Proceedings of ICAD2000
First International Conference on Axiomatic Design – Cambridge, MA – June 21-23, 2000
ICAD022

IMPLEMENTING AXIOMATIC DESIGN IN THE SYSTEMS ENGINEERING PROCESS: AN AXIOMATIC DESIGN CAPABILITY MATURITY MODEL

Jason D. Hintersteiner
hintersj@svg.com
SVG Lithography Systems, Inc.
901 Ethan Allen Highway
Ridgefield, CT 06877

Richard C. Zimmerman
zimmermanr@svg.com
SVG Lithography Systems, Inc.
901 Ethan Allen Highway
Ridgefield, CT 06877

ABSTRACT

Since its inception, Axiomatic Design has been applied to a wide variety of both engineering and non-engineering problems in numerous disciplines. Recent theoretical developments have further expanded its ability to represent complex engineering systems. The scope of Axiomatic Design usage, however, has traditionally been limited to particular applications or projects. Axiomatic Design, however, can be most useful as a design management tool applied across an engineering organization's Systems Engineering processes. The benefits of this approach would be both in technical areas related to the quality of the organization's designs as well as financial benefits in the effectiveness of Axiomatic Design in decreasing the organization's time to market and ultimately increasing customer satisfaction with more functional and more reliable products.

The success of such an implementation effort requires not only integrating Axiomatic Design methodology into all aspects of engineering practice, but also measuring and tracking the organization's capability and effectiveness in Axiomatic Design, in order to continuously refine and improve the integration efforts. To this end, an Axiomatic Design Capability Maturity Model has been developed, in order to provide a roadmap for implementation as well as coherent metrics that can be applied to determine which activities are successful and which activities may require improvement.

Keywords: Axiomatic Design, Systems Engineering, Design Management, Capability Maturity Model

1 INTRODUCTION

"Anyone trying to understand Systems Integration who isn't confused is just not thinking clearly."
-- Joseph H. Mize (1999)

While the fundamental concepts first put forth by Suh [1990, 2000] have been around for over 20 years, Axiomatic Design as a design methodology and philosophy has traditionally not been widely accepted outside academia. This trend, however, is not due to the methodology's lack of applicability to real-world engineering problems – indeed, numerous case studies by several researchers have been made that apply Axiomatic Design to problems in disciplines including mechanics, manufacturing, computer networking, software, optics, and biology. Nor is this trend due to a lack of proven business benefit – many analyses done by companies have demonstrated savings in time, money, and customer satisfaction. One example is a redesign of a flexure for a prealigner system at SVG Lithography Systems, Inc., where the lead time, part count, and manufacturing time were all significantly reduced while reliability was considerably increased. [DelPuerto and Garcia, 2000] An even more notable example is a case where an engineer from Standard-Thomson Corporation, while taking an evening class in Axiomatic Design, redesigned his company's factory floor for his class project. This new design ultimately saved the company over $2.4 million, and the engineer was promoted to Vice President. [McDonald, 1998]. The trend also does not seem to stem from a lack of interest or awareness of Axiomatic Design – the conference forum of this paper and the formation of the new Council on Axiomatic Design Research and Education (CADRE) reflects the growing interest in Axiomatic Design and its applications in both industry and academia.

Why, therefore, has Axiomatic Design met with such resistance outside academia?

One of the greatest obstacles is that technology transfer from academia to industry for *most* engineering design methodologies, including Axiomatic Design, has been difficult. Most corporately-sponsored research is directed at solving particular problems – hence, the end result is what tends to be transferred, without the methodology used to achieve that result. Furthermore, new engineering graduates rarely enter corporate positions immediately where they have the ability to influence the company's design methodology. [Fredriksson *et al.*, 1994]

Another traditional obstacle has been the difficulty in establishing guidelines consistent with the axioms, theorems, and corollaries of Axiomatic Design for determining FRs, DPs, Cs, and design matrices for complex systems. Work by Hintersteiner [1999, 2000], Suh [2000], Tate [1999] and others has mitigated this to some extent by the establishment of system templates and guidelines for system design. It is true, though, that not all of the issues related to complex system representation have been worked out, proving this area of Axiomatic Design is still a fertile area for research and development.

The most difficult and challenging obstacle, however, is that Axiomatic Design has not been readily extendable from the individual engineer working on his/her particular design problem to an organization-wide methodology capable of managing and controlling the design of an entire system. Two general approaches to implementing Axiomatic Design in an engineering organization have been taken in the past – either a top-down decree from management, or a bottom-up diffusion of knowledge from trained engineers. Both approaches have their advantages and risks. [Nordlund *et al.*, 1996] The problem with both of these approaches, however, is that few companies have had the level of in-house expertise, management support, and long-term planning necessary for success. Many difficult questions emerge and must be addressed early-on in such efforts, including:

- How can the Axiomatic Design process be structured to maintain both performance and consistency between designers?
- Which people in the organization need to be trained in Axiomatic Design, and what are the best methods for supervising and evaluating their work?
- How can the benefits to the organization achieved by using Axiomatic Design be measured and evaluated?

The goal of this paper, therefore, is to address these questions by understanding the different levels of Axiomatic Design maturity that an organization must go through in order to successfully implement Axiomatic Design. This is done by means of an *Axiomatic Design Capability Maturity Model* (AD CMM), which can be used to guide the development of both the Systems Engineering practices needed as well as the metrics that can be used to determine the organization's technical capability and business effectiveness in implementing Axiomatic Design.

This paper is organized as follows. Section 2 provides a brief overview of Axiomatic Design and the system architecture template that has been developed to apply Axiomatic Design to complex systems. Section 3 discusses the Axiomatic Design Capability Maturity Model. Section 4 highlights examples of the AD Systems Engineering practices and AD implementation metrics that are based on the AD CMM. Section 5 summarizes the discussion and provides a basis for future work.

2 AXIOMATIC DESIGN AND THE SYSTEM ARCHITECTURE TEMPLATE

"We must learn how to study a whole as a whole, not merely through an analysis of each of its constituents."
– Mary Parker Follett (1927)

When a system has a sufficiently large number of functions and many levels of detail, the design process is distributed among several engineering design teams, each of which is responsible for a subset of the design tasks. A *system* may be distinguished from a *component* by the fact that a system consists not only of process functionality but also requires functionality for controlling and supporting the processes.

Typically, each design team optimizes its design based on its assigned tasks and constraints, without necessarily accounting for the many design interrelationships that exist between the subsystems. It is disturbingly easy, therefore, for a team to be unaware of "external" design decisions that can have a negative impact on its design. Thus, when all of the separate designs are assembled, it is quite common to discover that the overall system does not function as intended. The real goal of the overall design effort is to optimize the performance of the system, which does not necessarily mean optimizing the performance of each component.

Problems can be compounded when the statement of the customer needs changes during the design effort. Because companies consider it infeasible to restart the design process from scratch, new and modified functional requirements and constraints are incorporated into the existing design as the changes occur. When trying to compensate for this, a change to one portion of the design can negatively impact other portions unintentionally. Moreover, one design team may not be aware of changes to another group's requirements, even though they are significantly impacted. Therefore, it is extremely important to have tools and methods to trace the impact of design decisions on both local and system-wide levels. [Hintersteiner, 2000]

For example, consider the relationship between hardware and software. Recent efforts in concurrent engineering have sought to perform hardware and software design in parallel in order to reduce overall development time. This cannot be done effectively, however, unless the intended functional requirements of both the hardware and the software and their interrelationships are well-understood. Hardware engineering and software engineering, however, are all-too-frequently treated as separate disciplines, with independent groups of designers, in product development. Because the design of the software control logic depends on the nature of the hardware to be controlled, software is typically developed after the hardware is mostly, if not completely, defined. According to Leveson [1995], this dependency leads to very short development times allocated for software as well as the last-minute addition of functionality to the software because of either hardware limitations or the perception that it will be "easier" to implement certain tasks in software. This philosophy often leads to undue software complexity as well as unpredictable system behavior. While the hardware design may be optimized, the quality of the overall system suffers as seen in the user interface and in the non-satisfaction of requirements that were identified late in the design process.

These issues can be addressed by applying Axiomatic Design to the design of complex systems. Using this approach, designers generate a *system architecture* that captures the hierarchical structure and the interrelationships between the functional requirements, design parameters, and constraints of the system. Additionally, the design axioms provide a rational means for evaluating the quality of proposed designs, make choices among these alternatives more explicit, and guide designers to consider solution alternatives at all levels of detail. Thus, the quality of the design can be evaluated so that the potential problems of a design can be detected and avoided during an early stage of the design process.

2.1 Axiomatic Design Theory

The *design process* is defined as the set of activities by which designers develop and/or select of a means (design parameters, or DPs) to satisfy objectives (functional requirements, or FRs), subject to constraints (Cs). 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, as well as observers to the design process, can quickly understand the relationships between the desired functions of an object and the means by which the functions are achieved.

The main concepts of Axiomatic Design are (1) *domains*, which separate 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 systemic level to more detailed levels; (3) *zigzagging*, which indicates that decisions made at one level of the hierarchy affect the problem statement at lower levels; and (4) *design axioms*, which dictate that the independence of the functional requirements must be maintained and that the information content (i.e., cost, complexity, etc.) must be minimized as criteria for high-quality designs. More thorough explanations and detailed case-study examples showing the practical application of Axiomatic Design theory are available in [Suh, 1990], [Suh, 2000], and [Tate, 1999].

2.2 System Architecture Template and the Fractal Nature of Systems

For large systems, a system architecture can be developed in the Axiomatic Design framework that breaks down the design into individual subsystems at each level of the design hierarchy. In this representation, a system is modeled as a series of interacting inputs and outputs. These functions follow a pattern at every level of the design hierarchy, as they fall into one of the following categories: (1) *process functions*, the elements of the system that perform value-added tasks and activities; (2) *transport and interface functions,* the elements of the system that perform internal transportation and external interface tasks, (3) *command and control functions*, the logic that schedules and coordinates the process functions; and (4) *support and integration functions*, the support structure for the process and control functions, including pneumatics, mechanical structure, electronics, etc. [Hintersteiner, 1999]

By treating each level of the hierarchy as a system composed of subsystems, the same types of functional requirements appear at each 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 multiple subsystems that are incorporated into the hierarchy of the overall system. The details at each level are unique, but the general types of functions and interrelationships remain consistent. [Hintersteiner, 1999]

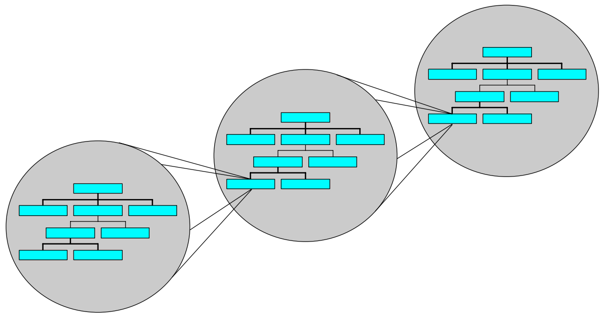

Figure 1: The concept of a fractal representation for systems. Each system is only one part of a larger system's hierarchy.

In the system architecture, the software for electromechanical systems is represented by means of command and control algorithms (CCAs) at each level of the design hierarchy. At each level, the CCA captures the logic of the interactions among the hardware elements at that level along with all of the communication protocols necessary to interact with its immediate parent and child CCAs. Thus, a software hierarchy emerges that mirrors the hardware hierarchy, through the software design is distributed and embedded within the hardware design. [Hintersteiner & Tate, 1998]

Because it is a system itself, the control software in complex systems follows the system architecture template. The software programs used to control systems (1) control the value-added tasks are performed, (2) interface with users (i.e., operators, maintenance engineers, or autonomous external computers), (3) coordinate the ordering and processing of commands and activities, and (4) recover from errors and perform other supporting tasks. Accordingly, it is possible to describe each CCA using the system architecture template, consisting of process programs, interfaces, control logic, and support and integration functions. [Hintersteiner & Nain, 1999]

2.3 Role of Constraints in System Design

Constraints (Cs) are defined as a specification of the characteristics that the design solution must possess to be acceptable to its customers and the company designing it. Constraints limit the set of acceptable design solutions (DPs) and influence the definition and scope of the functional requirements at lower levels of the design hierarchy.

Using error budgeting and allocation, a hierarchy of constraints is generated which parallels the hierarchy of FRs and DPs in the system architecture. At a particular level of the design hierarchy, the associated constraints are derived from the constraints applied to, and the design decisions made at, the parent level. Thus, as the designers evolve the decomposition into increasing levels of detail, the corresponding constraints become more refined and solution-specific. The advantage of this technique is that constraints are explicitly passed down from level to level in the system architecture. Hence, designers working on a particular subsystem have a clearly articulated set of constraints which they must follow. This minimizes the risk that important, though not necessarily obvious, constraints will be overlooked during the design process.

Constraints are generally classified into one of three categories: (1) *critical performance specifications* (C_P), which incorporate the specific values or thresholds on performance which must be met or exceeded for the system to be judged a success, (2) *interface constraints* (C_I), which define the interfaces between the system and its environment, as well as specifying operands, user interfaces and accessibility, safety issues, and so forth, and (3) *project constraints* (C_{PR}), which specify the resources available to the project, as well as issues related to backwards compatibility, imposed tradeoffs, and so forth. [Friedman *et al.*, 2000]

3 AD CAPABILITY MATURITY MODEL (AD CMM)

"The need to create sound syntheses and systemizations of knowledge… will call out a kind of scientific genius which hitherto has existed only as an aberration: The genius for integration. Of necessity, this means specialization, as all creative effort does, but this time… specializing in the construction of the whole."
-- Jose Ortega y Gasset (1944)

In the previous section, an overview was provided of the tools available for representing large-scale complex systems in the Axiomatic Design framework. By itself, however, this technical infrastructure is insufficient for implementing Axiomatic Design within an engineering organization. For success, an organizational infrastructure is also required. The key to this organizational infrastructure is a roadmap for understanding the organization's current level of capability in Axiomatic Design, as well as the direction in which the organization should be moving. Only with this type of roadmap in place can any coherent implementation plan be developed.

Researchers at CMU developed just such a roadmap for the Software Engineering and Systems Engineering professions, which they have dubbed the *improvement path for process capability.* This is shown in Figure 2. This roadmap consists of six levels of organizational process capability, ranging from not performed at all on an organizational level to a fully-developed standardized process that has quantitative metrics for performance and is continuously modified and improved. [Bate *et al.,* 1995]

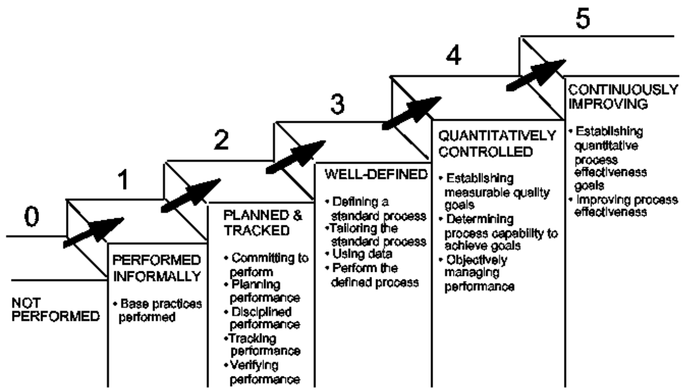
5
4
3
2
1
0
NOT PERFORMED
PERFORMED INFORMALLY
• Base practices performed
PLANNED & TRACKED
• Committing to perform
• Planning performance
• Disciplined performance
• Tracking performance
• Verifying performance
WELL-DEFINED
• Defining a standard process
• Tailoring the standard process
• Using data
• Perform the defined process
QUANTITATIVELY CONTROLLED
• Establishing measurable quality goals
• Determining process capability to achieve goals
• Objectively managing performance
CONTINUOUSLY IMPROVING
• Establishing quantitative process effectiveness goals
• Improving process effectiveness

Figure 2: The improvement path for process capability.

This generality of this model makes it readily applicable to the implementation of Axiomatic Design in the organization. In this case, the fundamentals of Axiomatic Design theory, the system architecture template, and any extensions to Axiomatic Design theory developed internally by the organization make up the "base practices" to be performed. The following list provides a detailed explanation of each maturity level for Axiomatic Design.

Level 0: Not Performed

This describes no performance at an organizational level, though individuals within the organization may be practicing Axiomatic Design for their own needs, independently of their required deliverables. Thus, there are no work products that can be readily identified or accessed. Performance of Axiomatic Design is obviously inconsistent and based entirely on individual knowledge and effort, and thus it is unlikely that a coherent set of base practices is being applied.

Level 1: Performed Informally

This level represents the beginning of organizational interest in Axiomatic Design. Typically, one or two small-scale projects may be selected to incorporate Axiomatic Design, in order to demonstrate what potential benefits, if any, can be gained. This helps to build knowledge of Axiomatic Design while minimizing the risk to the engineering programs, should the efforts fail. A high reliance must be placed on outside consultants and/or employed individuals with outside exposure to Axiomatic Design. Thus, the success or failure of the endeavor rests heavily on the knowledge, experience, and "force of personality" of these experts. Some of the base practices will be performed throughout the project as experience is gained by the team, though more are likely to be identified as useful as the project proceeds. There is likely to be a lack of consistent planning and tracking, as the organization is still learning and experimenting, and there is also likely to be great difficulty in transferring the experiences from one individual or team to another.

Level 2: Planned and Tracked

This level is the most significant in Axiomatic Design implementation, as it requires management commitment to incorporating Axiomatic Design into engineering processes. Resources, in terms of both dedicated staff and internal funding, must be provided to build up the necessary infrastructure. From a technical standpoint, this means building baseline system architectures for the key designs, in order to provide a starting point for design modifications and evaluations. From an organizational standpoint, an implementation plan must be formed, which includes determining how the system architecture is going to be used in the organization's Systems Engineering processes, developing the programs required to train engineers, and establishing the metrics to be used to determine the benefits that the organization is gaining by implementing Axiomatic Design.

Generally, the base practices will be performed on large-scale projects, where each project is responsible for its own planning and tracking. Thus, consistency will be achievable within a given project, though not necessarily across projects as the needs of each project and the level of detail that is required are still being determined. The metrics themselves will also be a mixture of qualitative and quantitative measures, based on what information is available as well as what information is perceived to be useful.

The results achieved at this level will be unique for each organization, based on both the corporate culture and the needs dictated by the organization's products and services. The Systems Engineering practices and metrics discussed in Section 4 show one concrete example of how Level 2 can potentially be achieved in an engineering organization.

Level 3: Well-Defined

This level is achieved when the "kinks" in the Axiomatic Design implementation have been worked out, and the base practices have been tailored to meet the organization's needs. At this point, standard operating procedures across the organization can be established based on the good and bad experiences of the projects. This includes well documented and approved procedures for the base practices that need to be achieved and the level of detail required. At this point, a critical mass of the engineering organization has been trained, allowing the processes to become self-sufficient without direct management intervention. Dedicated staff for each of the programs is still required, though at this point its responsibilities shift from being a driving force for implementation to advising and supporting the programs to ensure that the standard operating procedures are being followed.

Most engineering organizations implementing Axiomatic Design should be satisfied once Level 3 maturity is achieved. Reaching Level 4 or Level 5 maturity will evolve naturally in the organization, once the standards and procedures have been in place and used consistently over a long period of time.

Level 4: Quantitatively Controlled

Once sufficient data and knowledge exists, the qualitative metrics of the Axiomatic Design implementation will become *quantifiable*. Thus, a detailed, quantitative understanding of the Axiomatic Design implementation process is achievable, in terms of detailed performance measurements. Furthermore, the actual benefits achieved from using Axiomatic Design, based on whatever metrics the organization has deemed relevant, are comparable to a predictable performance based on past engineering programs.

Level 5: Continuously Improving

At the final level, the quantitative effectiveness metrics are well established and proven, and Axiomatic Design is fully integrated into engineering practices. At this level, dedicated Axiomatic Design staff and resources are no longer needed,[1] as Axiomatic Design is now performed reliably by everyone in the engineering organization, and the benefits from using Axiomatic Design are widely acknowledged. As the company itself evolves due to changing products and market conditions, minor changes and improvements to the Axiomatic Design standards and procedures may be necessary.

[1] A possible exception to this would be any staff and resources necessary to train new employees in Axiomatic Design. Conceivably, however, this task could be handled within the scope of other training programs, internal publications, and/or peer mentoring.

4 SYSTEMS ENGINEERING PRACTICES AND IMPLEMENTATION METRICS

"It must be remembered that there is nothing more difficult to plan, more doubtful of success, nor more dangerous to manage – than the creation of a new system."

– Machiavelli (1513)

As identified in the previous section, achieving Level 2 maturity is the most important step in Axiomatic Design implementation. Once this level of maturity is attained, the infrastructure and organizational momentum necessary to achieve Level 3 and higher are in place, and the work needed to generate the standards and quantitative metrics becomes relatively straightforward.

Achieving Level 2 maturity, however, is a very challenging endeavor, requiring both top-down support from management as well as bottom-up support from engineering. For management support, well-defined metrics must be established to show the benefits of using this approach to the bottom line, in terms of cost savings and improved product quality and customer satisfaction. For engineering support, a practical plan must be formed to define the ways Axiomatic Design is going to be used and controlled, as well as how engineers are going to be trained and rated on their Axiomatic Design performance.

This section outlines one potential plan that can be used to achieve Level 2 maturity, and is provided as a concrete though generalized example. This plan is included to serve as guiding principles for an engineering organization that has experimented with Axiomatic Design, achieved Level 1 maturity, and has undertaken the commitment to implement Axiomatic Design throughout its engineering organization. Of course, the implementation plan for each company is unique, as it must be based on several factors, including the business needs of the company, the types of products that are designed, the resources and expertise available, and the good and bad experiences that the company had while at Level 1.

The plan consists of three elements: Section 4.1 discusses the AD Systems Engineering practices that incorporates Axiomatic Design into Systems Engineering. Section 4.2 provides an overview of the importance of a good training and certification program. Section 4.3 outlines some potential metrics that can be used to rate the company's technical capability and business effectiveness in implementing Axiomatic Design.

4.1 AD SYSTEMS ENGINEERING PRACTICES

In order to understand how Axiomatic Design can be useful to Systems Engineering, it is necessary to understand what the goals of Systems Engineering are. According to the International Council on Systems Engineering (INCOSE), the mission of Systems Engineering is to "assure the fully integrated development and realization of products which meet stakeholders' expectations within cost, schedule, and risk constraints." [INCOSE & AIAA, 1997] Thus, Systems Engineering must take a "cradle-to-grave" approach, focusing on defining and documenting customer needs and required functionality, then proceeding with design synthesis and system validation. Consideration must be made to operations, performance, testing, manufacturing, cost, schedule, training, support, and disposal.

From the design perspective, therefore, the objective of Systems Engineering is to *control the design process* from concept generation through product delivery, in order to produce the optimum design solutions that will satisfy customer needs. This is shown graphically in Figure 3.

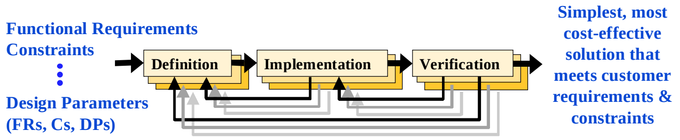

Figure 3: Overview of the Systems Engineering process flow.

Thus, to integrate Axiomatic Design into the Systems Engineering process, several key elements must be developed. First and foremost is the system architecture itself, which will be used to control the design and verification process. For each project, the system architecture can be decomposed to the module / subsystem level based on FRs, Cs, DPs, and the design matrices. Ultimately, this must be integrated with verification plans that define the testing and verification details for each FR and C, error budgets that allocate FRs and Cs from the top level to the level of relevant design activity, and external models (e.g. CAD, FEM, simulations, etc.) that characterize the design performance and provide the basis for acceptance of or resistance to new FRs and Cs. Furthermore, Axiomatic Design must also play a key role in formal design reviews, in order to demonstrate that an appropriate level of design maturity has been reached at key points in the program.

Successful integration requires appropriate management support as well as dedicated staff and resources. It cannot be the job of a core group of engineers to establish the system architecture for an entire program. Such a core group is not capable of keeping up with the daily changes to the design, nor are they capable of convincing the engineers that Axiomatic Design adds value to the design process.

Instead, a distributed organization is required where the engineers must have ownership and responsibility of their portion of the system architecture. To accomplish this, engineers must be trained and certified not only in the fundamentals of Axiomatic Design theory, but be given practical experience in how the organization uses and implements Axiomatic Design. In addition, the high-level system architecture must be generated early-on in a project to the point where the lower levels can be directly handed off to the responsible engineers. Systems Engineering must have this responsibility, and as the project proceeds, they must also assist the design engineers to ensure that the detailed levels of the system architecture remain consistent with the design assumptions made at higher levels. Furthermore, Systems Engineering must work directly with project managers to reconcile the ramifications of lower-level design decisions on coupling issues and other design inconsistencies. An example of this distributed organizational concept is shown graphically in Figure 4. In this example, the organizational structure is by major project, and high-level oversight of the Axiomatic Design efforts on each program is included to ensure cross-program consistency.

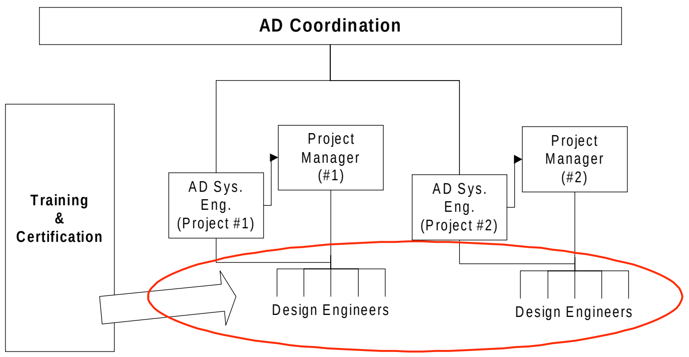

Figure 4: Organization required for effective implementation of Axiomatic Design.

Accordingly, the system architecture construction itself is distributed amongst several individuals, as is shown in Figure 5. To keep this up-to-date, an electronic platform to maintain and distribute the system architecture is extremely useful. Several potential tools are available, including commercial databases and spreadsheets. Acclaro™, developed by Axiomatic Design Software, Inc., also shows great potential for use as a distribution and storage system for the system architecture.

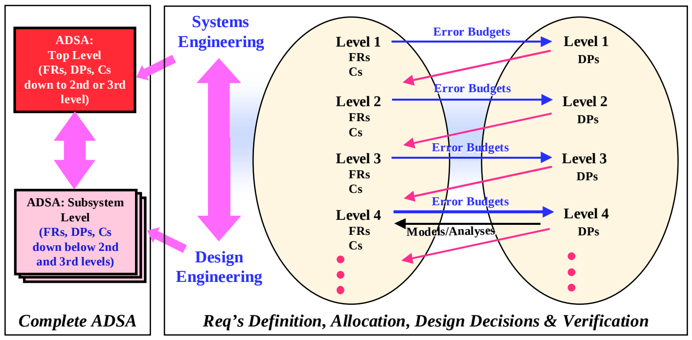
ADSA: Top Level (FRs, DPs, Cs down to 2nd or 3rd level)
ADSA: Subsystem Level (FRs, DPs, Cs down below 2nd and 3rd levels)
Complete ADSA
Systems Engineering
Design Engineering
Level 1 FRs Cs
Level 2 FRs Cs
Level 3 FRs Cs
Level 4 FRs Cs
Level 1 DPs
Level 2 DPs
Level 3 DPs
Level 4 DPs
Error Budgets
Models/Analyses
Req's Definition, Allocation, Design Decisions & Verification

Figure 5: Distributed organization of the Axiomatic Design system architecture.

The detailed Systems Engineering process flow for design definition, incorporating the distributed system architecture, is shown in Figure 6. For an evolutionary design, the process starts with the system architecture from the previous generation, where the new changes and additions to the FRs and Cs at the top-level are incorporated. These changes and additions are then mapped into the error budgets, models, and system design specifications, so as to be trickled down to lower levels of the system architecture. This process leads to shorter design times, and ultimately a shorter time to market.

It is important for a company that is serious about implementing Axiomatic Design in its Systems Engineering practices to be involved in the Axiomatic Design community. This includes publication in journals and conferences as well developing exchanges with other companies and academic institutions.

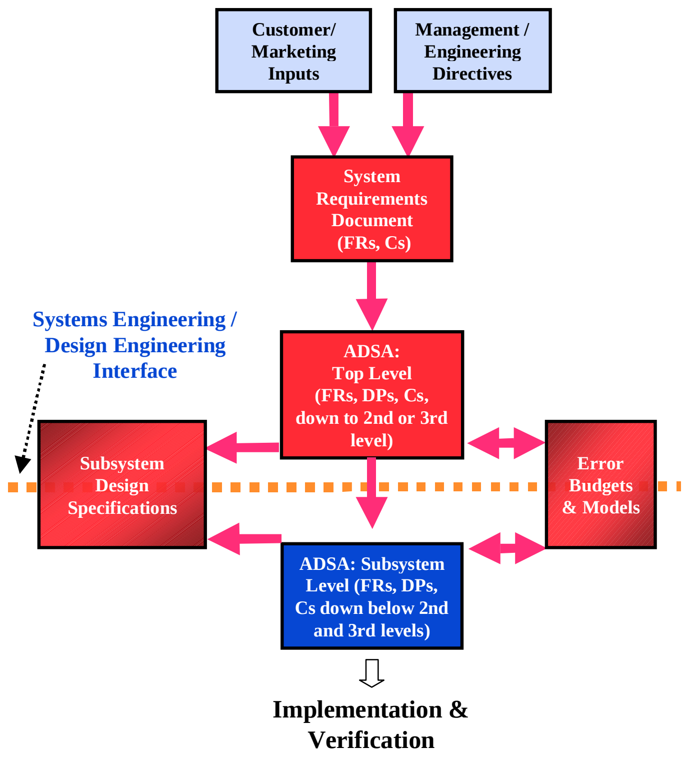
Customer/ Marketing Inputs
Management / Engineering Directives
System Requirements Document (FRs, Cs)
Systems Engineering / Design Engineering Interface
ADSA: Top Level (FRs, DPs, Cs, down to 2nd or 3rd level)
Subsystem Design Specifications
Error Budgets & Models
ADSA: Subsystem Level (FRs, DPs, Cs down below 2nd and 3rd levels)
Implementation & Verification

Figure 6: Detailed systems engineering process flow incorporating the Axiomatic Design system architecture.

4.2 Training and Certification Program

When a company is at Level 1, outside consultants are usually employed to teach the fundamental concepts of Axiomatic Design to engineers and to work with engineers on specific small-scale projects. This approach, however, is not sufficient for teaching large numbers of engineers to perform Axiomatic Design with sufficient *consistency* to enable them to work within the AD Systems Engineering practices. As discussed above, the design engineers themselves are responsible for providing their portions of the system architecture. Thus, a full system architecture has multiple authors, though to be useful it must maintain consistency at all levels of the hierarchy. Furthermore, the decision to maintain the system architecture electronically requires engineers to learn both the software tools as well as Axiomatic Design methodology. Accordingly, engineers must be trained and demonstrate mastery in the following skills:

- ***Fundamental AD theory*** – includes understanding FRs, DPs, PVs, constraints, design matrices, design axioms, decomposition, and constraint refinement
- ***Issues in Coupling*** – includes distinguishing between functional and physical coupling, qualitative issues in the traceability of requirements, and quantitative issues in sensitivity analyses
- ***System Architecture Template*** – includes representing systems, understanding the distinctions between process, transport/interface, control, and support/integration functionality, and understanding how the baseline system architectures for the major products are organized and managed
- ***Software Tools*** – includes entering information, applying to design analyses, and integrating into large-scale projects

Furthermore, classroom instruction is not sufficient for learning and mastering this material – practical application of these techniques are required. Accordingly, each engineer should be required to apply what they have learned to their work, and then present that application to their peers in a design review or company-wide Axiomatic Design forum. Engineers who demonstrate suitable proficiency with the material should be certified. The certification guidelines should be based on the company's internal needs as well as the knowledge standards recognized by the Axiomatic Design community.

4.3 AD Implementation Metrics

Ultimately, organizations are not only interested in how they are implementing Axiomatic Design, but in how well they are implementing it and what benefits they are receiving from this undertaking. Of course, the old adage applies – you are what you measure. The perceived quality of the implementation and the improvements made to it over time are a direct result of what metrics are applied and what activities are rewarded and encouraged. Accordingly, the metrics used must be chosen very carefully, to provide data that (a) is relatively easy to directly collect or indirectly ascertain, and (b) reflects useful information that can be used for future improvement. This is not a static process – as the organization's maturity with Axiomatic Design grows and the necessary knowledge and data is developed, metrics may be changed or added. Two example sets of metrics that can be applied are as follows. This is not necessarily a complete list, as organizations may decide they wish to track different factors based on their particular needs.

- *Capability metrics* provide a measure of the organization's technical performance in implementing Axiomatic Design, and include such elements as:
  1. the number of engineers who are trained and certified,
  2. the quality and reusability of developed subsystems,
  3. the overall quality of the system architecture in terms of the scope, level of decomposition, detail, and up-to-date information, and
  4. the level of participation in the external Axiomatic Design community.

- *Effectiveness metrics* provide a measure of the benefits to the organization's business practices as a result of implementing Axiomatic Design, and include such elements as:
  1. the increase in understanding of the design responsibilities by all parties early-on in the project,
  2. the reduction in time to market, with respect to the shrinkage in the number of design errors, rework efforts, engineering changes, and design review meetings, and
  3. the increase in product quality and customer satisfaction, in terms of how well the FRs and Cs are satisfied by the final product.

Note that while some of these metrics may include quantitative values, most are more qualitative, or are based on comparing the current performance with past performance on other programs. Such metrics are natural at Level 2 maturity, though as this data is collected and interpreted, more refined and quantitative metrics can be developed.

5 CONCLUSIONS

"Conceptual integrity is the most important consideration in system design."
– Fredrick P. Brooks, Jr. (1972)

Successful implementation of Axiomatic Design within an engineering organization requires understanding both the technical and managerial issues involved. Knowledge of Axiomatic Design principles is necessary, but standard operating procedures and metrics must be implemented in order to obtain both management and engineering support.

To this end, an Axiomatic Design Capability Maturity Model has been developed, to provide a clear roadmap for implementing Axiomatic Design. Using this roadmap, Systems Engineering practices can be established, along with custom training and certification programs. Furthermore, metrics can be developed to evaluate the success and utility of the implementation. Such an infrastructure enables an organization to successfully incorporate Axiomatic Design into its engineering practices, and reap the rewards that Axiomatic Design has the potential to offer.

6 ACKNOWLEDGMENTS

The authors would like to acknowledge the many engineers and managers at SVG Lithography Systems, Inc. who have provided us with invaluable feedback and support in our company's recent efforts to achieve Level 2 maturity in Axiomatic Design. Most notably, our thanks go to Dan Cote, VP of Engineering, for providing the tools, resources, and management support necessary for this effort.

We would also like to acknowledge several others, including Dr. Derrick Tate of Axiomatic Design Software, Inc., Professor Christopher Brown of the Worcester Polytechnic Institute, and Professor Nam P. Suh of the Massachusetts Institute of Technology for their insights into teaching Axiomatic Design to engineers and their detailed knowledge of Axiomatic Design theory.

7 ABOUT THE AUTHORS

Jason D. Hintersteiner received a Bachelor of Science and Master of Science in Mechanical Engineering at the Massachusetts Institute of Technology. His experience includes research projects in robotics, digital control, error modeling and compensation, computer networking, as well as two years of postgraduate work with Professor Nam P. Suh at MIT on the application of Axiomatic Design to large-scale systems. He is currently a Senior Staff Engineer at SVG Lithography Systems, Inc., and is chiefly responsible for coordinating the implementation of Axiomatic Design throughout the engineering organization.

Richard C. Zimmerman received a Bachelor of Science in Physics and a Bachelor of Arts in Mathematics from the University of Illinois, and a Master of Science in Optics at the University of Southern California. He has an extensive systems engineering background, stemming largely from his work on large-scale state-of-the-art optical systems, including high energy lasers and the Chandra x-ray telescope. He is currently the Director of Systems Engineering at SVG Lithography Systems, Inc.

8 REFERENCES

[Bate *et al.,* 1995] Bate, R., Kuhn, D., Wells, C., *et al. A Systems Engineering Capability Maturity Model*, Version 1.1. CMU Software Engineering Institute. November 1995. Document #: SECMM-95-01, CMU/SEI-95-MM-003.

[DelPuerto and Garcia, 2000] DelPuerto, S. and Garcia, J. "Axiomatic Redesign of Guide Flexures: Improving Product Reliability and Reducing Manufacturing Cost." *First International Conference on Axiomatic Design,* June 21-23, 2000, Cambridge, MA USA.

[Fredriksson *et al.,* 1994] Fredriksson, B., Killander, A. , and Nordlund, M. "An Effective Model of Transferring New Methods from Academia to Industry", *Second CIRP International Workshop on Design Theory and Methodology*, Stockholm, Sweden, pp. 57-67, June 16-17, 1994.

[Friedman *et al.,* 2000] Friedman, G., Hintersteiner, J. D., Tate, D., and Zimmerman, R. (2000). "Representation of Constraints in the System Architecture." *Fifth World Conference on Integrated Design and Process Technology*, June 4-8, 2000, Dallas, TX USA.

[Hintersteiner, 1999] Hintersteiner J. D., "A Fractal Representation for Systems", *International CIRP Design Seminar*, Enschede, the Netherlands, March 24-26, 1999.

[Hintersteiner, 2000] Hintersteiner J. D., "Addressing Changing Customer Needs by Adapting Design Requirements", *To be published.*

[Hintersteiner and Nain, 1999] Hintersteiner J. D., Nain A.S., "Integrating Software into Systems: An Axiomatic Design Approach", *Third International Conference on Engineering Design and Automation*, Vancouver, B. C. Canada, August 1-4, 1999.

[Hintersteiner and Tate, 1998] Hintersteiner J. D., Tate D., "Command and Control in Axiomatic Design Theory: Its Role and Placement in the System Architecture", *Second International Conference on Engineering Design and Automation*, Maui, HI, Aug. 9-12, 1998.

[INCOSE & AIAA, 1997] INCOSE & AIAA Sys. Eng. Tech. Comm. "Systems Engineering: A Way of Thinking, A Way of Doing Business, Enabling Organized Transition from Need to Product." http://www.incose.org/, August 1997.

[Leveson, 1995] Leveson N.G., *Safeware: System Safety and Computers*, Reading, MA: Addison-Wesley, 1995. ISBN 020-111972-2

[McDonald, 1998] McDonald, C. "Paying Attention to Professor Pays Off", *WPI Wire,* Vol. 12, Num. 1, June 1998.

[Nordlund *et al*., 1996] Nordlund M., Tate D., & Suh N. P., "Growth of Axiomatic Design through Industrial Practice", *3rd CIRP Workshop on Design and the Implementation of Intelligent Manufacturing Systems*, Tokyo, Japan, pp. 77-84, June 19-21, 1996.

[Suh, 1990] Suh N. P., *The Principles of Design*, New York: Oxford University Press, 1990. ISBN 0-19-504345-6

[Suh, 2000] Suh N. P., *Axiomatic Design: Advances and Applications*, New York: Oxford University Press, 2000. To be published.

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