Proceedings of ICAD2002
Second International Conference on Axiomatic Design
Cambridge, MA – June 10&11, 2002
Put paper number here

SYSTEM DEFINITION FOR AXIOMATIC DESIGN AIDED BY OBJECT-PROCESS METHODOLOGY

Nathan R. Soderborg
nsoderbo@ford.com
Ford Motor Company
MD 561, PDC Design Center
21175 Oakwood Blvd.
Dearborn, MI 48124

Edward F. Crawley
crawley@mit.edu
Massachusetts Institute of Technology
Room 33-207
77 Massachusetts Ave.
Cambridge, MA 02139

Dov Dori
dori@ie.technion.ac.il
Technion, Israel Institue of Technology
Haifa 32000, Israel
and
Massachusetts Institute of Technology

ABSTRACT

This paper suggests ways to improve formulation of Functional Requirements and Design Parameters in Axiomatic Design based on the contention that adequate descriptions of both function (WHAT) and architecture (HOW) require a *combination* of objects and processes. We describe how the definitional framework and expressive power of Object-Process Methodology can be used to represent system function and architecture. We introduce Object-Process Methodology templates for describing function and architecture, and apply these templates as an example to a simple system.

Keywords: axiomatic design, function, system architecture, object-process methodology, functional requirements, design parameters

1 INTRODUCTION

Representing systems well is an important task *and* tool for system architects. A good representation not only communicates what the system is and how it operates, it helps the architect develop the system, providing a means for organizing elements, understanding functional relationships, identifying critical interfaces, and guiding implementation and innovation.

Object-Process Methodology (OPM) and Axiomatic Design (AD) both provide useful representations of systems. OPM is a *descriptive* method. It represents systems through visual diagrams and textual descriptions. AD is an *evaluative* method. It represents systems through matrices that depict the presence of important relationships between functional requirements (FRs) and design parameters (DPs). Its axioms provide the basis for evaluating whether or not a design is "good."

As conceived by Nam Suh, AD provides a "scientific approach to design and synthesis" [Suh, 1990, p. 5]. The approach is rigorous for evaluating and comparing designs once FRs and DPs have been clearly defined, but the FR and DP formulation process is not yet rigorously specified. Whether this is even possible is debatable due to the creative element in such formulation. The use of natural language to capture and communicate FRs and DPs further complicates the problem. Rigor calls for elimination of subjectivity and ambiguity. In this paper, we explore how the formal system representation capability of OPM may serve as a sound basis for developing a rigorous approach to the definition of systems in terms of FRs and DPs.

1.1 OBJECT-PROCESS METHODOLOGY

OPM provides methods for representing systems both graphically and textually. Graphical representations, known as Object-Process Diagrams (OPDs), specify how objects are structured and how processes transform them by creating or consuming them, or changing their states. OPDs convey complex interconnections and non-linear relationships according to established standards. Textual representations, composed in Object-Process Language (OPL), provide English language descriptions of each OPD. Together, a set of OPDs and its corresponding OPL script, specify a system.

OPM departs radically from the Object-Oriented paradigm currently prevalent in software system development. OPM recognizes *processes* as stand-alone entities in addition to *objects*. The basic premise of OPM is that *objects and processes are two types of equally important classes of things*. Together, objects and processes faithfully describe the system's structure, function and behavior in a single, coherent model, in virtually any domain." OPM elements of OPDs fall into three categories: entities, procedural links, and structural relations. Entities are objects (symbolized by rectangles), processes (ellipses), and states (rounded-corner rectangles within objects). A set of triangular symbols represents fundamental structural relations. Various directed lines that connect processes to objects represent procedural links and process enablers.

1.2 OPM EXAMPLE

As a brief introduction to OPM, we share a simple example that conveys basic OPM constructs. The diagram in Figure 1 shows an example of a car decomposed into major systems with extra detail shown for the braking system. This detail includes the "Stopping" process and objects related to this process. Figure 2 contains the OPL text that is uniquely specified by the diagram. This text communicates in simple, standardized language what the diagram represents. Each link in the diagram corresponds to an OPL sentence. (Numbers are added to the diagram to help identify the link with its corresponding sentence.) Sentences appear in standard OPM style, using bold text for words that are not "reserved" with specific meaning in OPL.

2 "WHATS" AND "HOWS"

The "WHAT-HOW" decomposition is a classic approach to system design. WHAT refers to *what* is desired—an objective. HOW refers to *how* the objective is met—a solution. An entire system can be detailed by successively identifying the WHATs and

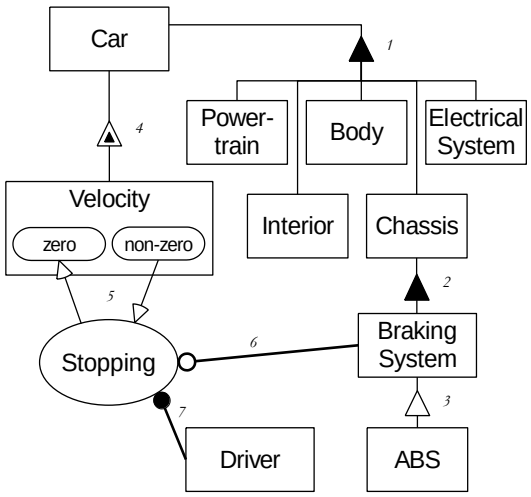


Figure 1. An example Object-Process Diagram (OPD)

1. **Car** consists of **Powertrain, Body, Electrical System, Interior** and **Chassis.** (Aggregation Sentence)
2. **Chassis** consists of **Braking System.** (Aggregation Sentence)
3. **ABS** is a **Braking System.** (Specialization Sentence)
4. **Car** exhibits **Velocity,** which can be **non-zero** or **zero.** (Exhibition and State Enumeration sentence)
5. **Stopping** changes **Velocity** from **non-zero** to **zero.** (Change sentence)
6. **Stopping** requires **Braking System.** (Instrument Sentence)
7. **Driver** handles **Stopping.** (Agent Sentence)

Figure 2. The Object-Process Language paragraph that specifies the OPD in Figure 1

HOWs at each level of the design hierarchy. This paradigm has been incorporated in Quality Function Deployment (QFD) and has been adopted by Suh in Axiomatic Design. Suh's version of WHATs and HOWs are FRs and DPs. He defines FRs as "[a] minimum set of independent requirements that completely characterize the functional needs of the product…in the functional domain;" and defines DPs as "the key physical variables…in the physical domain that characterize the design that satisfies the specified FRs" [Suh, 2001, p. 14]

2.1 OBJECT-PROCESS NECESSITY

Rigorous formulation of FRs and DPs requires careful definition of these terms. Words such as *function, functional need*, and *physical variable* mean different things to different individuals. OPM offers a framework for defining these words formally in terms of objects and processes. One's first impulse may be to answer the question WHAT? with an object—typically expressed as a noun. Similarly, one might naturally answer the question HOW? with a process—typically expressed as a verb. However, Suh recommends the opposite: "…FRs are stated in the imperative starting with verbs, whereas DPs usually are stated with nouns [Suh, 2001, p. 20].

We contend that *both* objects *and* processes are necessary to fully express both WHATs and HOWs. In the case of WHATs, many techniques for identifying system function employ the "verb-noun" rule. This rule specifies that a function be described by an active verb together with a noun. The verb corresponds to a process; the noun to an object. Furthermore, in distinguishing FRs from constraints, Suh says "a specific range of design values must be maintained for each FR at all times" [Suh, 1990, p. 29]. Causing this *level* to be established and maintained within the desired range is the requirement. In OPM the level is a *state* or *value* of an attribute object, and the activation of an associated process is required to change that state or value. Thus, the representation of a fully expressed FR ought to include at least one process that changes the states or values of an object.

Such "Object-Process Necessity" also applies to HOWs. Although Suh confines the definition of DPs to physical variables—representable as objects—the *way* in which a variable satisfies an FR is a process. It is the more natural answer to the question "HOW?" Dori suggests a HOW is properly expressed as an architecture—a *structure/behavior combination* that attains the functional WHAT [Dori, 2002]. In OPM, structure is represented by objects connected by structural links; behavior is represented by a combination of processes and objects connected by transformation links. Hence, as is the case for WHATs, full expression of HOWs requires both objects and processes.

2.2 DEFINITIONS OF FUNCTION: WHATS

Suh defines function generally: "By the word *function* we mean the desired output" [Suh, 1990 p. 38]. Otto and Wood provide a more specific definition for *product* function.

> "A function of a product is a statement of a clear, reproducible relationship between the available input and the desired output of a product, independent of any particular form…The product function is the overall intended function of the product—what it is to do; [it] is the simplest representation of the product, usually just a noun and an active verb." [Otto and Wood, 2001, p. 151]

This definition is based on guidelines of Pahl and Beitz, who have developed an extensive methodology for engineering design [Pahl and Beitz, 1996]. Applying the term function specifically to the conversion of energy, material, or signals in engineering applications, they capture the flow of these items within a system in diagrams known as *function structures*. Examples of functions recorded in these diagrams might be "increase pressure," "transfer torque," or "reduce speed."

Each of these definitions describes function as an action fulfilling a purpose, or succinctly stated: "process with intent" [Crawley, 9/9/2000, p. 24]. In OPM function is an attribute of an object that describes what the object does, what phenomenon it exhibits, what service it supports, or what it is used for. This definition emphasizes WHAT and is not concerned with HOW. It distinguishes function from dynamics, as dynamics is about *how* the object operates, while function is about *what* it does.

A secondary but important element in these definitions is that function is independent of form. It describes the intended *effect* of a system's operation on the beneficiary user and the environment, not the operation itself. Thus multiple systems can fulfill the same function. Conversely, a particular system may provide a variety of different benefits to different users. Each of these users derives a different intended effect; hence function is

ultimately defined by the system's beneficiary and is not necessarily objective.

2.3 Object and Process Elements of Function

Our description of function as "process with intent" explicitly identifies process as the dynamic or "action" element of function. Others also recognize this connection: "A function is defined in terms of a description of a process." [Otto and Wood, p. 165] Such processes describe the architect's intended *service* that is to be provided by the system or the user's intended *use* of the system. As an example, consider a freezer, one of the systems Suh examines in his books. The intended service the freezer provides is to preserve food. Stated as an OPM process, this service is **Food Preserving**.

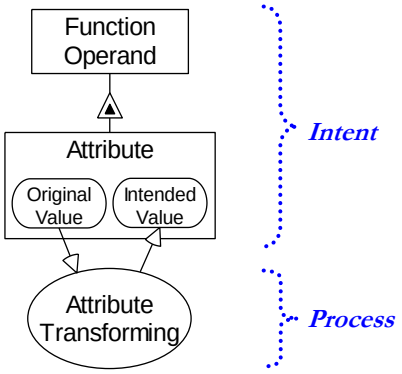
A basic tenet of OPM states that no process exists unless it is associated with at least one object, for the transformation of which it is responsible. For a function process, the associated object is the function operand, without which intent cannot be fully revealed. In OPM such an operand is called a "transformee," emphasizing that a process transforms the object. In the freezer example, food is the operand or transformee; the freezing process transforms food by changing the it from an unfrozen state to a frozen state. Intent is fully revealed by identifying the desired change of states in the object.

OPM allows the creation of a standard representation for function that captures both its process and object elements. Figure 3 portrays an OPM template for a generic function. (Note: dotted lines and words in italic font are annotations not actually part of the OPD.) The process element of function appears in the OPD in **Attribute Transforming**. The object element appears in the objects **Operand** and its **Attribute**. Intent is made explicit by identifying the desired change of values in **Attribute**. Note that it is not always necessary to separate the attribute from the operand. Sometimes it makes sense to show the transforming process acting on operand states directly. However, identifying an attribute explicitly can be useful because it will typically correspond to an important performance metric.

2.4 Definitions of Architecture: HOWs

Typical definitions of architecture in the context of system design emphasize *structure*. For example, Rechtin and Maier describe architecture as "structure—in terms of components, connections, and constraints—of a product, process or element" [Rechtin and Maier, 1997, p. 251]. But other definitions go beyond structure. Crawley has defined architecture as the "embodiment of concept and the allocation of functionality and definition of interfaces among the elements [Crawley, 9/8/2000, p. 11]. Otto & Wood describe architecture as "the mapping from product function to the product form" [Otto and Wood, 2001, p. 358]. Ulrich and Eppinger capture a similar idea, but more abstractly. "The architecture of a product is the scheme by which the functional elements are arranged into physical chunks and *by which the chunks interact*" [Ulrich and Eppinger, 2000, p. 183, italics added].

This last definition identifies both object (physical chunks) and process (interaction) elements. The OPM principle that accurate representation of systems requires equal status of objects and processes leads to a definition of system architecture that clearly recognizes both the static and dynamic aspects of



Operand exhibits **Attribute** with values **Original Value** and **Intended Value.**

Attribute Transforming changes **Attribute** of **Operand** from **Original Value** to **Intended Value**

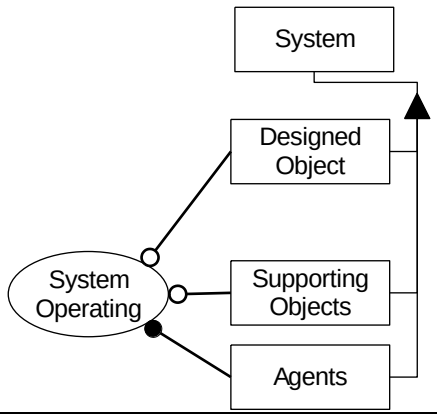
Figure 3. OPD and OPL Script for a generic Function

architecture: "System architecture is the overall system's structure-behavior combination, which enables it to attain its function while embodying the architect's concept." [Dori, 2002, p. 261].

2.5 Object and Process Elements of Architecture

The static aspect of architecture is structure, represented in OPM by structural relations among objects. These objects are the physical (or informatical) elements of a system: "the parts, components, and subassemblies that ultimately implement the [system's] functions" [Ulrich and Eppinger, 2000, p. 183]. In the freezer example, a sensor and a compressor operate to maintain air temperature. This sensor/compressor combination is comprised of objects that are part of the refrigerator structure. Combined with the other parts, they enable the freezing process.

The process elements of architecture are the operational elements of a system, represented in OPM by processes: "the individual operations and transformations that contribute to the overall performance of the [system]" [Ulrich and Eppinger, 2000, p. 182]. In the freezer example, the sensor *senses* air temperature and *signals* the compressor if the temperature gets too high; the compressor *cools* the air. These operations describe in part *how* the freezer *freezes*, fulfilling the function of food preserving.



System consists of **Designed Object**, **Supporting Objects**, and **Agents**.

System Operating requires **Designed Object** and **Supporting Objects**.

Agents handle **System Operating**.

Figure 4. OPD and OPL Script for a generic System Architecture

Figure 4 portrays an OPM template for a generic architecture that formalizes the key idea that a HOW includes processes *and* objects. In this representation, the process is **System Operating**. **System** is *generic* in that it includes

- **Agents**: humans that operate or control the system.
- **Designed Object**: the object designed by the architect that acts within the system as a solution to fulfill the function.
- **Supporting Objects**: additional objects in the system's environment—not designed by the architect—required for the system to fulfill the function.

In a typical WHAT-HOW decomposition it is the designed object that is usually singled out as the HOW, because this is the object that the architect's organization will produce. However, other elements in the system's operating environment usually affect successful fulfillment of the function. Capturing these elements can be important for creating a good system representation.

2.6 Function vs. Architecture: Summary

We have made a distinction between function and architecture that aligns function with the question "What is the system supposed to do?" and architecture with "How does the system do it?" Architecture is a "structure-behavior" or structure-dynamics combination [Dori, 2002]. Our observation of the dual object/process nature of function leads to an analogous description of function as an "operand-use combination." The use is what the system is supposed to do—the service intended by the architect or the use intended by the user; the operand is what is affected or transformed. Table 1 organizes these ideas under the categories of WHATs and HOWs:

Table 1. Organizing WHATs & HOWs via OPM

WHAT? What result do you desire?		**HOW?** How does the system achieve it?	
Function: Operand-Use Combination		**Architecture:** Dynamics-Structure Combination	
Object Element	**Process Element**	**Process Element**	**Object Element**
What should be affected?	What is the desired effect?	How does the system operate?	How is the system structured?
Operand-State, Transformee	Use, Service	Behavior, Operation	Structure, Form

The table subdivides the high-level WHAT and HOW questions into object- and process-related sub-questions. Together, these four sub-questions elicit the basic information required for a good WHAT-HOW decomposition of a system:

1. ***What should be affected??***
   (object: operand attribute)
2. ***What is the desired effect??***
   (process: changing attribute's value or state)
3. ***How does the system operate?***
   (process: system operating)
4. ***How is the system structured?***
   (object(s): system elements and their relationships)

The order of the questions follows the general flow of system development. It is rare that one question is answered *completely* before moving on to the next question. Iteration will occur, especially between Questions 1 and 2 and between Questions 3 and 4. Question 3 follows Question 2 because its answer—the dynamic part of the HOW—is a specific solution to the dynamic part of the WHAT. In OPM terms, the answer to Question 3 is a "specialization," of the answer to Question 2. It is represented in an OPD by a specialization link (white triangle) between the function-related process and architecture-related process.

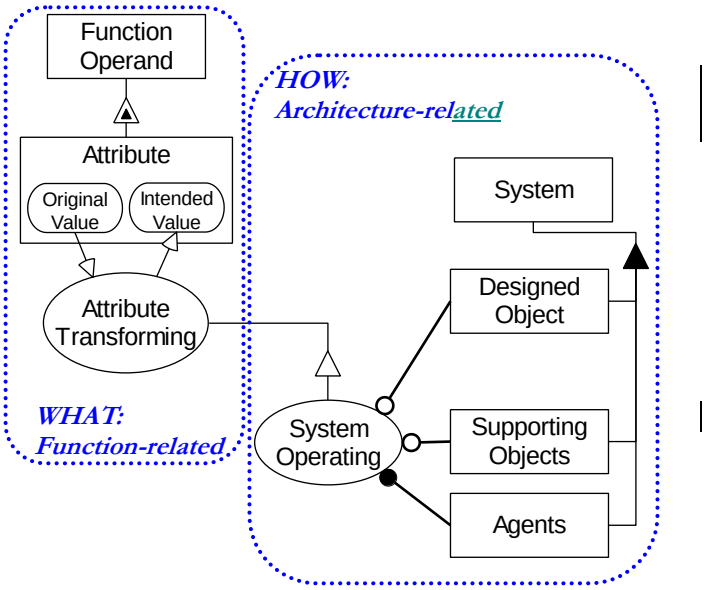


Figure 5. Generic OPD Template for WHATs & HOWs

Figure 5 combines the OPDs for function and architecture into one generic template by connecting them with a specialization link. The OPL script for this diagram is the combination of scripts for the two previous diagrams with the addition of the specialization sentence:

System Operating is **Attribute Transforming**.

This simple sentence reveals the system architect's choice of *concept*. Concept is the "product or system vision, idea, notion or mental image that maps form to function and embodies working principles" [Crawley, 1/23/2001, p.16-17]. In the freezer example, freezing using a freezer is the selected concept for food preserving. Of course, this function can be fulfilled by other concepts as well, e.g., canning using a cannery or dehydrating using a dehydrator. Each of these concepts conveys a *general* idea for a process and associated objects that carry out that process; thus each is the initial description of architecture for the system to be designed.

3 FORMULATING FRs AND DPs

System design is typically a non-linear, iterative process. Suh summarizes the process in the following way:

> "…the design process begins with the recognition of a societal need. *The need is formalized, resulting in a set of FRs*. The selection of FRs, which defines the design problem, *is left to the designer*. Once the need is formalized, ideas are generated to create a product (or an organizational structure). This product is then analyzed and compared with the original set of FRs through a feedback loop. When the product does not fully satisfy the specified FRs, then one must either come up with a new idea, or change the FRs to reflect the original need more accurately. This iterative process continues until the designer produces an acceptable result." [Suh, 1990, p.27]

The axioms of AD are used primarily in the feedback loop of this process. They help the designer analyze and evaluate a design once FRs and DPs have been chosen, but they do not provide a means of initially formulating useful FRs and DPs in a repeatable way. Such formulation, an essential and challenging aspect of AD, is "left to the designer." Suh states: "the perceived needs must be reduced to an imaginative set of FRs as the first and *most critical* stage of the design process. In the absence of a proper set of FRs, a good design is not likely to result," [Suh, 1990, p.32, italics added].

What makes a "proper" set of FRs? What distinguishes FRs from other types of requirements and constraints? What makes a useful set of DPs? What distinguishes DPs from design concepts? We now explore techniques and examples that can help develop formal answers to these questions. Building on our OPM representation of WHATs and HOWs, we show how the template in Figure 5 and questions in Table 1 provide tools for formulating useful FRs and DPs and stating them in a manner that minimizes ambiguity.

3.1 FRs

The distillation of societal needs into useful and unambiguous requirements is an exercise in communication comprised of *recognition* and *formalization*. Recognition of needs involves the mental processing and assimilation of knowledge through observing, reading, and listening (as well as, perhaps, tasting, smelling, and touching). Formalization typically requires the conversion of this "sensed" knowledge into language, and thus is inherently susceptible to ambiguity. For example, early in Suh's first book, he lists the four highest-level FRs for a solar-powered car as "simplicity, efficiency, light weight, and reliability" [Suh, 1990, p. 31]. This list requires refinement and clarification before it can be considered useful and unambiguous.

Over time, some general guidelines for stating FRs have been captured. For example, "[FRs] characterize the functional needs of the product" [Suh, 2001, p. 14]; "FRs are stated in the imperative starting with verbs" [Suh, 2001, p. 20]; "FRs must be defined in a solution-neutral environment" (for innovative designs) [Suh, 2001, p. 14]; "[A] specific range of design values must be maintained for each FR at all times" [Suh, 1990, p. 29].

OPM can aid formalization of requirements by providing a framework to address five key elements of requirement definition [Sommerville and Sawyer, 1997, p.141]:

- A template
- Simple and consistent use of language
- A diagram
- A supplement to natural language
- Quantitative specification.

In this paper we have addressed the first four of these elements. Quantitative specification, though not explicitly address is easily incorporated in our framework by specifying quantitative object values in OPDs. We have presented a template and diagram in the form of a generic function-architecture OPD. We have ensured simple and consistent use of language by adhering to the rules of OPL that accompany each diagram. We have provided a supplement to natural language by linking OPL paragraphs to OPDs.

Ensuring that FRs always answer the two WHAT questions will provide a repeatable process for phrasing FRs more formally. Constructing FRs using the OPM template will clearly identify the essential objects and processes associated with an FR and document the architect's intent. This will help eliminate ambiguity that often arises in FR definition.

3.2 DPs

The translation of requirements into an effective design requires a mixture of creativity and judicious application of lessons learned from previous designs. This mental process involves formulating a concept. Suh speaks of finding DPs by "conceptualizing a solution" [Suh, 2002, p.18] As we have noted, this conceptualization includes identifying the process and object elements that combine as an architecture to embody the HOW. Suh limits his definition of DPs to "key physical variables…that characterize the design" [Suh, 2001, p. 14]. Thus, even though he describes DPs as HOWs, his definition makes a DP a subset of a HOW that is an attribute of the solution but not the solution itself.

One guideline for DPs is that they "are usually stated with nouns" [Suh, 2001, p. 14]. Actual FR-DP decompositions show that DPs, although stated as nouns (objects), typically represent a system solution that includes an implicit process. It may not be

until the decomposition terminates at the lowest component level that DPs are truly "parameters," i.e., basic part or process characteristics adjustable by the system architect during design and representable in mathematical equations from which partial derivatives can be computed for the design matrix.

In a fashion similar to that for FRs, OPM helps reduce ambiguity in the definition of DPs. The five key elements for clearly communicating requirements are equally valid for clearly communicating design solutions. In addition, ensuring that DPs are identified properly through answering the two HOW questions will help make the formulation of DPs more formal and repeatable. Constructing DPs using the OPM template will clearly identify the essential objects and processes associated with a DP, placing the physical variables in the context of the entire system solution intended by the architect.

3.3 EXAMPLE

We have used a freezer system to help illustrate our points so far. Suh includes two different examples related to a freezer in his books [Suh, 1990, p.8 and Suh, 2001, p. 20, 34]. Here we combine elements from both those examples to show how the OPM template and WHAT-HOW questions can add clarity to the FR-DP formulation process through two levels of decomposition.

The system architect begins by asking the questions:

1. What should be affected? The societal need is long-term preservation of food; **Food** (the operand) should be affected.

2. What is the desired effect? Food should be preserved.

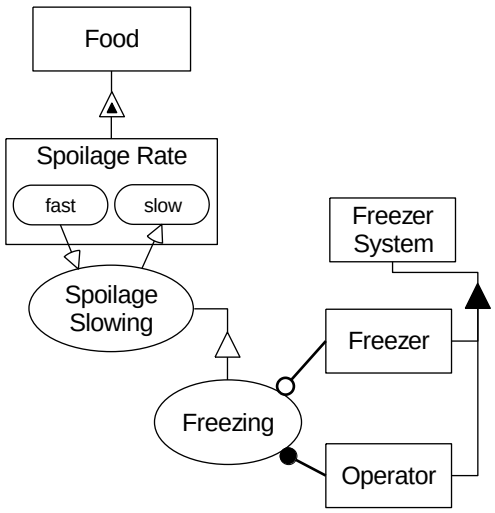
The architect tries to capture these ideas in the template. This requires explicitly identifying the desired change of state in **Food**. After some thought, the architect iterates through the questions again, deciding that the desire is to extend the "shelf-life" of food, which is equivalent to slowing the spoilage rate of the food. The phenomenon of "spoilage slowing" is perhaps easier to measure objectively, so **Spoilage Rate** is identified as the attribute (metric) of food that should be affected. The specific change required is that this rate must be changed from **fast** to **slow**. These are states that can be quantitatively specified based on the needs of the system users, (the "slow" state can be defined with a specific quantitative target value).

The architect now asks:

3. How does the system operate? The answer selected is **Freezing**. This is the desired method for slowing spoilage.

4. How is the system structured? The architect envisions a **Freezer** comprised of an enclosure or cabinet with an interior temperature below freezing. People must place or remove food from the cabinet, so the overall **Freezing System** includes an **Operator**.

The development of the system to this point is captured in the OPD in Figure 6. The architect has selected a concept for the system and begun defining it's architecture. The object **Freezer** may be referred to as a DP, but it is necessary to comprehend the entire concept in order to represent the system and define FRs at the next level of decomposition. To formulate these FRs the architect re-asks Questions 1 and 2 about the architecture defined so far. The suggestion of Pahl and Beitz to address conversion of energy, material, and signals helps narrow the focus. Additionally, the concept will imply important FRs.



Food exhibits **Spoilage Rate,** which can be **fast** or **Slow.**

Spoilage Slowing changes **Spoilage Rate** from **fast** to **Slow.**

Freezing is **Spoilage Slowing.**

Freezing System consists of **Freezer** and **Operator**.

Freezing requires **Freezer**.

Operator handles **Freezing**.

Figure 6. OPD and OPL for First Level of Freezer System

1.1 What should be affected? The temperature of the air enclosed in the cabinet. Heat energy needs to be removed from the interior of the cabinet.

1.2 What is the desired effect? Change the temperature of the enclosed air from high to low (FR1).

2.1 What should be affected? Because FR1 cannot be accomplished if the transfer of heat through the cabinet is too high, the rate of heat transfer should be affected by a heat transfer slowing process.

2.2 What is the desired effect? Change the heat transfer rate from fast to slow (FR2).

3.1 What should be affected? The conceptual architecture employs a cabinet that encloses the food content. Food cannot be accessed without a means to open the cabinet. The cabinet should be affected by a content accessing process.

3.2 What is the desired effect? Change the cabinet from open to closed (FR3).

It is useful to compare the FRs generated by this process to those from Suh's original examples. Suh's statement for FR2 was "Minimize energy consumption." This is a need, not entirely transparent as a single function. This need could apply to properties of either the cooling system or cabinet. In our formulation at this level of decomposition, we link it specifically to a function of the cabinet (which is how Suh ended up applying it). There is an opportunity at the next level of decomposition to link this need to energy consumed by the cooling system.

Suh's statement for FR3 was "Provide access to the items stored." This sentence is an imperative addressed to the architect—not a functional description of the system. What does

it mean to enter an "x" in the design matrix for a DP against an FR stated in this way? It implies that adjusting the DP will affect the actions of the architect, in this case, "providing access." Although most people would understand this is not the proper interpretation of Suh's original FR, our template and questions provide additional rigor to reduce ambiguity that could arise.

Continuing on with defining the HOWs, the architect asks the related questions:

1.3 How does the system operate? The freezer cools the temperature of the enclosed air by operating a cooling system.

1.4 How is the system structured? As a cooling system that consumes electrical energy. Based on their functions, the elements of the cooling system (compressor, sensor, blower, condenser) will appear in the next level of decomposition.

2.3 How does the system operate? The cabinet insulates the enclosed air from external heat, slowing the heat transfer rate.

2.4 How is the system structured? As insulation inside the cabinet.

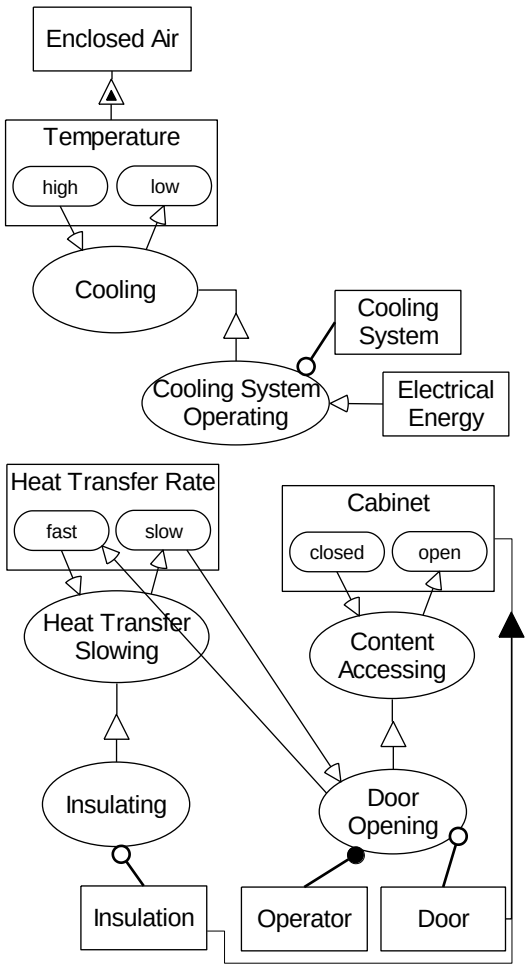
3.3 How does the system operate? The cabinet opens to allow content accessing.

3.4 How is the system structured? As a door located in the cabinet.

The summary of the system at the second level of decomposition is captured in Figure 7. In this OPD the **Freezing** process of the first OPD is—in OPM terminology—"zoomed into." What are the DPs in this diagram? Suh's DPs were a freezing system (DP1), "thermal insulation material in the door" (DP2), and "vertically hung door" (DP3). These are objects that comprise part of the HOWs. But knowing these alone is not enough to eliminate confusion in filling out the design matrix. For example, how does one determine whether DP3 affects FR2? Using thinking analogous to computing a partial derivative, one should ask whether a change in the "vertically hung door" causes a change in "heat transfer slowing." But it is not possible to answer this question unless the kind of change in DP3 is clearly understood. The change could be "opening the door," in which case there is certainly a change in the heat transfer rate. In fact, the OPD clearly indicates this through the effect links from **Door Opening** to **Heat Transfer Rate**. These links are an explicit OPM representation of an "x" in the design matrix connecting DP3 and FR2. However the kind of change for DP3 that was intended in Suh's example is a change in orientation of the door. By changing the door from a vertical to a horizontal orientation, Suh observes that the effect of opening the door on the heat transfer rate may be sufficiently small to say that DP3 is no longer affects FR2. In this case, the real "parameter" that the architect can change is "orientation," which is an attribute of the door.

4 SUMMARY

Combining Object-Process Methodology with Axiomatic Design creates a powerful synergy that formalizes early stages of system design. Formulating Functional Requirements and Design Parameters using an OPM template stimulates system architects' creativity and enables them to follow an ordered set of WHAT-HOW questions to define a system at increasing levels of granularity. OPM provides means for graphic and natural language specification that reduces ambiguity in FRs (WHATs) and associated DPs (HOWs). Additionally, the OPM representation of a system can add clarity to filling out the design matrix. Future work should further explore the relationships between a system's OPD set and the design matrix at the various refinement levels.



Enclosed Air exhibits **Temperature,** which can be **high** or **low.**

Cooling changes **Temperature** from **high** to **low.**

Cooling System Operating is **Cooling.**

Cooling System Operating requires **Cooling System**.

Cooling System Operating consumes **Electrical Energy**.

Heat Transfer Rate can be **fast** or **slow.**

Heat Transfer Slowing changes **Heat Transfer Rate** from **fast** to **slow.**

Insulating is **Heat Transfer Slowing.**

Insulating requires **Insulation**.

Cabinet can be **open** or **closed.**

Cabinet consists of **Door and Insulation.**

Content Accessing changes **Cabinet** from **closed** to **open.**

Door Opening is **Content Accessing.**

Door Opening changes **Heat Transfer Rate** from **slow** to **fast.**

Door Opening requires **Door**.

Operator handles **Door Opening**.

Figure 7. OPD and OPL for Second Level of Freezer System

5 ACKNOWLEDGMENTS

We acknowledge Ford Motor Company for support of Mr. Soderborg while a student in the System Design and Management program at Massachusetts Institute of Technology. This work is based on Mr. Soderborg's thesis for which Professors Crawley and Dori were advisors.

6 REFERENCES

[1] Crawley, Edward, Lecture Notes for MIT Course ESD.34J, Systems Architecture, January and Fall 2000.

[2] Crawley, Edward, Lecture Notes for MIT Course ESD.34J, Systems Architecture, January and Fall 2001.

[3] Dori, Dov, *Object-Process Methodology: A Holistic Systems Development Paradigm*, Springer-Verlag, 2002.

[4] Otto, Kevin N., and Wood, Kristin L., *Product Design: Techniques in Reverse Engineering and New Product Development*, Prentice Hall, 2001.

[5] Pahl, Gerhard, and Beitz, Wolfgang III, *Engineering Design: A Systematic Approach*, Translated by Ken Wallace, Springer-Verlag London Limited, 1996.

[6] Rechtin, Eberhart and Maier, Mark W., *The Art of Systems Architecting*, CRC Press, 1997.

[7] Soderborg, Nathan R., "Representing Systems through Object-Process Methodology and Axiomatic Design," *Master's Thesis,* System Design and Management Program, Massachusetts Institute of Technology, 2002.

[8] Sommerville, Ian, and Sawyer, Pete, *Requirements Engineering: A Good Practice Guide*, John Wiley & Sons, Chichester, England, 1997.

[9] Suh, Nam P., *The Principles of Design*, Oxford University Press, 1990.

[10] Suh, Nam P., *Axiomatic Design: Advances and Applications*, Oxford University Press, New York, 2001.

[11] Ulrich, Karl T. and Eppinger, Steven D., Product Design and Development, 2nd Ed., McGraw-Hill, 2000.