  - Alternating-current
  - Frequency
- Voltage-pulse
- Electrical-energy
- Electrical-impedance
- Resistance
- Capacitance
- Inductance

Thermal-domain
- Entropy
- Temperature
- Entropy-flow
- Heat

Hydraulic-domain
- Volume
- Volume-flow
- Volume-flow-rate
- Pressure
- Pressure-momentum
- Hydraulic-force
- Hydraulic-energy
- Hydraulic-impedance

Pneumatic-domain
- …

Magnetic-domain
- …

Acoustic-domain
- …

Electromagnetic-domain
- …
- Optical-domain
  - …

Chemical-domain
- …

Nuclear-domain
- …

Signal
- Generic-signal
- Analog
  - Generic-analog
  - Oscillatory
- Discrete
  - Generic-discrete
  - Binary