Function Modeling Based on Interactions of Mass, Energy and Information

Yang Bo and Filippo A. Salustri

Industrial and Manufacturing Systems Engineering
The University of Windsor, Windsor, Ontario, N9B 3P4, Canada
yang2@uwindsor.ca
salustri@uwindsor.ca

Abstract

The authors are working towards a universal product model theory that captures essential aspects of designed product representation. In a previous paper, Salustri established the basis of a theory of function modeling that could be integrated into the universal product model theory. The current paper expands on that work by establishing a small set of primitive function types in terms of basic interactions between mass, energy, and information. It is found that function is intimately coupled to many other aspects of product modeling, including ontology, mereology, and context logic. The authors believe it can represent most (perhaps all) product functions. Some implications of the framework for knowledge representation and for product modeling are discussed.

Introduction

The ultimate goal of the authors' research is to develop an integrated, logically rigorous knowledge representation for engineering design, including taxonomic and ontologic structure, function representation, and contexts of usage. This framework, called the Axiomatic Information Model for Design (AIM-D), is discussed in detail elsewhere [Salustri 1996]. AIM-D is intended to be useful at any stage of any design process. Therefore, knowledge about product function must be represented within the framework.

Many other researchers have identified the importance and relevance of function modeling, but a well-defined and universally accepted definition of the term *function* does not yet exist.

The current work builds on a function modeling framework proposed in [Salustri 1998]. The authors propose that function is an interaction between three fundamental kinds of entities: mass, energy, and information. This formulation leads to a small, arguably complete taxonomy of primitive functions, from which arbitrarily complex function descriptions can be developed. The authors' approach has yet to be formalized within the general logical framework of AIM-D. However, the outline presented here clearly defines the range and scope of the representation, and appears to be sufficient to represent many functions.

The rest of this paper is organized as follows. First, a brief discussion of the background work by Salustri is given. Then, a new definition of function intended to motivate our model of product function is introduced. The model itself is then presented, with comparisons to some other approaches found in the literature. Various reasoning tasks to which our model may be applied are then described, as well as the interrelations between function and other aspects of product modeling. Finally, concluding remarks and directions of future work are presented.

Background

This paper builds on the work presented in [Salustri 1998]. In that work, function and behavior are not seen as fundamental characteristics of real-world objects, but as descriptions of those characteristics that are dependent on the reference frame of the user of that information. From this observation, it is shown that a function in one context can be a behavior in another context, leading the author to propose that function and behavior are different views of the same thing, and thus should be uniformly represented in knowledge-based systems (KBSs).

Furthermore, [Salustri 1998] provides the basis of a representational form for so-called *predicative descriptions* (i.e. function or behavior). A *verb-object pair* (VOP) is suggested for this purpose. The verb part is intended to capture the active, dynamic part of the function, and the object part is intended to capture some notion of the entity upon which the function acts. It is then shown that abstraction can occur on both the verb and the object parts. While this kind of coupling makes some reasoning tasks (like subsumption) much more complex, it also provides a very rich expressive form for function and behavior.

[Salustri 1998] only sketches the model in preliminary terms. In the current paper, those ideas are pursued further, with particular emphasis on the identification of fundamental function types.

Function Definition

There are various, often conflicting, definitions of *function* in the literature; no universally accepted definition is

currently known. A brief review of some of these definitions is germane to the later sections of this paper.

Definitions from the Dictionary

Dictionaries can provide naïve, but useful, definitions. With respect to products, *function* is defined as (1) "a natural or usual purpose (of a thing)" and (2) "the way in which something works" [Longman 1987]. A purpose implies some intention or goal on the part of a designer or user; some researchers (e.g. [Prabhakar and Goel 1997]) have used this aspect in their work. The second definition suggests an explanation of a product's operation; though rare in the function modeling literature, this too has been used (e.g. [Salustri 1998]).

Though insufficient for our technical purposes, even a dictionary definition can provide some clues of what is most commonly held to be the meaning of *function.*

Definitions from the Technical Literature

The authors have found that definitions in the technical literature can be grouped coarsely into three categories.

The first category typified by [Prabhakar and Goel 1997a], has already been mentioned: function deals with the purpose of a product as intended by the designer. The problem here is that such definitions are cognitive rather than physical. While work in the cognitive aspects of design is very important, the current authors are more interested in establishing a basis for function rooted in reality rather than in our perceptions of it.

The second category of work treats product function as an effect on the environment of the product (e.g. [Chandrasekaran and Kaindl 1996].

The third category, exemplified by [Qian and Gero 1996], defines *function* as relationships between inputs and outputs of energy, mass, and information, or as changes in the fluxes thereof. This category is most closely related to the definition used by the current authors.

Clearly, each of these definitions has some aspects of worth, yet none are comprehensive enough to capture the fullness of definition that is desired.

A New Model of Function

The authors' research is working towards a more comprehensive and formal definition of function. Our current working definition is as follows:

> *A function is any mechanism by which transformations from one* ***basic element*** *- mass (M), energy (E), or information (I) - to another occurs in an entity, in response to environmental stimuli.*

That is, the function of a product describes all possible interactions between a product and its operational environment. Clearly, aspects of other definitions are present here.

The notion of identifying mass, energy, and information as basic elements has been very popular since it was first proposed [Rodenacker, 1971], and has withstood great scrutiny by many researchers. Its stability in this regard makes it very attractive to the authors, especially as the foundational layer of our emerging structure.

Function Types

If functions only occur between the basic elements, then there are only nine basic types of product functions. These types are enumerated in Table 1, where each row represents an initial state, each column represents a final state, and a cell describes a typical transforming function type. Each type is identified with a descriptor (in bold) and an example. Many different functions may fit into one cell. For example, the function in the M-M cell represents the movement of a mass, but M-M functions also include those that change the amount of mass in an entity (e.g. *to fill* or *to empty* or, more generally, *to store*).

The organization of each function type in Table 1 is a matter of on-going research. The selection of descriptors is itself problematic; for example, *to store* could be a valid descriptor for a M-M function, as described above, or for an I-I function such as storing information in a program. A more fundamental open issue in this regard is whether the use of the same descriptor for functions of different type (such as to store) indicates a true similarity of the functions or rather a cognitive or even linguistic artifact.

Another point of interest is the difference between function types that lie on the diagonal in Table 1 and those that lie off it. The diagonal types appear to be those that affect the characteristics of a thing without affecting its basic identity (e.g. mass remains mass). The function types off the diagonal seem to be those that can change both characteristics and basic identity. The implications of this observation are not clear at this time, but will be investigated.

Comparison to Other Approaches

Here we briefly compare our approach to function types to the some other approaches in the literature.

[Achinstein 1983] recognized three types of functions: design functions, usage functions, and service functions. This categorization is rooted in the cognitive roles played by designers and users, and as such falls outside the scope of the current authors' work, which focuses on inherent product function.

[Lind 1994] divided functions into two types:

- *Mass and Energy function*: There are six sub-type functions within this catalog. They are "source", "sink", "storage", "balance", "transport" and "barrier".
- *Action function*: There are four sub-type functions within this catalog. They are "maintain," "destroy," "produce," and "suppress."

The authors question the necessity of having sources and sinks, which are concepts taken from systems engineering. From an engineering perspective, the universe is a closed system. Sources and sinks are convenient approximations

that mark system boundaries, but we do not believe they are fundamental function types in any physical sense. Sources and sinks will be dealt with by other modeling constructs in the overall product model theory (AIM-D) of which the current work is a part. In addition, Lind's approach does not account for the inherent coupling between function and the object(s) upon which the function acts.

[Keuneke 1991] classified functions into four types: *to make*, *to control*, *to maintain* and *to prevent*. [Sasajima et al. 1995] uses the same form but add a new *to enable* type. Though substantially different from others, this approach, like Lind's, still does not recognize the coupling between functions and objects. Furthermore, due to their relatively abstract nature, it is difficult to use induction to argue that the list is complete: Sasajima et al. found it necessary to add a new function type four years after Keuneke's original work.

Overall, the authors believe a reasonable argument can be made for the completeness of our approach. Our reading of the literature suggests that categorizations of function in other research have been constructed in an inductive fashion, on the premise that it is impossible to enumerate *a-priori* all possible functions. In our approach, all the primitive functions arise out of a very small, very stable set of primitive entity types (i.e. mass, energy, and information). We find it hard to imagine a fourth entity type being suddenly discovered. Though this certainly does not mean our model if verifiably complete, it does suggest that its completeness is more likely than other approaches.

Table 1: Types of Product Functions

	M	E	I
M	**To move**: motion of one gear causes another gear to move.	**To power**: burning fuel gives off energy.	**To activate**: closing a switch sends a signal.
E	**To energize**: the volume of a heated fluid increases.	**To convert**: a wire carrying a current radiates heat.	**To detect**: a photocell responds to light with a signal.
I	**To actuate**: controller signals a robot to move.	**To regulate**: Amplifier output is controlled by received signal.	**To transfer**: digital to analog conversion.

Representing Product Function

The application of the product function model to KBSs for design is a fundamental aspect of the authors' work. To that end, we now consider one possible KB scheme for our function model.

A function specification (FS) has six attributes:

- *Function descriptor* – a specification, in the form of a verb, of the type of function (bold terms in Table 1);
- *Input descriptor* – the name of an object or type of object typical of the input used by the function or the initial state necessary for the function to occur;
- *Output descriptor* – similar to the input descriptor, but typical of the output or final state;
- *How link* – a reference to another FS describing the subfunctions needed to obtain the given one.
- *Why link* – the inverse of the *how* link.
- *Value* – the magnitude and direction of the function.

Each descriptor is represented by a term denoting a knowledge item. The terms' denotations are retrievable by an appropriate query engine. The FS itself would be named by a term. This scheme is easily implemented in many kinds of systems, including object-oriented systems like Java and description logic languages like CLASSIC.

The *how* and *why* links are used to connect function specifications into a function/behavior tree. This aspect is taken directly from [Salustri 1998].

The knowledge items denoted by terms would be arranged to form a specialization hierarchy. In order to remain consistent with our model, per Table 1, that taxonomy would have to contain "primitives" for mass, energy, and information, that represent the most general descriptions of things that provide functions.

The terms denoted by the function descriptors would be arranged in a specialization hierarchy also. However, this hierarchy is based on verbs or actions rather than nouns or things. It is unclear at this point what abstraction mechanism(s) are best suited for development of this taxonomy.

The representation shown here would couple the hierarchies of objects/things and actions/verbs, providing a linkage for reasoning engines to perform a variety of tasks, some of which are discussed briefly in the next section.

Potentially Supported Reasoning Tasks

The goal of any knowledge representation is to enable and facilitate automated and semi-automated reasoning processes. In this section, we introduce some of the reasoning tasks that the authors eventually intend to implement using our function model. This is not an inclusive list, but it does indicate the potentially broad application of our function model. For the sake of this presentation, we assume there exists an ontology of functions and objects for some particular application domain. Though the development of such an ontology is clearly problematic, it is beyond the scope of the authors' current work.

Function retrieval. Given some product that has been described in structural terms (i.e. in terms of parts and properties), we envision an automated system to "extract" functions. The system would match patterns of connectivity of parts in the structural description to the ontology relating object terms in the function specifications. It would then extract the functions provided by those patterns and suggest them as possible functions of the initial product. Such a system could be used to extract function information from "legacy" product models, or to verify that a product provides all the functions it is supposed to provide.

Function Decomposition. Functional analysis commonly proceeds by decomposing coarse-grained functional requirements usually provided at the outset of a design process into more fine-grained functional specifications. This decomposition may be aided by way of our model. For example, consider an automobile engine whose top-level functional requirement is to convert fuel (mass) to energy (i.e. a *power* function, per Table 1). There are three function types able to provide mass on their outputs: *move*, *energize*, and *actuate* functions. Furthermore, three function types use energy as inputs: *energize*, *convert*, and *detect* functions. Therefore, the framework can provide guidance for designers performing functional decomposition without placing unnecessary constraints upon them.

Case-based reasoning. An ontology of function specifications provides the foundation for creating a case library for a function-oriented case-based reasoning engine. This engine would be able to advise designers about potential structural descriptions of functionally stated design problems, based on similarities to cases in the library. This kind of system could be used during the early, upstream stages of a design process to suggest design concepts based on past experience of an enterprise.

This is of particular importance for routine or variant design tasks. The cases matched by the reasoning engine could represent alternative variations of existing designs, where the variations are developed by attempting to reconcile differences between function specifications of the given design problem with respect to cases in the library.

Top-down functional design. The KBS into which the authors' function model will be incorporated allows specification of partial knowledge (details to be presented in a forthcoming paper). Being able to represent partial function descriptions means that top-down development of functional specifications of products – again, very useful in the early stages of a design process – is possible. This kind of support can stimulate creative design by allowing designers to focus on exploring functionality separate from the structures that will provide those functions in a final product.

Explicating Customer Requirements. It is often the case that establishing customer requirements of a product is an iterative process involving designers, manufacturers, (representatives of) the client, and other specialists. The authors believe that a KBS supporting fully integrated function modeling could allow this process to occur in a more timely and effective manner in two ways. Firstly, it provides a framework for posing descriptions of functional requirements. Secondly, it allows straightforward translation of those requirements into KB systems that can be used to develop product models. We intend to pursue this matter more fully, once the model has been more fully developed.

Other Relevant Concepts

One of the authors' basic research goals is to develop an integrated theory of all aspects of product models. It is therefore necessary to examine the interrelations between our function model and other aspects of product modeling that are or will be incorporated into the overall theory.

Structure

Various methodologies are identified in the literature for mapping function to structure. For example, [Keuneke, 1991] advocates a two-pass process. In the first pass, the result of function decomposition is used to identify a component for each lowest-level function to be provided. In the second pass, these components are reorganized and integrated into the parts of the product. Based on our reading of the literature, the authors believe any of these methodologies can be constructed around our model; that is, the model is a representation that does not present unreasonable constraints on processes that may use it.

Context

"The main motivation for studying formal contexts is to resolve the problem of generality in AI. Context eliminates certain ambiguities or multiple meanings in the message." [Akman and Surav, 1996].

Contexts are crucial in our model because they implement the mapping between terms used to name functions, and the objects and types within function specifications. When many designers work on the same design, there will invariably be mismatches between what they know and the names they use to identify those items of knowledge. This is also true of software agents of every sort. By explicitly incorporating context logic into our overall theory, the authors are providing support for multiple definitions of terms within the theory and, by extension, within KBSs implementing the theory.

In particular, making the theory context-sensitive will provide support for the arbitrary naming of functions and components of function specifications, while ensuring that the knowledge in those specifications can be transmitted between agents (human or otherwise) in a reliable fashion.

Mereology

Mereology is the branch of logic dealing with the study of part-whole relations. Insofar as parts and wholes are fundamental concepts in engineering, mereology is an essential aspect of product model theory. Function decomposition requires a mereological relationship. Furthermore, function defines the role that an object plays in a larger system as well as the way that the physical structure of the object responds to external stimuli. That is, function relates an object's physical structure (parts) to systems that contain it (wholes). Thus, function and mereology are tightly coupled concepts. Salustri and Lockledge (1999) are currently working towards a formalization of mereology that will be consistent with the current work herein on

function modeling.

Completeness versus Validity

The authors' model of function is not yet formalized. However, the authors offer an informal argument to support our contention that our representation is complete. Our model enumerates completely the possible function types based on binary combinations of the basic elements (ternary and other combinations can be constructed from these). All functions must be classifiable into one of the nine types in Table 1. This argument assumes that there are only three basic elements. Although the authors cannot imagine others, this is not sufficient for a formal proof. We hope to make headway in this regard in the near future.

Further Work

Substantial work remains to be done on this model. One of the authors, Yang, is pursuing its development as a graduate level thesis. Some important open issues include:

Canonicalization of Function Types. Some of the open issues regarding the proposed representation of function types (Table 1) have already been discussed; these issues indicate that some of the nine basic types could be derived from others. Determining if this is indeed the case will depend on further study of our domain of interest (engineering design), and the kinds of ontological commitments needed to integrate the function model into AIM-D.

Relations between Function Types. Clearly, relationships exist between the function types in Table 1. For example, *move* functions will typically require *power* functions to provide the energy for motion. The authors intend to seek a categorization of these relationships. This could aid in the development of tools to support function-based reasoning tasks (such as case-based reasoning).

Improving Computational Efficiency of KBSs. The eventual goal of this work is to develop a KBS for design engineers that will include function representation and reasoning facilities. In order to do this, however, various computational issues (such as subsumption and decidability) ought to be examined, to determine the limits of computations that can be reasonably carried out in a system incorporating our function model.

Representing Intention/Purpose. It is typical of many engineering disciplines to consider design intent as a subset of product function. In keeping with this perspective, we hope to study design intent after the function formalism proposed herein has matured.

Acknowledgement

The authors gratefully acknowledge the National Sciences and Engineering Research Council of Canada for funding this work under grant number OGP0194236.

References

Achinstein, P. 1983. The nature of explanation. Oxford: Oxford University Press.

Akman, V., and Surav, M. 1996. Steps toward formalizing context. *AI Magazine* 17(3):55-72.

Chandrasekaran, B., and Josephson, J. R. 1996. Representing Function as Effects: Assigning Functions to Objects in Context and Out, *AAAI Workshop on Modeling and Reasoning with Function*, pages 30-37.

Chandrasekaran, B., and Kaindl, H. 1996. Representing Functional Requirements and User-system Interactions. AAAI Workshop on Modeling and Reasoning about Function, pages 78-84.

Keuneke, A. 1991. Device Representation, *IEEE Expert*, April, pages 22-25.

Lind, M. 1994. Modeling goals and functions of complex industrial plants. *Applied Artificial Intelligence* 8:259-283.

Longman World Publishing Corp. 1987. Longman Dictionary of Contemporary English.

Prabhakar, and Goel. 1997. Addressing Incompleteness of Device Models by Adaptable Function Modeling of Devices for Operating Environments. To appear in *Artificial Intelligence in Engineering*. Elsevier Applied Science.

Qian, L. and Gero, J.S. 1996. Function-behavior-structure Paths and their role in analogy-based design. *Artificial Intelligence for Engineering Design, Analysis and Manufacturing* 10:289-312.

Rodenacker, W. 1971. Methodisches Konstruieren. Berlin: Springer-Verlag.

Salustri, F.A. 1996. A formal theory for knowledge-based product model representation. In *Knowledge-Intensive CAD II: proceedings of the IFIP WG 5.2 workshop*. Chapman & Hall.

Salustri, F.A. 1998. Function Modeling for an Integrated Framework: a Progress Report. Proc. FLAIRS-98.

Salustri, F. A. and Lockledge, J. C. 1999. Towards a Formal Theory of Products Including Mereology. To appear, Proc. 12th Int'l Conf. on Engineering Design.

Sasajima, M., Kitamura, Y., Ikeda, M., and Mizoguchi, R. 1995. FBRL: Function and behavior representation language. Proc. IJCAI-95, pages 1830-1836.