A Fractal Representation for Systems

Jason D. Hintersteiner
Axiomatic Design Group
Massachusetts Institute of Technology
77 Massachusetts Avenue, #31-261
Cambridge, MA 02139
jdhinter@alum.mit.edu

Abstract: To facilitate the design of evolving systems, tools are needed to capture all performance issues and evaluate design ideas and proposals quickly during the conceptual stage, so that better designs can be generated. By using such tools, engineers can quickly identify and understand how their design decisions impact and are impacted by choices made concerning other components in the system. Thus, rational design decisions will be made during conceptual design, minimizing if not eliminating the need to address design problems during implementation.

This paper presents a methodology, based on axiomatic design theory, for constructing a *system architecture* for complex systems that standardizes the classification of functions and modules used to represent a system. This is important for several reasons, including capturing the performance requirements and components of the system in a logical, coherent, and comprehensive manner, facilitating communication between engineers and managers on a large design project, and providing good technical documentation of the design decisions made and the reasoning behind them. A system architecture is applicable to systems of any size, including systems that are subsystems of a larger system. Thus, the decomposition of a system follows the same general pattern and layout at each level of the design hierarchy where the parent design parameter is a "system" in its own right. Hence, the overall system is represented in a *recursive* manner throughout the design hierarchy. This representation has several advantages, the most significant of which is that the design of individual subsystems can be generated by different teams of engineers on a large design project, while the functionality of the overall system as well as the interrelationships between the different subsystems are thoroughly and consistently represented.
Keywords: System Architecture, Systems Engineering, Design Rationale, Design History, Axiomatic Design

1. INTRODUCTION

When a system becomes sufficiently complex, the design process must be distributed among several engineering design teams, each of which are given smaller and more manageable tasks to accomplish. These teams are supervised by one or more layers of management, which have the difficult task of coordinating the activities of the different teams and providing a bridge of communication as and when necessary.

Typically, each design team will try to optimize its design based on its assigned tasks and constraints. It is very easy, however, for a designer to be unaware of other designer's decisions which will have a negative impact on his/her design. Thus, when each of the individual designs are put together, it is quite common to discover that the overall system does not function as intended. This results from the fact that each individual subsystem design has been locally optimized, without accounting for the complex interactions between the subsystems. Thus, a change to one portion of the design can negatively impact other portions unintentionally, because no framework exists to trace the impact of the design choices between the task divisions.

The main source of difficulty in designing large systems is that, in most cases, good representations of the design either do not exist or are not used to their full potential. [Söderman, 1998]. Hence, it is extremely important to have a methodology to trace the impact of design decisions on both a local level as well as a system level, since the real goal of the design effort is to optimize the performance of the *system*, which does not necessarily mean optimizing the performance of each *component*.

This issue can be addressed by applying axiomatic design theory to the design of complex systems. This approach generates a *system architecture*, which captures the hierarchical structure of the functional requirements (FRs), design parameters (DPs), and constraints (Cs) of a system. Using the design axioms, the quality of the design can be evaluated by means of design matrices, so that potential problems of a particular design can be detected and addressed during the design process. Axiomatic design is reviewed briefly in *Section 2*.

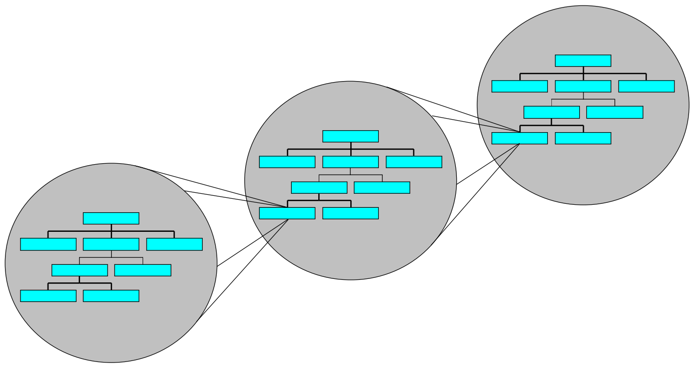

Figure 1: *The concept of a fractal representation for systems. Each system is only one part of a larger system's hierarchy, but the pattern and layout of the representation for each system remains consistent.*

In this paper, a *system* is distinguished from a *component* in the sense that a system not only consists of process functionality, but requires distinct functionality for controlling and supporting the processes. Hence, by treating every level of the hierarchy as a "system" composed of "subsystems", these types of functional requirements appear at every hierarchical level. Thus, the system is represented in a *recursive* manner, so that no matter how deep in the hierarchy one looks, the general pattern and layout of the representation remains consistent. This concept is shown graphically in *Figure 1*. Accordingly, a fractal representation emerges of the different subsystems incorporated into the hierarchy of the overall system, where the details at each level are unique, but the overall types of functions represented by the FRs and their interrelationships with the DPs remains consistent. This representation is described in *Section 3*, and a case-study example of this technique is discussed in *Section 4*.

2. BACKGROUND: AXIOMATIC DESIGN

Axiomatic design provides a framework for describing "design objects" which is consistent for all types of design problems and at all levels of detail. Thus, different designers can quickly understand the relationships between the intended functions of an object and the means by which they are achieved. Additionally, the design axioms provide a rational means for evaluating the quality of proposed designs, and the design process that is used guides designers to consider alternatives at all levels of detail and to make choices between these alternatives more explicit.

In axiomatic design terminology, design is defined as the development and selection of means (DPs) to satisfy objectives (FRs), subject to constraints (Cs). The main concepts of axiomatic design include: (1) *domains*, which separate conceptually the functional and physical parts of the design; (2) *hierarchies*, which categorize the progress of a design in the functional and physical domains from a system level to more detailed levels; (3) *zigzagging*, which indicates that the decisions made at one level of the hierarchy affect the problem statements at lower levels; and (4) *design axioms*, which dictate that the independence of the FRs must be maintained (Independence Axiom) and that the information content (i.e. cost, complexity, etc.) must be minimized (Information Axiom) in order to generate a design of good quality. For a more thorough explanation of axiomatic design theory, refer to [Suh, 1990] and [Suh, 1999].

3. SYSTEM ARCHITECTURE: APPLICATION TO SYSTEMS

By comprehensively documenting the interrelationships between the FRs, Cs, and DPs at every level of the design hierarchy, a system architecture can be generated and used as a tool for decision making. The strength of the system architecture is that, in addition to the operational flow of the system, it also captures the order in which design decisions have to be made, and indicates how the alteration of one part of the system can potentially impact other parts. [Tate, 1999] Here, *system design* is defined as a design in which the task of the designers is to integrate several DPs into a whole in order to satisfy some set of FRs, subject to a given set of constraints. This differs from other design methodologies in that the process followed is: (a) identify the FRs and their corresponding constraints, (b) determine possible DPs, and (c) integrate them into a system. In this case, an understanding of the interrelationships among the different DPs is vitally important. By not doing so, the design process becomes a confusing muddle, which can ultimately lead to several iterations and poor design decisions. In system design, it is desirable that the same process be useful at all levels of the design; the same approach may be followed recursively, starting at the system level and continuing until the design is complete. Thus, the system architecture is useful both when design concepts are initially generated and evaluated, as in new designs, and when changes are proposed to existing designs.

3.1. Classification of functions at every hierarchical level

One of the fundamental assumptions in the development of the system architecture is that the system can be modeled as a series of interacting inputs and outputs. In a general sense, the input-output transformations in systems can be broken into three functional categories: process functions, command and control functions, and support and integration functions.

3.1.1. Process functions

The process subsystems perform the "physical processing" of operands (e.g. a part being produced, data being evaluated and manipulated, etc), as well as the transport of operands through the system (e.g. obtaining a part from an external environment and moving it to different process stages, data transmission from one network node to another, etc.). For example, in a semiconductor manufacturing machine, the process subsystems represent the different processing stages through which the wafer passes within the machine, as well as the robots or other wafer handling mechanisms that transport wafers between the different processing stages. Within each subsystem, all of the necessary sensors and actuators, along with the algorithms to control the individual activities and coordinate them with the other activities in the subsystem, are specified as part of the functional decomposition. In addition, all of the support structures and assemblies which bind the subsystem together are also delineated. In a good system design, process FRs and DPs should either be uncoupled or decoupled from each other. At the system level, transport DPs are generally decoupled from physical processing DPs, since the nature of the physical processes usually dictates the nature of the required transport.

3.1.2. Command and control functions

Command and control algorithms (CCAs) are used to represent the logic necessary to schedule and coordinate the different process subsystems at each level of the hierarchy. A CCA will use the instantaneous output of the individual process and transport operations at that level (i.e., as reported by CCAs lower in the hierarchy) to make necessary adjustments to ensure that the final operand outputs are as desired (i.e., as specified by the parent CCA). These algorithms appear at every level of the design hierarchy where active control is required. They are decoupled from the process subsystems, since the design of the process DPs dictates which parameters are monitored and controlled. CCAs at higher levels are primarily responsible for scheduling and coordination, while CCAs at the lowest levels are primarily responsible for coordinating individual outputs to actuators based on information from sensors and the operators. [Hintersteiner & Tate, 1998]

In many complex systems, the functional requirements of the system change over time. These types of systems require CCAs that are designed with sufficient flexibility so that different DPs can be allocated at different times to meet the needs of the changing FRs. Thus, system command and control is required to allocate different DPs at different times, as part of the normal operation of the system. The information provided in the system architecture is useful for understanding the order in which such DP values should be changed during operation. [Tate, 1999]

3.1.3. Support and integration functions

Each level of the hierarchy requires a support framework for binding the process subsystems and control logic together into a coherent system. In mechanical systems such as manufacturing machines, the support framework includes requirements for environmental control, pneumatic/water lines, electronics assemblies, layout and configuration, and framing/panel assemblies. In software systems, the support and integration functionality includes resetting variables and operational states, garbage collection, and so forth.

3.2. Constraints

Constraints are defined as the set of performance specifications and design restrictions which impact FRs and therefore limit the acceptable range of possible design solutions (DPs). Since the system architecture organizes the design in terms of its functionality, it can be very useful for determining how marketing and management constraints impact the FRs and restrict the scope of the DPs. The performance specifications provided by the customer, management, government and industry standards, and safety regulations are specified as constraints at the system level.

Like FRs and DPs, constraints can be refined and clarified as decomposition progresses. Many high level constraints also influence the specification of lower-level FRs. In general, the more constraints that exist at the system level, the harder it will be to generate an acceptable set of DPs, and therefore the harder it will be to maintain an uncoupled or decoupled design. By tracing the constraints and refining them as the design is decomposed, the impact of the high-level constraints on low level functionality (i.e., child FRs) can be determined. Hence, the way each constraint impacts each FR must be examined. The impact will, of course, eliminate some potential DPs. Once a DP is chosen, the constraints are combined and/or refined to more specific constraints at the child level.

By understanding how constraints impact the FRs and are refined through the hierarchy, the sources of error in the design that negatively impact the performance of critical parameters can be determined. Once these sources of errors are identified, tolerance allocations can be made in the form of an *error budget*. Research is currently being undertaken in this area to determine the best method for representing and tracing constraints through the system architecture. [Hintersteiner, et al., 1999]

3.3. Evaluation of the design at each level

Table I shows a generic listing of FRs and DPs. The top row indicates the index at this level, where "#" refers to the index in later rows, and the "φ" indicates the full index of the parent FR/DP.[1] The parent FR and DP are included in the table in order to place the FRs and DPs at this level in context with their parent FR/DP. At the top level, the parent FR is the mission statement for the overall system, and the parent DP is the overall system itself. There can be an *arbitrary but non-zero number* of process FRs at each level of the hierarchy, depending upon the specific design, and the indices can be renumbered accordingly. These will *always* be followed by one control FR and one support/integration FR.

There may be situations where alternative choices for DPs may exist, such as cases where a selection must be made between two or more possible design alternatives, and cases where a particular system platform may utilize different DPs at different times. In such situations, alternative DPs can be listed as shown in *Table I*. Each alternative DP will, in general, have different sets of sub-FRs and constraints, and so a separate decomposition should be provided for each alternative.

Table I: *Generic listing for FRs and DPs (any hierarchical level).*

Index: j.#		TITLE		
	Functional Requirements (FRs)		**Design Parameters (DPs)**	
	Name	*Description*	*Description*	*Ver*
		Parent FR	***Parent DP***	
1	*Process*	Perform physical process #1	Process subsystem #1	A
2	*Process*	Perform physical process #2	(a) Process subsystem #2 (1st option) (b) Process subsystem #2 (2nd option) (c) Process subsystem #2 (3rd option)	In
3	*Process*	Perform physical process #3	Process subsystem #3	De
4	*Transport*	Perform process #4 (transport)	Transport subsystem	Dr
5	*Control*	Schedule and coordinate all local process functions	Command and control algorithm (CCA)	T
6	*Support*	Integrate subassemblies	Support framework	U

$$\begin{bmatrix} \text{Perform Process \#1} \\ \text{Perform Process \#2} \\ \text{Perform Process \#3} \\ \text{Perform Process \#4} \\ \text{Control processes} \\ \text{Integrate subass'y} \end{bmatrix} = \begin{bmatrix} X & O & O & O & O & O \\ ? & X & O & O & O & O \\ ? & ? & X & O & O & O \\ ? & ? & ? & X & O & O \\ X & X & X & X & X & O \\ X & X & X & X & X & X \end{bmatrix} \begin{bmatrix} \text{Process subsystem \#1} \\ \text{Process subsystem \#2} \\ \text{Process subsystem \#3} \\ \text{Transport subsystem} \\ \text{CCA} \\ \text{Support Framework} \end{bmatrix} \quad (1)$$

The last column in the table is used for verification codes, so that the designer can specify the verification procedure to ensure that the DP is satisfying its corresponding FR. Verification may be done by several means, including testing (T), inspection (I),

[1] e.g. If φ = 3.4a.2, the parent FR is FR.3.4a.2, and the FRs in this table are FR.3.4a.2.1, FR.3.4a.2.2, etc.

demonstration (De), drawings (Dr), analysis and simulation (A), or proven (i.e., unchanged) technology (U).

When the list of FRs and DPs is formulated at a particular hierarchical level, a design matrix is used to correlate how the DPs impact the FRs. An axiomatic design matrix equation of the form {FR}=[A]{DP}is constructed and the matrix elements evaluated, as shown in *Equation 1*.[2] Note that the order of FRs in a lower-triangular matrix generally indicates the order of design importance, since the FR/DP in the top row should be the first to be decomposed. [Tate, 1999] An explanation should be provided for each off-diagonal "X" in the design matrix.[3] In general, the control and support FRs will be impacted by the specific design of the process DPs, and hence at least the last two rows will be decoupled, as shown in *Equation 1*.[4] Coupling can occur between the process DPs, as well as between the process and control and support DPs, though this is undesirable as it violates the Independence Axiom.

Given that most large-scale systems evolve from earlier systems when new constraints emerge, the system architecture enables the designer to determine whether a proposed change to the design has many downstream side effects. Ideally, a proposed change should impact only a few, if any, FRs; this occurs when the design is completely or mostly uncoupled. Typically, however, a design change may impact many other DPs, which can be very undesirable in terms of monetary costs and development time. The combination of design matrices and hierarchies can be used to trace proposed changes throughout the design and understand their effects. This includes choosing between alternative concepts, guiding the sequence of the design, and developing options for decoupling coupled parts of an existing design. [Harutunian, et al., 1996]

3.4. Decomposition

As the decomposition progresses, each process DP can be treated as a system in its own right, with its own sub-process FRs as well as requirements to schedule and coordinate these process FRs and integrate its own subassemblies. Therefore, the decomposition follows the same format at every level, and each subsystem has its own CCA and support framework, which influences the design of the parent level CCA and support framework. Thus, as the process functions are decomposed top-down, the full system logic and system support framework is built both top-down and bottom-up, due to the decoupled nature of the control and support FRs at every level of the hierarchy. In

[2] Each element A_{ij} (row i, column j) in the matrix is evaluated by asking, "Can we design (or change the existing design of) DP.j without impacting how we address FR.i?" [Tate, 1999] In the design matrix, a "X" signifies a relationship between DP.j and FR.i, and an "O" signifies that no relationship exists. The diagonal elements (i=j) in [A] should be "X", since DP.i is chosen to satisfy FR.i.

[3] A "?" or "*" is used within the design matrix in cases where the relationship between an FR and a DP is either variable or unknown. In some instances, a description may also be desirable for an "O", especially in cases where the "O" is only valid under certain conditions, or where an "O" appears in an on-diagonal term, meaning that the DP does not satisfy its corresponding FR.

[4] Note, if a particular process subassembly is completely passive and requires no active control, there will be a zero in the corresponding column for the control FR.

addition, constraints on these FRs will emerge from parent-level constraints and parent-level design decisions.

It should be noted that, at sufficiently detailed levels, it may not be appropriate to consider the decomposition of a design object as a subsystem. For example, the design of a specific hardware component, such as the geometric specification of a basket at the end of a movable mechanism, does not qualify as an independent subsystem. Since the basket itself does not require active control or any form of external support that is not already specified by the larger subsystem of which it is a part, it is considered to be a *component* of the subsystem, and the system representation presented in this paper does not apply to the decomposition of the component. It is therefore an important job for the designer to make the distinction between *systems* and *components*.

3.5. Documentation of the design

Design documentation is typically performed at the very end of the design project, and often omits discussion of the reasoning behind design decisions. Accordingly, most design documentation efforts do not specify what problems were encountered during the design process, or what steps were taken in order to make the design function properly. As a result, a future designer only has access to information about the end product, and thus will know very little about the rationale behind the original decisions. By generating a system architecture, documentation of the design is generated *while the design is progressing.* This can prove to be very advantageous during the design process, as areas of potential coupling can be detected early on – the system architecture can be used as a communication tool between different design teams, so that one design team can avoid making design decisions that will have a negative impact on other design teams. Furthermore, at the beginning of a redesign effort for a next-generation product, the system architecture can be reexamined, so that the reasoning behind the choices of certain DPs along with an understanding of the constraints under which the designers had worked can be more clearly understood. Where it is appropriate, photographs, CAD drawings, and/or schematic diagrams showing the assemblies at that level of the hierarchy can be provided. Such figures should label the name and DP index number for the subsystems shown, to illustrate better what the DPs in the design matrix represent and how they interact with other DPs in the system.

4. CURRENT APPLICATIONS

This methodology has been and continues to be applied to several case-study examples. One good example of this is application of this technique to the current design of the Micrascan III Photolithography Tool, manufactured by SVG Lithography Systems, Inc. The system level FRs and DPs are shown in *Table II*. The first three FRs show the different processes (i.e. lithography, wafer transport, and reticle transport), followed by an FR to control the interactions between the processes and an FR to integrate the processes and control. The corresponding design matrix is shown in *Equation 2*. Since this product is the basis of current development platforms, the system architecture is currently being used to identify improvement areas in the design, as well as demonstrate the potential impacts of changing performance requirements (i.e. constraints) on different parts of the design.

Table II: *System level decomposition of the Micrascan III.*

Index: #	**MICRASCAN III PHOTOLITHOGRAPHY TOOL**			
	Functional Requirements (FRs)		**Design Parameters (DPs)**	
	Name	*Description*	*Description*	*Ver*
		Print patterns onto wafers coated with photoresist	***Micrascan III Photolithography Tool***	
1	*Lithography Process*	Transfer and overlay pattern from reticle to wafer	Photolithography process system	A, T
2	*Wafer Transport*	Manipulate wafers	Wafer handling system	A, T
3	*Reticle Transport*	Manipulate reticles	Reticle handling system	A, T
4	*Control*	Schedule and coordinate all local process functions	MSIII command and control algorithm (CCA)	De
5	*Support*	Integrate subassemblies	MSIII support framework	I, T

$$\begin{bmatrix} \text{Transfer Pattern} \\ \text{Manipulate Wafers} \\ \text{Manipulate Reticles} \\ \text{Control processes} \\ \text{Integrate subass'y} \end{bmatrix} = \begin{bmatrix} X & O & O & O & O \\ X & X & O & O & O \\ X & O & X & O & O \\ X & X & X & X & O \\ X & X & X & X & X \end{bmatrix} \begin{bmatrix} \text{Photolithography Process} \\ \text{Wafer Handling System} \\ \text{Reticle Handling System} \\ \text{MSIII CCA} \\ \text{MSIII support framework} \end{bmatrix} \qquad (2)$$

Current development work for this methodology includes the following: (1) incorporation of issues related to physical integration of DPs (i.e. increasing physical coupling while avoiding functional coupling), (2) development of methods to allow constraints to be traced coherently and consistently through the system architecture, (3) expansion of the representation to include a coherent representation for software systems, and (4) creation of software tools to assist in the generation and maintenance of system architectures for large systems.

5. CONCLUSIONS

This paper has presented the concept of developing a system architecture, based on axiomatic design theory, to assist in the design process for new and evolving systems. By organizing the system design hierarchically in terms of its functionality and applying the design axioms, the designer is forced to methodically examine the elements of the design matrices to judge whether or not interrelationships exist. Thus, the system architecture is a very good tool for documenting design decisions and the reasoning behind them, and can thus be used to facilitate the evolution of the system by tracing the impact of proposed changes. If the designers strictly follow axiomatic design principles, a design can be created in which functional requirements are satisfied independently, with minimum information content. In the case of an existing design, a design that already exists may or may not satisfy the design axioms, but the generation of a system architecture can reveal potential options for decoupling the design as it evolves.

For complex systems, this methodology shows how a system and its composite subsystems can be represented to maintain consistency throughout the design hierarchy. This representation can then be used to manage and evaluate design changes, determine the impact of constraints on the set of viable design solutions, create command and control algorithms for both static systems and systems where FRs change over time, and facilitate the development of design documentation and the sharing of information between designers during the design process.

REFERENCES

[Harutunian, et al., 1996] Harutunian V., Nordlund M., Tate D., Suh N. P., "Decision Making and Software Tools for Product Development Based on Axiomatic Design Theory," *Annals of the CIRP*, Vol. 45/1, 1996.

[Hintersteiner & Tate, 1998] Hintersteiner, J. D. and Tate, D. "Command and Control in Axiomatic Design Theory: Its Role and Placement in the System Architecture." *Proceedings of the 2nd International Conference on Engineering Design and Automation,* Maui, HI. August 9 – 12, 1998.

[Hintersteiner, et al., 1999] Hintersteiner, J. D., Tate, D., Friedman, G., and Zimmerman, R. "Representation of Constraints in the System Architecture." *Work in progress.*

[Söderman, 1998] Söderman, M. "Tools for Creating Understanding and an Integrated Dialogue in the Early Stages of Product Design." *Proceedings of the 2nd International Conference on Engineering Design and Automation,* Maui, HI. August 9 – 12, 1998.

[Suh, 1990] Suh, N. P. The Principles of Design. *Oxford University Press,* NY. 1990.

[Suh, 1999] Suh, N. P. Axiomatic Design: Advances and Applications. *Work in progress.*

[Tate, 1999] Tate, D. " A Roadmap for Decomposition: Activities, Theories, and Tools for System Design." *Ph.D. Thesis,* Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA. February, 1999.