Cite As: A. M. Farid, "An engineering systems introduction to axiomatic design," in Axiomatic Design in Large Systems: Complex Products, Buildings & Manufacturing Systems (A. M. Farid and N. P. Suh, eds.), ch. 1, pp. 1–47, Berlin, Heidelberg: Springer, 2016.

An Engineering Systems Introduction to Axiomatic Design

Amro M. Farid, *Senior Member, IEEE,*

***Abstract*—Since its first publication in 1978, Axiomatic Design has developed to become one of the more commonly applied engineering design theories in the academic literature and industrial practice. In parallel, model-based systems engineering (MBSE) has developed from industrial origins in the aerospace, communications and defense sectors. As the scope of humanity's engineering efforts grows to include ever-more complex engineering systems, the engineering design methodologies that guide these efforts must also develop. These two, now well-established but independently developed, engineering design methodologies now appear well poised to support the synthesis, analysis, and re-synthesis of large complex engineering systems. As the first chapter in this book on the application of Axiomatic Design to Large Complex Systems, it introduces the fundamentals of Axiomatic Design within the context of engineering systems and as a conceptual foundation for subsequent chapters. It also relates Axiomatic Design's key concepts and terminology to those found in current model-based systems engineering techniques including SysML. The chapter concludes with applications in which Axiomatic Design has served to advance the development of engineering systems including quantitative measures of life cycle properties, design of cyber-physical systems, and design of hetero-functional networks.**

***Index Terms*—axiomatic design, large fixed engineering systems, large flexible engineering systems, graph theory, life cycle properties, resilience, reconfigurability, systems engineering, model-based systems engineering, system architecture, MBSE**

I. Introduction

A. The Evolution of Axiomatic Design

Since its first publication in 1978 [1], Axiomatic Design [2], [3] has developed to become one of the more commonly applied engineering design theories in the academic literature and industrial practice [4]. It arose from the need to make the field of design more of a science rather than an art [2], [3]. The originator of Axiomatic Design, Prof. Nam P. Suh, believed from his own practical experience as a designer that if design curriculum had a more solid theoretical foundation then a new generation of engineering designers could be trained to make more effective products and systems in less time and at lower cost. Consequently, Axiomatic Design's most distinguishing characteristic is the use of design axioms which guide the designer through the engineering design process. From these axioms, many theorems and corollaries have been subsequently proven [2], [3]. This theoretical foundation facilitated many subsequent academic works in engineering design [5], [6] without diminishing the practical application of engineering design in industry [4]. In the beginning, Axiomatic Design found applications within Suh's home field: mechanical engineering of products [2]. Since then, Axiomatic Design has expanded to many other disciplines including software and more generally large complex systems in the 21st century [7]–[9]. This successful expansion into new design applications of ever larger system scale has suggested a degree of universality to Axiomatic Design as a theory.

Amro M. Farid: Thayer School of Engineering, Dartmouth College, 14 Engineering Drive, Hanover NH 03755. Mechanical Engineering, Massachusetts Institute of Technology, 77 Massachusetts Avenue Cambridge, MA 02139, USA. Email:amfarid@dartmouth.edu,amfarid@mit.edu

B. The Evolution of Model-Based Systems Engineering

Meanwhile, the modern systems engineering field developed methodologically from industrial origins in the aerospace, communications, and defense sectors [11].

Definition 1. Systems Engineering [12]: An interdisciplinary approach and means to enable the realization of successful systems. It focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, and then proceeding with design synthesis and system validation while considering the complete problem: operations, cost and schedule, performance, training and support, test, manufacturing, and disposal. SE considers both the business and the technical needs of all customers with the goal of providing a quality product that meets the user needs.

Here, the focus is on complex product and systems where many teams of engineers have to integrate their efforts on complex products from first conception to final decommissioning (i.e. "birth to death") [12]. To support this emerging field, the International Council on Systems Engineering (INCOSE) was founded as a professional organization to develop and disseminate the practice of systems engineering [13]. Consequently, many academic departments were founded [13] and along with several archival journals [14], [15]. INCOSE has also sought to standardize systems engineering knowledge to improve communication and "interoperability" between practitioners [11], [12].

One important aspect of this activity has been the trend towards model-based systems engineering.

Definition 2. Model-based systems engineering (MBSE) [12]:the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases.

While Wayne Wymore is often credited with introducing a mathematical foundation for MBSE [16], much of its development arose only recently from the need to manage system complexity as physical systems integrated more and

TABLE I
A CLASSIFICATION OF ENGINEERING SYSTEMS BY FUNCTION AND OPERAND [10]

Function/Operand	Living Organisms	Matter	Energy	Information	Money
Transform	Hospital	Blast Furnace	Engine, electric motor	Analytic engine, calculator	Bureau of Printing & Engraving
Transport	Car, Airplane, Train	Truck, train, car, airplane	Electricity grid	Cables, radio, telephone, and internet	Banking Fedwire and Swift transfer systems
Store	Farm, Apartment Complex	Warehouse	Battery, flywheel, capacitor	Magnetic tape & disk, book	U.S. Buillon Repository (Fort Knox)
Exchange	Cattle auction, (illegal) human trafficking	eBay trading system	Energy market	World Wide Web, Wikipedia	London Stock Exchange
Control	U.S. Constitution & laws	National Highway Traffic Safety Administration	Nuclear Regulatory Commission	Internet engineering task force	United States Federal Reserve

more control, automation, and information technology [17]. It may be viewed as a trend away from a "document-centric approach" to systems engineering towards a "model-centric" approach integrated into all systems engineering processes [17]. At the heart of this initiative has been the development of several modeling standards, most notably the Systems Modeling Language (SysML) [18], [19]. While these are primarily graphical in nature, they directly support the integration of quantitative models.

C. The Emergence of Engineering Systems

The maturation of Axiomatic Design and MBSE as engineering design methodologies and theories into the arena of large complex systems is timely. In the 20^{th} century, individual technology products like the generator, telephone, and automobile were connected to form many of the large scale infrastructure networks we know today: the power grid, the communication infrastructure, and the transportation system [10]. Over time, these networked systems developed even more interactions while continuing to incorporate many new technology artifacts (e.g. renewable energy, smart phones, & electric vehicles). Naturally, this meant greater complexity, not just because of the greater interaction within these systems, but also because of the presence of an expanding heterogeneity of functionality. Furthermore, these already large scale, complex, network systems began to develop interactions between themselves in what is now called *systems-of-systems* [20], [21]. The "smart grid" [22], the energy-water nexus [23], the electrification of transport [24] are all good examples where one network system has fused with another to form a new and much more capable system. This trend is only set to continue. The energy-water-food nexus [25] fuses three such systems and the recent interest in smart cities [26] provides a platform upon which to integrate all of these efforts. This work classifies such systems as engineering systems:

Definition 3. Engineering system [10]: A class of systems characterized by a high degree of technical complexity, social intricacy, and elaborate processes, aimed at fulfilling important functions in society.

As engineering systems have evolved so too must the role of the engineer within them [10]. The scope of engineering systems, and in particular systems-of-systems, often spans the traditional borders of individual engineering disciplines (e.g. mechanical, electrical, civil, chemical). Furthermore, as engineering systems become ever more ubiquitous and intertwined with daily life the requirements that they must fulfill also grow and diversify. Therefore, engineering systems should not be viewed in terms of cost and quality of function alone but also include a full taxonomy of system requirements (See Figure 4). These "requirements" are not just of the traditional type where a single client contractually expects specific line items from the engineer; rather in engineering systems requirements also take the form of policies, regulations, and standards where engineers are one of many public and private stakeholders that help to shape the planning and operation of the engineering system in the present and the future [10]. Engineering design methodologies and theories, at their current stage of development, and when interpreted formally and strictly, are likely inadequate for engineering systems. However, they are likely to provide the mental constructs and models that may serve as foundations for coherent methodological developments.

D. Classification and Characterization of Engineering Systems

The challenge of developing consistent methodological foundations for engineering systems is formidable. Consider the engineering systems taxonomy presented in Table I [10]. It classifies engineering systems by five generic functions that fulfill human needs: 1.) transform 2.) transport 3.) store, 4.) exchange, and 5.) control. On another axis, it classifies them by their operands: 1.) living organisms (including people), 2.) matter, 3.) energy, 4.) information, 5.) money. This classification presents a broad array of engineering domains that must be consistently treated. Furthermore, these engineering systems are at various stages of development and will continue to do so for decades, if not centuries. And so the field of engineering systems must equally support design synthesis, analysis, and re-synthesis while supporting innovation; be it incremental or disruptive. Axiomatic Design and MBSE present themselves as promising engineering design methodologies and theories with the flexibility to address the breadth of different engineering systems.

E. Methodological Challenges in Engineering Systems

Across the broad array of engineering systems applications, several important and recurring themes have emerged

as methodological challenges. One of these is the required attention to life cycle properties or "ilities" [10].

Definition 4. Life Cycle Properties ("ilities") [10]: Desired properties of systems, such as flexibility or maintainability (usually but not always ending in "ility"), that often manifest themselves after a system has been put to its initial use. These properties are not the primary functional requirements of a system's performance, but typically concern wider system impacts with respect to time and stakeholders than are embodied in those primary functional requirements. The "ilities" do not include factors that are always present, including size and weight (even if these are described using a word that ends in "ility").

Life cycle properties usually have an emergent nature that can not be predicted from individual system components. Therefore, understanding the factors that enable these properties is fundamental to engineering systems as they develop over many years.

A second engineering systems challenge is in their cyber-physical nature [10], [11]. Engineering systems, as expected, are largely physical in order to realize their important primary function. In the meantime, by virtue of their size and complexity, they require many decision-making components be they human or automated control. Designing, planning, and controlling such large-scale cyber-physical systems goes well-beyond traditional control theory research. It now includes more fundamental questions that balance centralization vs distribution, automation vs human decisions, and authority versus cooperative negotiation bounded within a context of human stakeholders and actors.

Finally, a third engineering systems challenge is managing the integration of hetero-functional systems-of-systems. As mentioned in Section I-C, well-known engineering systems such as those that deliver electricity, information, natural gas, water, transportation, and healthcare are fusing [10], [27]. In the meantime, engineering education remains organized into departments along these well-established and often self-reinforcing silos [10]. Very few universities prepare engineers to span two or more integrated engineering systems; even fewer do so while addressing the fundamental questions into life cycle properties and cyber-physical systems. Efforts to address these three challenges requires a methodological base founded within engineering design methodologies and theories such as Axiomatic Design and MBSE.

F. Contribution

As the first chapter in this book on the application of Axiomatic Design to large complex systems, it seeks to introduce the fundamentals of Axiomatic Design as a conceptual foundation for subsequent chapters. These include complex products, buildings, and manufacturing systems. As a group, they contain many of the same challenges found in other application domains for systems research. Therefore, the chapter also seeks to relate Axiomatic Design's key concepts to those found in current MBSE techniques including SysML. As the discussion is of an introductory nature, the chapter draws heavily from several well established texts in Axiomatic Design [2], [3], MBSE [18], [19], [28], [29], and engineering systems [10]. It, also, seeks to clarify nuances within and between these texts that can cause confusion or misinterpretation. Finally, the chapter returns to the engineering systems discussion provided in this introduction. It concludes with directions in which Axiomatic Design has served to address the methodological challenges facing engineering systems today.

G. Chapter Outline

The remainder of the chapter proceeds as follows. Section II introduces Axiomatic Design and its relationship to MBSE in terms of four domains of engineering design: stakeholder requirements, functional architecture, physical architecture, and process domains. Next, Section III focuses specifically on the design synthesis and analysis of the allocated architecture as the mapping between the functional and physical architectures. Next, Section IV discusses the relationship between these domains with a focus on Axiomatic Design's Independence & Information Axioms. Section V goes on to address how Axiomatic Design manages the complexity of systems via a dual functional and physical system hierarchy. Section VI then highlights potential applications of Axiomatic Design in the development of engineering systems. The chapter is brought to a close in Section VII.

II. Four Domains in the Engineering Design of Systems

From an Axiomatic Design perspective, the engineering design of systems consists of four domains. They are defined here as follows drawing upon consistent definitions from both the MBSE and Axiomatic Design literature.

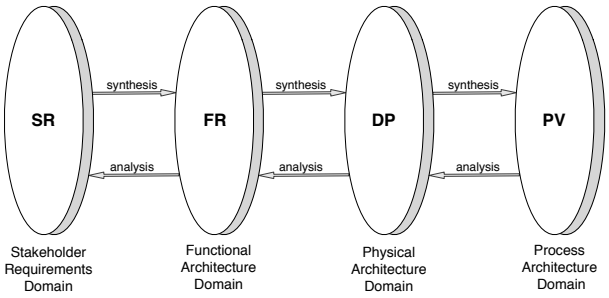


Fig. 1. Four Domains in the Engineering Design of Systems – An Axiomatic Design Perspective (Adapted from [3])

Definition 5. Stakeholder Requirements Domain [18]: a collection of statements that describe the system properties and behaviors that all stakeholders need to be met.

Definition 6. Functional Architecture Domain [28] – a logical model of a functional decomposition plus the flow of inputs and outputs to which input/output requirements have been traced. It constitutes the system behavior or function.

Definition 7. Physical Architecture Domain [28], [30] – the components of a system and the relationships amongst them. It constitutes the system form.

Definition 8. Process Architecture Domain [3]: the set of processes and their relationships that characterize how the physical architecture is generated or produced.

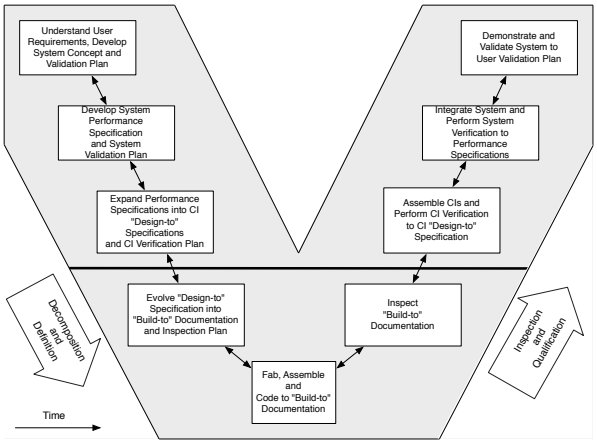


Fig. 2. Traditional Top Down Systems Engineering "Vee"(Adapted From [28], [31])

These four domains are sequentially mapped one onto the next as shown in Figure 1. Motion from left to right represents an engineer's synthesis activity from "what needs to be achieved" to "how it is to be achieved" [3]. Motion from right to left represents an engineer's analysis activity which supports validation and verification. The first three of these domains are consonant with the 'vee" in traditional top-down systems engineering depicted in Figure 2 [28], [31]. Figure 1's synthesis path is consonant with Figure 2's left half describing decomposition and definition. Meanwhile, Figure 1's analysis path is consonant with Figure 2's right half describing integration and qualification. Furthermore, Figure 3 shows that the SysML language supports the engineering design domains with three classes of diagrams: requirements, behavior, and structure. Each of these domains is now described in turn.

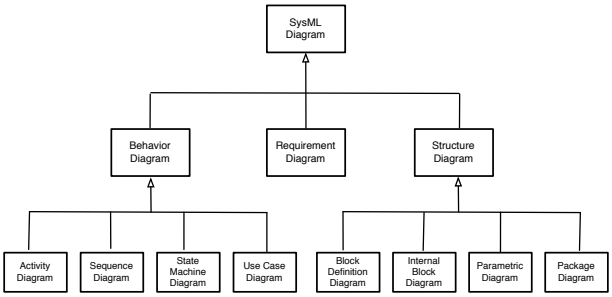


Fig. 3. Taxonomy of SysML Diagrams (Adapted From [18], [19])

A. Stakeholder Requirements Domain

In Axiomatic Design, the stakeholder requirements domain is more often called the customer domain in recognition of AD's roots in the engineering design of products with customers as sole stakeholders. The elements of the domain are called customer needs $\mathbf{CN}$ [3]. Here, the term stakeholder requirements domain is used instead to address engineering systems' multiple stakeholders. Similarly, the domain is populated with stakeholder requirements $\mathbf{SR}$ [28].

Definition 9. Stakeholder Requirements [28]: Statements by the stakeholders about the system's capabilities that define the constraints and performance parameters within which the system is to be designed. These stakeholders' requirements focus on the boundary of the system in the context of these mission requirements, are written in the stakeholders' language, are produced in conjunction with the stakeholders of the system, and are based upon the operational needs of these stakeholders.

The main challenge with the stakeholder requirements domain is that the "voice" of the stakeholder is not that of the engineer. Therefore, the engineer must work with all stakeholders to determine a complete set of stakeholder requirements [3], [28]. These are then used to "translate" and derive a set of system requirements $\mathcal{SR}$ in an engineering language.

Definition 10. System Requirements [28]: A translation (or derivation) of the originating requirements into engineering terminology.

This requirements engineering process is usually completed before the rest of the synthesis path in Figure 1 can continue. That said, subsystem and component requirements may be derived at a later stage to support internal delegation or external subcontracting [28]. The interested reader is referred to several dedicated texts on requirements engineering [32]–[35]. Buede provides a relatively concise treatment that consists of seven steps [28]:

1) Develop the operational concept
2) Define the system boundary
3) Develop the system objectives hierarchy
4) Develop, analyze, and refine requirements (stakeholders' and system)
5) Ensure requirements feasibility
6) Define the qualification system requirements
7) Obtain approval of system documentation

Definition 11. Operational Concept [28]: A vision for what the system is (in general terms), a statement of mission requirements, and a description of how the system will be used. The shared vision is based on the perspective of the system's stakeholders of how the system will be developed, produced, deployed, trained, operated and maintained, refined, and retired to overcome some operational problem and achieve the stakeholders' operational needs and objectives. The mission requirements are stated in terms of measures of effectiveness. The operational concept includes a collection of scenarios (one or more for each group of stakeholders in each relevant phase of the system's life cycle).

Definition 12. Objectives Hierarchy [28]: A hierarchy of objectives that are important to the system's stakeholders in a value sense; that is, the stakeholders would (should) be willing to pay to obtain increased performance (or decreased cost) in any one of these objectives. It is also the definition of the natural subsets of the fundamental objective into a collection of performance requirements.

Stakeholder and system requirements may be classified as shown in Figure 4. Using a requirements classification structure serves to organize requirements (especially in large systems) so as to avoid conflicts and/or duplication. The system requirements include the functional requirements as a subset and appear later in the functional architecture domain (Section II-B). Note that in Axiomatic Design, this requirements taxonomy is traditionally reduced to customer

needs **CN**, functional requirements **FR**, and constraints **C**. More recently, Thompson adds non-functional requirements **nFR**, selection criteria **SC**, and optimization criteria **OC** to the taxonomy [36] and highlights common errors in their misclassification [37].

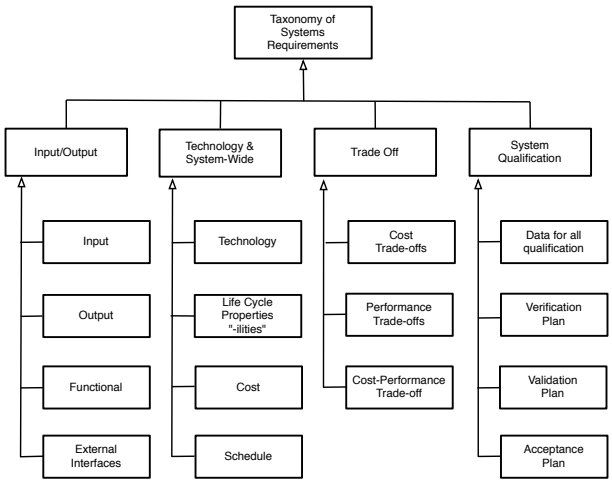


Fig. 4. Requirements Classification in Model Based Systems Engineering (Derived from [28])

Requirements engineering documention is well supported by the requirements diagram in SysML [18] (See Figure 5). Derived requirements serve during the synthesis path as the primary relationship within the requirements domain up to and including the functional requirements. This is similar to the "House of Quality" in Quality Function Deployment methodologies [38]–[40]. During the analysis path, the "verified relationship" links requirements to functions (in test cases) and the "satisfied relationship" links requirements to components.

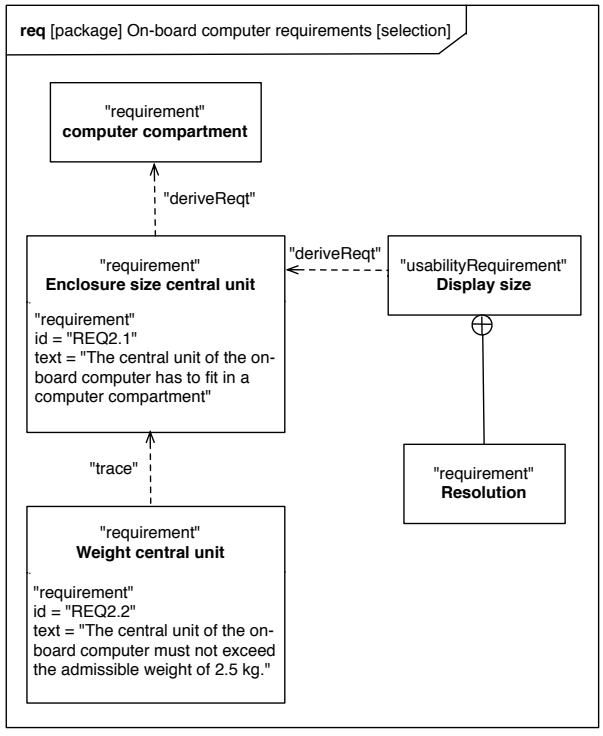


Fig. 5. An Example Requirements Diagram in SysML

As expected, the stakeholder requirements domain is primarily described in text with some numerical specifications. It is only after several steps of engineering synthesis and modeling can more mathematical treatments begin to be applied.

B. Functional Architecture Domain

Most engineering design methodologies and theories include some form of functional architecture domain [4], [29]. As shown in Figure 6, it consists of functions that are arranged in serial or in parallel and may be nested into hierarchies. As described in Section I-D, these functions may transform, transport, store, exchange or control their operands which in turn may be classified as material, energy, information, money, or people [10]. By convention, each function is defined as a transitive verb stated in the third person singular followed by its associated object/operand. It must also be defined in a solution-neutral way that doesn't presuppose the technologies within the physical architecture [3], [28].

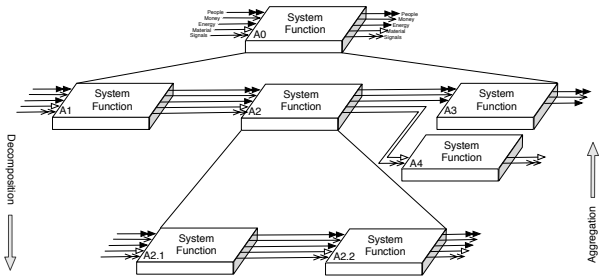


Fig. 6. A Functional Architecture with Parallel, Serial, and Nested Interactions

There is considerable variation in nomenclature across the Axiomatic Design and MBSE literature in regards to the elements of functional architecture domain. Generically, they are called functions. In Axiomatic Design, the functions are instead called *functional requirements*. Both of these conventions are used in this chapter. The AD convention serves several logical purposes. First, it emphasizes that what the system must do, its system function, is not defined in a vacuum but rather is the logical consequence of the stakeholder requirements identified previously. Second, it recognizes that functional requirements viewed at a high level are just as binding as when they are decomposed to a lower level. Third, in not distinguishing between a functional requirement and a function, Axiomatic Design is not distinguishing between what the system must do and what the system does (as designed). After all, they should be the same. In contrast, Buede considers that functional requirements have a dual purpose: first as a subclass of the system requirements and second as (only) the top level of the functional architecture [28]. This serves to link the two domains with common elements but distinguish between functional requirements and the rest of the system functionality. To complicate matters, the term "process" is sometimes used in place of function in MBSE [10], [27], [29], [41], [42]. While this practice is common, it ultimately confuses the differences between the functional architecture domain and the process architecture domain in the Axiomatic Design framework in Figure 1.

Generation of the functional architecture is very much a synthetic – "forward-engineering" – activity. In order to be successful, it requires that the set of functions be mutually exclusive and collectively exhaustive [3]. There is a general consensus in engineering design that overlapping system functions will cause downstream design errors [3], [28]. Meanwhile, generating an exhaustive set of functions begins from the previously defined operational concept and high level functional requirements [28]. Several notable functionality templates have been developed over the years to spur designer creativity and prevent unintentional omission of functionality [43]–[45]. Again, the identification of solution-neutral functions supports maximally innovative physical designs downstream [3].

That said, it is not uncommon to generate the functional architecture as an analytical – "reverse-engineering" activity. Often, in an engineering systems context, a part or a whole of the system has already been built [10] and the development of the functional architecture is required to determine how "evolve" the system to stage of development. Furthermore, well-known functions have tried-and-true physical solutions which may be reimplemented successfully as part of design patterns or in novel configurations. It is often unnecessary to "reinvent the wheel". Therefore, using a reverse engineering analytical mindset, functions can be identified as an abstraction of existing components in the physical architecture. For example, I-beams in buildings support weight, and railways transport trains. In reality, however, the functional architecture, at its multiple levels of decomposition, must be developed in parallel with the physical architecture via the allocated architecture [3], [28] and will be discussed in detail from an Axiomatic Design perspective over several sections.

The development of the functional architecture is well supported in SysML [18], [19]. As shown in Figure 3, this includes four diagrams that provide complementary views of the overall system behavior at different levels of engineering design detail. These include activity, state machine, use case, and sequence diagrams. While all of these are useful in detailed design, the first three have common applications in conceptual design prior to the synthesis of the physical architecture. These include:

- **Activity diagrams** – support general purpose functional-modeling that closely resemble Figure 6. Functions in this diagram are called actions.
- **State machine diagrams** – support the organization of functionalities into modes of operation. Functions in this diagram are implicit to what happens during a particular operating state.
- **Use case diagrams** – support the interactions with external entities such as people and organizations. Functions in this diagram are called use cases.

While each of these diagrams have formal semantics, their appropriate use is often daunting for novice systems engineers beginning a conceptual design. As a partial alternative, the object process methodology supports system function models in a manner that resembles simplified activity diagrams [41], [42]. In both models, it is important to distinguish the flow of power from the other types of operands. This is because the directionality of power flow does not fully coincide with the direction of *dynamic causality* [46]. For example, a voltage source will impose a voltage on downstream functions (e.g. loads) but they in return (e.g. by their impedance) will impose the required current.

The functional architecture domain also very much lends itself to mathematical description. From a static perspective, functional elements (at any given level of abstraction) can be organized into a directed graph and its associated adjacency matrix [47]. Alternatively, adjacency matrices have been called "N^2" diagrams or design structure matrices within the engineering systems literature [48].

Definition 13. Directed Graph (digraph) [47]: D, consists of a collection nodes B, and a collection of arcs E, for which we write $D = (B, E)$. Each arc $e = \langle b_1, b_2 \rangle$, is said to join node $b_1 \in B$ to another (not necessarily distinct) node b_2. Vertex b_1 is called the tail of e, whereas b_2 is its head.

Definition 14. Adjacency matrix [47]: A, is binary and of size $\sigma(B) \times \sigma(B)$ and its elements are given by:

$$A(y_1, y_2) = \begin{cases} 1 & \text{if } \langle b_{y_1}, b_{y_2} \rangle \text{ exists} \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

where the $\sigma()$ gives the size of a set.

Here, the functions would represent the nodes and would be interconnected with the lines found in activity diagrams.

Example 1. Consider the second level of abstraction of the functional architecture in Figure 6 as a directed graph. It's (functional architecture) adjacency matrix is

$$A_f = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{2}$$

From a dynamic perspective, and perhaps one of the central tasks in traditional engineering effort, functional elements can be replaced by their mathematical function equivalents called device models. Given inputs $\mathbf{U}$, outputs $\mathbf{Y}$, and state variables $\mathbf{X}$, device models with algebraic equations take the form $\mathbf{Y} = g(\mathbf{X}, \mathbf{U})$. Device models with differential equations take the form $\dot{\mathbf{X}} = f(\mathbf{X}, \mathbf{U})$ where the output $\mathbf{Y}$ is algebraically related to the states and inputs $\mathbf{Y} = g(\mathbf{X}, \mathbf{U})$ [49]. Device models with difference equations take the form $\mathbf{X}_k = f(\mathbf{X}_k, \mathbf{U}_k)$ where the output $\mathbf{Y}_k$ is algebraically related to the states and inputs $\mathbf{Y} = g(\mathbf{X}_k, \mathbf{U}_k)$ [50]. These device models are then aggregated via the functional architecture and may then be simulated to quantitatively understand aggregate *system behavior* and overall system performance.

C. Physical Architecture Domain

The physical architecture domain embodies the engineering system and is made up of mutually connected components. Again, there is considerable variation in nomenclature across the Axiomatic Design and MBSE literature in regards to the elements of the physical architecture domain. Generically, they are called components which may be aggregated into modules,

resources, and subsystems. Furthermore, they may be characterized by attributes such as size, shape, color. In Axiomatic Design, both attributes and their associated components at any level of aggregation are called design parameters **DP**. The rationale for the AD convention is discussed later in Section IV. Both of these conventions are used in this chapter.

In traditional top-down systems engineering and Axiomatic Design, the generation of the physical architecture proceeds as a synthesis activity in concert with the allocated architecture [3], [28] to be discussed in Section III. In contrast, bottom-up design methodologies generate the physical architecture as an analytic activity pre-supposing the set of components and their associated technologies [28].

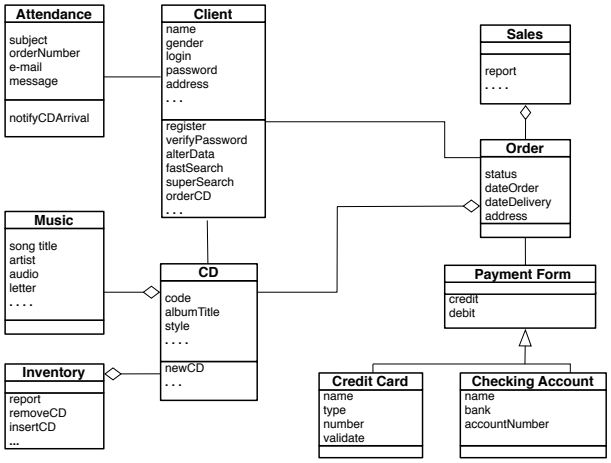


Fig. 7. An Example SysML Block Definition Diagram (Adapted from X)

The development of the physical architecture is well supported in SysML [18], [19]. As shown in Figure 3, this includes four diagrams that provide complementary views of the overall system structure at different levels of engineering design detail. These include the block definition, internal block, parametric, and package diagrams. While all of these are useful in detailed design, the block definition diagram is most often applied in conceptual design before detailed analytical equations are available. Figure 7 shows an example SysML block diagram which include many of the system thinking concepts related the conceptual design of physical architecture. Association links represent interconnected components. They would ultimately realize function output as material, energy, information, people or money. Composition links represent whole-part relationships. Classification links represent generalization-specialization relationships. It is important to note that the SysML block definition diagram either models the generic or the instantiated physical architecture but not both at the same time.

Definition 15. Generic Physical Architecture [28]: A description of the partitioned elements of the physical architecture without any specification of the performance characteristics of the physical resources that comprise each element.

Definition 16. Instantiated Physical Architecture [28]: A generic physical architecture to which complete definitions of the performance characteristics of the resources have been added (including the number of each type of resource).

For example, a block definition diagram can represent generic relationships between roles in an organization chart or they can instantiate those roles to the specific people that hold them. In the precursor to SysML, the UML 2.0 specification reserved class diagrams for the generic physical architecture and the object diagram for the instantiated physical architecture [51].

Much like the functional architecture domain, the physical architecture domain also lends itself to mathematical description. From a static perspective, physical elements (at any given level of abstraction) can be organized into a directed graph and its associated adjacency matrix [47].

Example 2. Consider the second level of abstraction of the functional architecture in Figure 7 as a graph. It's (physical architecture) adjacency matrix is

$$A_p = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3}$$

From a dynamic perspective, the evolution of a physical architecture remains a subject of cutting edge research.

D. Process Architecture Domain

Recalling Definition 8, the process architecture domain is composed of the set of processes and their relationships that characterize how the physical architecture is to be produced. In many ways, it strongly resembles the functional architecture domain [52]. Each process is defined as a transitive verb stated in third person singular followed by its associated object/operand. Indeed, recent work on the Axiomatic Design of manufacturing systems (independent of the products that they produce) treats manufacturing processes as elements of the manufacturing system's functional architecture domain [53]–[55]. Therefore, there is significant similarity in how the two domains are modeled in general as well as in SysML.

Despite the similarities between the two domains, their respective roles in MBSE and Axiomatic Design are fundamentally different. In Axiomatic Design, as shown in Figure 1, the process domain occurs as a synthesis *after* the physical architecture has been developed. In contrast, MBSE does not explicitly treat downstream "manufacturability" as a fourth domain. It is possible to include manufacturing requirements and constraints as part of the requirements engineering process in the stakeholder requirements domain. However, such an approach assumes that the physical architecture, its required manufacturing processes, and the stakeholders that own them are already known to some degree in advance.

E. Multi-Domain Mapping in the Engineering Design of System

With Axiomatic Design's four domains introduced, it becomes clear that the engineering design of large systems is particularly complex. As shown in Figures 1 and 2, not only must engineers proceed sequentially from one domain to the next to synthesize the system, they must also retrace those steps backwards to analytically validate and verify the original

intent of synthesis. This is a tremendous task of information management and requires the three activities identified in Table II.

TABLE II
INFORMATION MANAGEMENT TASKS IN THE ENGINEERING DESIGN OF SYSTEMS

- **Element Information**: All of the elements in all of the domains must be systematically identified.
- **Intra-Domain Information**: The links between elements within a given domain must be systematically identified.
- **Inter-Domain Information**: The links between elements across two domains must be systematically identified.

In order to conceptualize this undertaking, graph theory again proves to be useful. The literature has proposed the engineering systems matrix (ESM) (Figure 8) as a form of multi-domain matrix [56], [57]. In addition to the four domains

	System Drivers	Stakeholders	Stakeholder Requirements	Functions	Components	Processes
System Drivers	System Drivers x System Drivers	Stakeholders x System Drivers	Requirements x System Drivers	Functions x System Drivers	Components x System Drivers	Processes x System Drivers
Stakeholders	System Drivers x Stakeholders	Stakeholders x Stakeholders	Requirements x Stakeholders	Functions x Stakeholders	Components x Stakeholders	Processes x Stakeholders
Stakeholder Requirements	System Drivers x Requirements	Stakeholders x Requirements	Requirements x Requirements	Functions x Requirements	Components x Requirements	Processes x Requirements
Functions	System Drivers x Functions	Stakeholders x Functions	Requirements x Functions	Functions x Functions	Components x Functions	Processes x Functions
Components	System Drivers x Components	Stakeholders x Components	Requirements x Components	Functions x Components	Components x Components	Processes x Components
Processes	System Drivers x Processes	Stakeholders x Processes	Requirements x Processes	Functions x Processes	Components x Processes	Processes x Processes

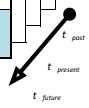


Fig. 8. Engineering Systems Multiple-Domain Matrix [28], [31]

of engineering in Figure 1, it adds the system drivers domain and the stakeholder domain.

Definition 17. System Drivers Domain [56], [57]: A representation of the non-human portion of the environmental domain and are composed of the set of all non-human components that act or are acted on by the system. The system drivers can include the economic, political, and technical influences that constrain, enable, or alter the characteristic of components in the system

Definition 18. Stakeholder Domain [56], [57]: A social network of stakeholders in an engineering system which may be classified as external or internal. The external stakeholders constitute the remaining portion of the environmental domain and consist of the human entities that affect or are affected by the system but that do not control components within the system boundary. Likewise, internal stakeholders are the human entities that contribute to the goals of the system and control components within the system.

This addition serves to recognize that an engineering system exists within a context in which multiple system drivers are influencing its conception, planning, and operation. It also recognizes the presence of multiple stakeholders which may indeed pose conflicting or misaligned stakeholder requirements. The elements of these six domains combined essentially form the nodes of the underlying engineering systems graph. The non-zero elements of the engineering systems matrix indicate the presence of links between them; be they within a given domain or across multiple domains. The engineering systems matrix in Figure 8 is naturally highly sparse but it nevertheless serves as a tool to manage the information highlighted in Table II. Finally, it is important to recognize that the engineering systems matrix can be viewed as "snapshots" in time; either as the system is designed, or as it is operated.

From the lens of the engineering systems matrix, Figure 1 now appears highly structured. It addresses the bottom four blocks of the main block diagonal. Its mappings address their first off-block diagonals immediately above and below the main block diagonal. The other interactions are intentionally omitted so as to avoid needless complexity in the engineering design process. In other words, an engineering design process that specifically eliminates needless interactions is one that can allow the system to be designed, planned, and operated more efficiently.

III. THE ALLOCATED ARCHITECTURE: DESIGN SYNTHESIS & ANALYSIS

Much of the focus of Axiomatic Design has gone specifically to the mapping between the functional and the physical domains [3]. In MBSE, this is often called the allocated architecture [28].

Definition 19. Allocated Architecture [28]: A complete description of the system design, including the functional architecture allocated to the physical architecture; derived input/output; technology, system-wide, trade-off, and qualification requirements for each component; an interface architecture that has been integrated as one of the components; and complete documentation of the design and major design decisions.

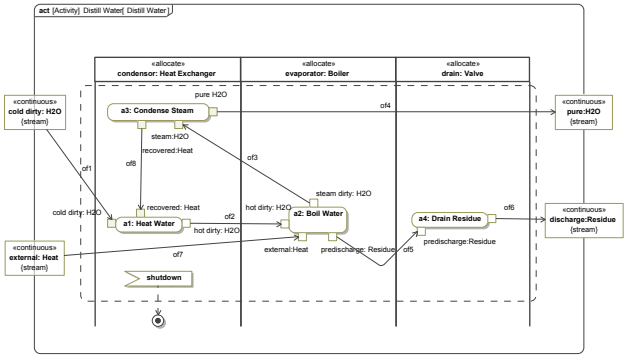


Fig. 9. Example: System Processes, Resources & Allocation [**?**]

There are several ways to model the allocated architecture in model based systems engineering. Two of these are described here. The first has already been shown in Figure 7 where a method can be executed as a function allocated to a given class. For example, "verify password" is such a method for the "Client" class. The second approach is shown in Figure 9. Here, an activity diagram has been partitioned into "swim lanes" so that a given action is allocated to the physical component that executes it. In such a way, the functional and physical architecture domains are closely tied.

While the other activities in the engineering design of systems are certainly not to be neglected, there are several reasons for Axiomatic Design's specific focus on the allocated architecture. First, the allocation of function to form represents in engineering design the *"the moment of synthesized embodiment"*. In other words, prior to that moment, the design only had a set of functional requirements but afterwards, the design now includes a set of physical elements or design parameters **DP** that now describe a physically embodied way to achieve these functions. Second, this allocation of from to function is done *quantitatively* (rather than graphically) to the level of mathematical detail that is available at the time. Third, the nature of this allocation, as later sections describe ultimately drive many aspects of an engineering system including its life cycle properties. The successful transition from the functional to the physical domain requires effective design synthesis and analysis.

A. Design Synthesis

Engineering discussions on design synthesis are often neglected. Casually speaking, a designer's "creativity" is engaged and "voila" innovation happens! However, a rigorous understanding of design synthesis must root itself into the formal foundations of philosophy, logic, and linguistics. After all, it is a process which brings a *system model* $\mathcal{M}$ into being from the mind(s) of its designer(s). In this regard, the Ullmann triangle [58] shown in Figure 10 proves to be a useful construct. It derives from fundamental works [59], [60] upon which much of modern linguistics is based. In the left-hand triangle, a *domain conceptualization* $\mathcal{C}$ is an immaterial entity that only exists in the mind of a community of users of a *language* $\mathcal{L}$ [61]. As such, it is a mental abstraction of a *real domain* $\mathcal{D}$ (i.e. as it is observed in the natural sciences) [61]. Furthermore, the language $\mathcal{L}$ is composed of set of modeling primitives which collectively represent the domain conceptualization $\mathcal{C}$ [61]. The right-hand triangle instantiates the one the left. The abstraction $\mathcal{A}$ is an instance of the domain conceptualization $\mathcal{C}$ [61], and now abstracts a system model $\mathcal{M}$ as the output of a design process. Such a process is not direct. It must return to the domain conceptualization $\mathcal{C}$ its representing language $\mathcal{L}$ and its associated modeling primitives. The system model $\mathcal{M}$ then follows as an instance of the language $\mathcal{L}$.

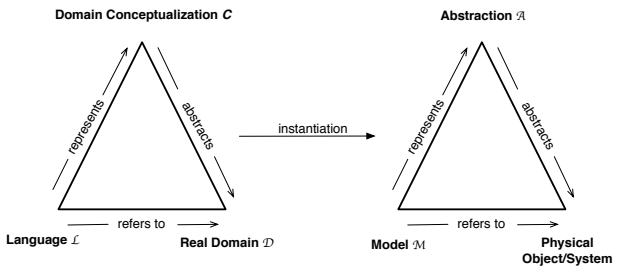


Fig. 10. The Role of the Ullman Triangle in Design Synthesis

One practical challenge in the engineering systems field is that modeling primitives are domain specific. For example, the topic of motion in machine design is often treated with primitives like linkages, cams, and gear trains [62]. Similarly, the design of dynamic systems across multiple energy domains has lead to primitives such as generalized capacitors, inductors, resistors, transformers, and gyrators [46]. In business dynamics, stocks and flows are often used as primitives [63]. More broadly, the engineering systems literature has recently developed simple but encompassing taxonomies of function and form [10]. The object process modeling language, as the name suggests, uses objects and processes as primitives [42].

In all cases, the *domain appropriateness* and *comprehensibility* of a language can be formally assessed. Guizzardi writes [61]: "In order for a model $\mathcal{M}$ to faithfully represent an abstraction $\mathcal{A}$, the modeling primitives of the language $\mathcal{L}$ used to produce $\mathcal{M}$ should faithfully represent the domain conceptualization $\mathcal{C}$ used to articulate the represented abstraction $\mathcal{A}$". A formal assessment of a language $\mathcal{L}$ yields the properties of *soundness*, *completeness*, *lucidity*, and *laconicity* which are graphically depicted in Figure 11 and formally defined [64].

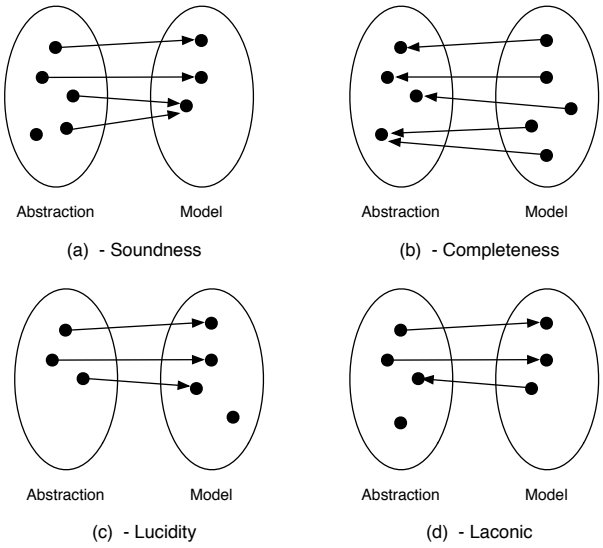


Fig. 11. Graph Theoretical Representation of Mapping between a Model and its Abstraction: (a) Soundness (b) Completeness (c) Lucidity (d) Laconicity [64].

Definition 20. Soundness [64]: A language $\mathcal{L}$ is sound w.r.t. to a domain $\mathcal{D}$ iff every modeling primitive in the language has an interpretation in the domain abstraction $\mathcal{A}$.

Definition 21. Completeness [64]: A language $\mathcal{L}$ is complete w.r.t. to a domain $\mathcal{D}$ iff every concept in the domain abstraction $\mathcal{A}$ of that domain is represented in a modeling primitive of that language

Definition 22. Lucidity [64]: A language $\mathcal{L}$ is lucid w.r.t. to a domain $\mathcal{D}$ iff every modeling primitive in the language represents at most one domain concept in $\mathcal{A}$.

Definition 23. Laconicity [64]: A language $\mathcal{L}$ is laconic w.r.t. to a domain $\mathcal{D}$ iff every concept in the abstraction $\mathcal{A}$ of that domain is represented at most once in the model of that language.

The absence of these properties violate conversational maxims that assume thought is "relevant, clear, unambiguous, brief, not overly informative, and true" [65]. Interestingly, UML [66] has been assessed relatively positively in the context of Figures 10 and 11 [61]. Perhaps, this result may serve to provide a

theoretical reason for the successful adoption of SysML/UML as an integral part of design synthesis in MBSE.

While the process of conceptualization in Figure 10 is necessary to define design synthesis, it is ultimately insufficient. Afterall, it must be reconciled with the constrained mapping presented in Figure 1. To this end, the term synthesis may be defined in its philosophical sense.

Definition 24. Synthesis [67]: the third stage of argument in a dialectic which reconciles the mutually contradictory first two propositions of thesis and antithesis.

Therefore, design synthesis can be defined as:

Definition 25. Design Synthesis: a synthesis process which reconciles the conceptualization of a set of design parameter primitives $\mathbb{DP}$ (as a thesis) with the satisfaction of a set of functional requirements $\mathbf{FR}$ (as an antithesis). Mathematically,

$$\mathbf{DP} = f_s(\mathbf{FR}, \mathcal{DP}) \tag{4}$$

Furthermore, it is understood that these design parameter primitives are domain appropriate, comprehensible and effectively represent a conceptualization of the designer's experience. Two distinct designers may generate different sets of design parameters $\mathbf{DP}$ given that they may retain different design parameter primitives $\mathcal{DP}$ in their mental conceptualization.

B. Design Analysis

Unlike design synthesis, engineering discussions on design analysis are given significantly greater attention. Perhaps this is because, the inputs of design analysis are design parameters. As entities, they are well described, often-quantitatively, in the natural sciences which form the roots of the modern engineering science. In contrast, design parameter primitives exist in the ontological sciences which draw from philosophy, logic and linguistics. Furthermore, while design synthesis requires the reconciliation of design parameter primitives with functional requirements to identify a set of design parameters, design analysis takes the previously identified design parameter information to determine whether they satisfy the functional requirements. In a sense, design synthesis defines the nature of an engineering system/artifact and design analysis refines it.

Axiomatic Design describes design analysis with a design equation:

$$\mathbf{FR} \$ f_a(\mathbf{DP}) \tag{5}$$

where $f_a()$ retains the "function of" meaning and the relatively new symbol \$ means "satisfies" when read from right to left. When the design parameters and functional requirements quantitatively represent the physical quantities of an engineering system, then $f_a()$ comes to represent its associated laws of physics. In such a way, Equation 5 can be rewritten as:

$$\mathbf{FR} = f_a(\mathbf{DP}) \tag{6}$$

whose first derivative gives:

$$\mathbf{\Delta FR} = [B]\mathbf{\Delta DP} \tag{7}$$

where now the non-zero elements of the *design matrix* $B(i,j)$ highlight the existence of a dependence between an arbitrary $\mathbf{FR}_i$ and an arbitrary design parameter $\mathbf{DP}_j$[1]. It is important to very clearly distinguish $f_s()$ and $f_a()$; while the former describes the designers' mental process of generating the design parameters, the latter describes the laws of physics that relate the now already existing design parameters to their functional requirements. Axiomatic Design does not require the designer(s) to have full knowledge of the mathematical form of $f_a()$ during design synthesis. As Section V later discusses, the knowledge of these mathematical forms may not be fully available during early-stage conceptual design. Instead, its axioms only require the designer(s) to have knowledge of the existence of non-zero elements in B and act accordingly. Graphically, the designer need only have the intent of allocating a functional element to a physical one as depicted in Figure 9.

IV. The Independence & Information Axioms

Axiomatic Design was developed out of a need to make the field of design more scientific [2], [3]. In 2001, Suh writes: "The goal of Axiomatic Design is manifold: to make human designers more creative, to reduce the random search process, to minimize iterative trial-and-error process, to determine the best designs among those proposed, and to endow the computer with creative power through the creation of a scientific base for the design field." [3]. These lofty goals brought about a highly intensive and *empirical* research process in which the common elements of *"good"* designs were identified [2], [3]. These common elements were ultimately distilled into Axiomatic Design's two axioms. The interested reader is referred to [2] for further details on the research process used to develop Axiomatic Design. These two axioms, stated today, are:

Axiom 1. The Independence Axiom [2], [3]: Maintain the independence of the functional requirements (FRs).

Axiom 2. The Information Axiom [2], [3]: Minimize the information content of the design.

Consequently, these axioms have lead to the development of Axiomatic Design's many theorems and corollaries summarized for convenience in the Appendix of this book. Each of these is now discussed conceptually.

A. The Independence Axiom

The Independence Axiom is a statement that applies as equally to design synthesis as design analysis. Its interpretation in the former requires that the set of functional requirements be *mutually exclusive and collectively exhaustive* [2], [3]. In other words, the requirements engineering process that produces

[1]Note that many works on Axiomatic Design, including later chapters in this book, simply write $\mathbf{FR} = [B]\mathbf{DP}$ to concisely convey the meaning of Equations 5-7. While this notational short-hand is often sufficient to properly implement Axiomatic Design, it does cloud the small but meaningful differences between the three equations. Furthermore, such a shorthand suggests that the $f()$ in Equation 6 is a linear matrix equation consisting of real numbers when indeed no such restriction is formally required.

the functional requirements may be viewed as an ontology development activity that produces part of the system's design language. Furthermore, it is very difficult to conceive any synthesis function $f_s()$ that retains its nature as a function when its input domain is neither mutually exclusive nor collectively exhaustive. This agrees with the discussion in Section II-B which made this requirement to avoid downstream design errors.

The Independence Axiom is applied in design analysis through the use of Equation 6 and more specifically the matrix properties of the design matrix B. When B is a diagonal matrix, then the system is said to be *uncoupled*. When B is either a lower triangular matrix or may be converted into a lower triangular matrix by row swapping operations, then the system is said to be *decoupled*. When B does not have either of these two forms, then the system is said to be *coupled*. Uncoupled designs are preferred over decoupled ones. And coupled designs are said to not comply with the Independence Axiom. Therefore, the application of the Independence Axiom has a component that allows the synthesis function f_s to exist and then it guides the designer(s) through an analysis step to verify if the resulting laws of physics describe an uncoupled or coupled system.

Researchers, educators, and practitioners often experience several misconceptions as they convey the Independence Axiom to their peers. The most notable of these misconceptions is in the concept of coupling. As Section II-B has described, the functional domain contains couplings that occur from the sequential relationship between functions. The MBSE literature often calls these couplings *interactions* [68]. They are formally modeled by the existence of non-zero elements in the functional domain's adjacency matrix. Similarly, and as Section II-C has described, the physical domain contains couplings that occur from the sequential relationship between components. The MBSE literature often calls these couplings *interfaces* [68]. They are formally modeled by the existence of non-zero elements in the physical domain's adjacency matrix. Both of these are examples of *intra-domain* information[2]. In analysis, the Independence Axiom exclusively addresses the *inter-domain* information with the design matrix that describes the allocation architecture. Intra-domain coupling is not relevant.

Another concern that emerges over the Independence Axiom is its statement in the imperative rather than more traditionally as a declarative (e.g. 1*X=X - multiplicative identity axiom) or a conditional (e.g. if x=y, then y=x – reflexivity axiom) statement. Here, again, it is important to recall that design is both synthesis as well as analysis. A statement in the imperative is conducive in the practical sense to the process of design synthesis. In other words, Suh's Independence Axiom is directed to a design synthesis practitioner rather than a design analyst audience. The Independence Axiom can be recast as a declarative statement as follows.

[2]Note that the flows of matter, energy, information, money, and people within interfaces and interactions are collectively the same artifacts. However, their representation need not be the same in the two domains. Indeed, it is easy to prove, that they are same if and only if the design matrix is square and diagonal.

Axiom 3. Independence Axiom (Recast): Maintaining the independence of the functional requirements **FR** during design synthesis yields "good" designs.

Another misconception arises when Suh [3] speaks of *good* designs being synthesized as a result of the application of Axiomatic Design. Here, the criticism is directed to the term "good" as a statement of subjective value rather than a quantifiable scientific measure. The Independence Axiom's mathematical statement of a diagonal (or lower triangular) design matrix is matched to the qualitative notion of a "good design" by extensive empirical observation. Methodologically, and logically, this is an extension of the corresponding concept in ontological science. The formal mathematical definitions of soundness, completeness, lucidity, and laconicity yield the qualitatively and widely held conversational maxims of "relevance and clarity". Both statements are built upon extensive empirical observation relating a qualitative conceptual idea to a formal definition in the corresponding analytical model. There is no difference in the nature of the logical reasoning.

B. The Information Axiom

The Information Axiom introduced at the beginning of the section applies in design analysis once the Independence Axiom has been applied. It calls for the minimization of a design's information content I which is defined in terms of the probabilities P_i of satisfying each of the functional requirements FR_i [3].

$$I = -\sum_{i}^{\sigma(FR)} \log_2 P_i \tag{8}$$

These probabilities may be understood practically by Figure 12. Each functional requirement may be specified as a design range. In practice, however, the true value of the functional requirements falls within a probability density function that is characterized by a system range. The area under the probability density function that falls within the design range provides a measure of the probability of satisfying a given functional requirement. $A_{cr_i} = P_i$ [3]. A deeper discussion of the Information Axiom and its applications is provided in Chapter 2.

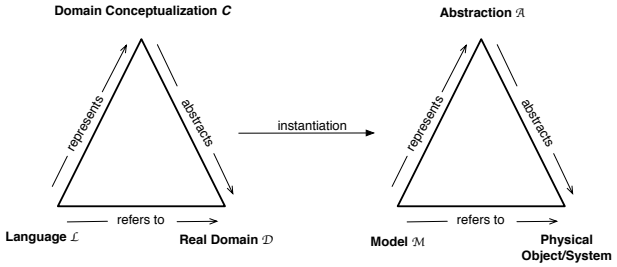


Fig. 12. A Practical Understanding of the Information Axiom: Design Range, System Range & Probability Density Function of a Functional Requirement

C. Axiomatic Design's Theorems and Corollaries

These two axioms form the foundation of Axiomatic Design. Over several decades, many theorems and corollaries have been proven from these two axioms. The interested

reader can find many of these summarized in the appendix of this book with citations to their original references and corresponding proofs.

V. Functional & Physical System Hierarchy in Large Systems

At this point, a careful reader would recognize that the previous section's treatment of Axiomatic Design was at a single level of decomposition and hence is insufficient to address the functional and physical system hierarchy in large systems as represented in Figures 6 and 7. This section now expands the discussion of the previous one to address the systems thinking concepts of decomposition and specialization in large fixed and large flexible engineering systems.

Definition 26. Large Fixed Engineering System [3]: an engineering system with a large set of functional requirements which do not evolve over time and whose components also do not change over time.

Definition 27. Large Flexible Engineering System [3], [27]: an engineering system with many functional requirements that not only evolve over time, but also can be fulfilled by one or more design parameters.

A. Large Fixed Engineering Systems

Large fixed engineering systems continue to follow the Axiomatic Design discussions provided in Sections X and Y. A synthesis function $f_s()$ is used to conceptually represent a designers generation of a set of design parameters $\mathbf{DP}$, and an analysis function $f_a()$ following the laws of physics is assessed to determine adherence to the independence and information axioms. For small systems (i.e. those with a very few functional requirements), such a process is relatively straightforward.

For large fixed engineering systems, however, such an approach is impractical for two reasons. The first issue is in the size of $\mathbf{FR}$ and $\mathbf{DP}$ in $f_s()$. In 1956, as a psychologist, Miller [69] noted that human short term memory is limited to 7 +/-2 elements. Therefore, the synthesis function $f_s()$ is ill-defined beyond this size. Instead, the functional requirements must be *aggregated* into this manageable size and design parameters must be synthesized *conceptually* at a corresponding *level of abstraction*. For example, the design parameters can now be whole subsystems such as whole drive trains, buildings, or organizations. This brings about the second practical issue which is in the nature of $\mathbf{FR}$ and $\mathbf{DP}$ in Equation 4. $\mathbf{FR}$ and $\mathbf{DP}$ are no longer real numbers and so $f_a()$ is no longer well-defined as an algebraic or differential equation. In practice, designers may not know the exact impact of a given design parameter on a given functional requirement, and yet they must continue to synthesize engineering systems in spite of this. Inevitably, this causes a profound intellectual conflict between the mathematical rigor of engineering analysis and the creativity of engineering synthesis. It appears most vividly early on in the conceptualization of an engineering system where interestingly engineering design decisions have the greatest impact.

Axiomatic Design resolves this conflict by allowing design analysis to occur, albeit with a less precise form of mathematics. At higher levels of abstraction, early on in the conceptualization of an engineering system, $\mathbf{FR}$ and $\mathbf{DP}$ represent elements not numbers. Therefore, Equation 3 must be represented using graph and set theory. In large fixed engineering systems, Equation 3 becomes

$$\mathbf{FR}\$(B \circledast \mathbf{DP}) \tag{9}$$

where the aggregation operation $\circledast$ is defined as:

Definition 28. Aggregation Operator $\circledast$ [53], [70]: Given boolean matrix A and sets B and C, $C = A \circledast B$ is equivalent to:

$$C(i) = \bigcup_j a(i,j) \wedge b(j) \tag{10}$$

The \$ in Equation 9 is often replaced with a simple $=$ as a matter of notational convenience without change in the underlying meaning.

$$\mathbf{FR} = B \circledast \mathbf{DP} \tag{11}$$

Note that, B now comes to represent an (undirected) incidence matrix between the sets $\mathbf{FR}$ and $\mathbf{DP}$.

Definition 29. Incidence matrix [47]: M of size $\sigma(B) \times \sigma(E)$ is given by:

$$M(i,j) = \begin{cases} -1 & \text{if } b_i \text{ is the head of arc } e_j \\ 1 & \text{if } b_i \text{ is the tail of arc } e_j \\ 0 & \text{otherwise} \end{cases} \tag{12}$$

As mentioned previously, the presence of nonzero elements in the design matrix B, is graphically represented in SysML as an allocation of a functional element to a physical element as shown in Figure 9. Axiomatic Design collates these graphical interactions to highlight their underlying mathematical form. Furthermore, Equation 11 also implies that Equation 6 remains true and consequently the Independence Axiom can still be applied without change.

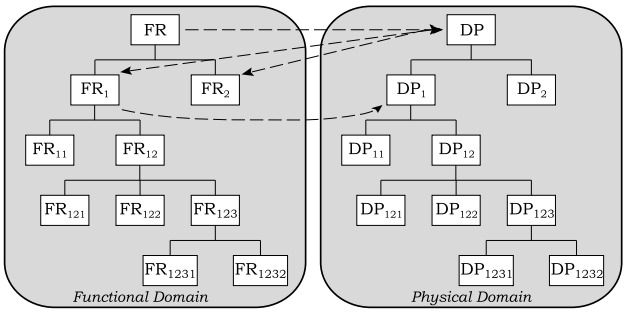


Fig. 13. Synthesis Paths in Simultaneous Hierarchical Physical and Functional Decomposition

The introduction of graph and set theory into the discussion now allows a formal understanding of how Axiomatic Design manages system complexity and multiple layers of abstraction. Figure 13 shows Axiomatic Design's dual-hierarchy of the functional and physical architecture domains. It represents the full allocated architecture of the system and is generated along the depicted synthesis arrows by what Suh calls a "Zig-Zag" approach. The highest level of design parameters are

synthesized from the highest level of functional requirements by Equation 4 and then analyzed by Equation 11 for adherence to the Independence Axiom. At this point, a new set of decomposed functional requirements $\underline{\mathbf{FR}}$ must be synthesized based upon the designer's knowledge of the higher level functional requirements $\mathbf{FR}$ and design parameters $\mathbf{DP}$.

$$\underline{\mathbf{FR}} = f_s(\mathbf{FR}, \mathbf{DP}) \tag{13}$$

As with Equation 4, Equation 13 describes the designer's mental process of synthesis as an abstract mathematical function. Again, two distinct designers may produce the decomposition entirely differently depending on their knowledge of the design parameters $\mathbf{DP}$ as an abstract model of the system in real life. The result of the decomposition can be analyzed using the aggregation operation [71].

$$\mathbf{FR} = A_f \circledast \underline{\mathbf{FR}} \tag{14}$$

where A_f is a binary functional aggregation matrix that describes to which high-level functional requirement, each low-level functional requirement pertains [71]. Strict mutually exclusive aggegation places a constraint on the nature of the aggregation matrix.

$$\mathbf{1}^{\sigma(\mathbf{FR})T} A_f = \mathbf{1}^{\sigma(\underline{\mathbf{FR}})T} \tag{15}$$

Once, the new set of functional requirements $\underline{\mathbf{FR}}$ have been synthesized, the corresponding set of design parameters $\underline{\mathbf{DP}}$ can again be synthesized by Equation 4. Consequently, the aggregation of the physical architecture can be analyzed [71].

$$\mathbf{DP} = A_p \circledast \underline{\mathbf{DP}} \tag{16}$$

where, again, strict mutually exclusive aggregation requires

$$\mathbf{1}^{\sigma(\mathbf{DP})T} A_f = \mathbf{1}^{\sigma(\underline{\mathbf{DP}})T} \tag{17}$$

Consequently,

$$B * A_p = A_f \underline{\mathrm{B}} \tag{18}$$

where B is the higher level design matrix and $\underline{\mathrm{B}}$ is the lower level design matrix. Equation 18 may be solved when the left and right inverses of B and $\underline{\mathrm{B}}$ respectively exist. Furthermore, when they are identity matrices, (e.g. the Independence Axiom is fulfilled), the aggregation in the functional and physical architectures becomes the same. $A_f = A_p$.

B. Large Flexible Engineering Systems

The Axiomatic Design of large flexible engineering systems was first mentioned by Suh in his 2001 text [3] and has since been significantly developed [27], [53], [54], [71]–[73]. Large flexible engineering systems typically require attention at higher levels of abstraction but are otherwise similar to large fixed systems. Equation 4 describes design synthesis and Equation 11 describes design analysis. Recalling Definition 27, the distinguishing feature of flexibility is achieved by a strict adherence to the Independence Axiom. Therefore, $\mathbf{B} = \mathbf{I}^n$, where n equivalently represents the number of design parameters or functional requirements. Conceptually, this is because a non-identity design matrix would imply that either a single design parameter affects more than one functional requirement or vice versa. Consequently, when it comes time to *reconfigure* the engineering system with an addition or removal of a functional or physical element, other changes would need to be made as well. In contrast, adherence to the Independence Axiom, enables a "plug & play" engineering system where functional and physical elements can be added or removed at will [55], [70].

By Definition 27, large flexible engineering systems have functional requirements that can be fulfilled by potentially many design parameters. An identity design matrix does not show this. Therefore, in order to reveal this functional redundancy, the set of functional requirement *instances* $\mathbf{FR}$ must be distinguished from the set of functional requirement *classes* $\mathbb{FR}$[3]. A new design equation can then be written to relate $\mathbb{FR}$ to $\mathbf{DP}$.

$$\mathbb{FR} = \mathbf{J} \odot \mathbf{DP} \tag{19}$$

where $\mathbf{J}$ represents the system knowledge base and $\odot$ represents matrix boolean multiplication.

Definition 30. System Knowledge Base [27], [53], [54], [71]–[73]: A binary matrix $\mathbf{J}$ of size $\sigma(\mathbb{FR}) \times \sigma(\mathbf{DP})$ whose element $\mathbf{J}(w, v) \in \{0, 1\}$ is equal to one when action e_{wv} (in the SysML sense)[4] exists as a functional requirement $\mathbb{FR}_w$ being executed by a design parameter $\mathbf{DP}_v$.

Definition 31. Matrix Boolean Multiplication $\odot$ [27], [53], [54], [71]–[73]: Given sets or boolean matrices B and C and boolean matrix A, $C = A \odot B$ is equivalent to:

$$C(i, k) = \bigvee_j A(i, j) \wedge B(j, k) \tag{20}$$

Interestingly, it is equally valid to replace the set of design parameters instances $\mathbf{DP}$ with the set of design parameter classes $\mathbb{DP}$. In such a case, Axiomatic Design addresses the design of generic or *reference architectures* rather than specific, instantiated or system architectures [74]–[76].

By Definition 27, large flexible engineering systems have functional requirements that can evolve over time. To that effect, the Axiomatic Design literature introduces a system constraints matrix.

Definition 32. System Constraints Matrix [27], [53], [54], [72], [73]: A binary matrix $\mathbf{K}$ of size $\sigma(\mathbb{FR}) \times \sigma(\mathbf{DP})$ whose element $\mathbf{K}(w, v) \in \{0, 1\}$ is equal to one when a constraint eliminates action e_{wv} from the action set.

A *reconfiguration process* is said to change the value of the system constraints matrix [77]. Therefore, the system knowledge base contains information on the *existence* of capabilities in the engineering system. Meanwhile, the constraints matrix contains information of their *availability* [76], [78]. Quantitatively keeping track of these capabilities is done via

[3]Note that many works on Axiomatic Design do not make this distinction between functional requirement instances and functional requirement classes because it is rarely needed within a single design work. Here, the distinction is made in order to maintain the conceptual link between large fixed and large flexible engineering systems and the universality of the Independence Axiom in both cases.

[4]The word "action" is meant in the technical sense of allocated functional elements in SysML's activity diagram. See Figure 9 for details. These actions represent capabilities in the engineering system.

the system's structural degrees of freedom as a measure [27], [53], [54], [71]–[73].

Definition 33. LFES Sequence-Independent Structural Degrees of Freedom [27], [53], [54], [71]–[73]: The set of independent actions $\mathcal{E}_S$ that completely defines the available processes in a LFES. Their number is given by:

$$DOF = \sigma(\mathcal{E}) = \sum_{w}^{\sigma(\mathbb{FR})} \sum_{v}^{\sigma(\mathbf{DP})} [\mathbf{J} \ominus \mathbf{K}](w, v) \tag{21}$$

Consequently, the redundancy of functional requirement $\mathbf{FR}_w$ is [27], [53], [71]:

$$\mathcal{R}_w = \sum_{v}^{\sigma(\mathbf{DP})} [\mathbf{J} \ominus \mathbf{K}](w, v) \tag{22}$$

The flexibility of the design parameter $\mathbf{DP}_v$ is [27], [53], [71]:

$$\mathcal{F}_v = \sum_{w}^{\sigma(\mathbb{FR})} [\mathbf{J} \ominus \mathbf{K}](w, v) \tag{23}$$

These measures are important because redundancy and flexibility are important enabling properties for many life cycle properties [27], [55], [71].

Large flexible engineering systems require a careful discussion of the Axiomatic Design dual hierarchy [70], [71]. Fundamentally, this is because functional and physical elements can be added or removed. Consequently, their respective hierarchies must be allowed to change as well. Developing the Axiomatic Design dual hierarchy for large flexible engineering systems, downwards in the direction of design synthesis, proceeds in the same way as for large fixed engineering systems. The system is viewed in terms of functional requirement *instances* rather than classes. Because the Independence Axiom has been strictly maintained, each structural degree of freedom can be designed as previously described as if it were its own system. The engineering design problem is separable. Therefore, the addition or removal of a structural degree of freedom adds or removes all of the associated lower branches in the dual hierarchy.

It is also useful to consider the dual-hierarchy of a large flexible engineering system upwards in the direction of design analysis. Here, it is no longer required to aggregate the physical and functional hierarchies simultaneously [27], [71], [73], [76]. It is particularly common in bottom-up design to aggregate only the physical hierarchy into higher-level design parameters or *resources*. A corresponding functional aggregation may not occur. This is because physical aggregation and functional aggregation do not have the same meaning and do not necessarily imply each other [41], [42]. Consider, for example, five tasks as functional requirements and five individuals as design parameters each of whom completes one task. This a large flexible engineering system that fulfills the Independence Axiom. The five individuals may be aggregated into a resource called a team without making any statement about the five tasks. They may not be related in any way (.i.e. share any functional interaction). Similarly, the five tasks may be aggregated into a project without making any statement about the five individuals who complete them. They may have never met (i.e. share any physical interface). Physical aggregation is particularly interesting because it yields resources with high flexibility. An addition or removal of a design parameter yields the corresponding change in a resource's flexibility. In contrast, the functional aggregation of a large flexible engineering system may result a rigid top-down structure. Any time the set of functional requirements changes, the functional hierarchy would need to change as well. In a project, the elimination of a single task causes the elimination of the project as a whole.

Thus far, the two systems thinking concepts of instantiation and aggregation/decomposition have been discussed as a means of managing system complexity. The discussion now turns to the last such concept: specialization/generalization. The Axiomatic Design for large flexible engineering system literature has addressed this topic implicitly in several works [27], [53], [54], [72], [76]. More explicitly, bottom-up generalization is a form of conditional aggregation.

Definition 34. Generalized Design Parameter: A generalized design parameter $\widetilde{\mathbf{DP}}_i$ is an aggregation of a set of design parameters $\mathbf{DP}$ if any $\mathbf{DP}_k \in \mathbf{DP}$ is capable of doing any of the common functional requirements $\mathbf{cFR} \subseteq \mathbf{FR}$.

$$\widetilde{\mathbf{DP}}_i = A_g \circledast \mathbf{DP} \tag{24}$$

where $A_g(i,k) = 1$ iff $J(j,k) = 1$ for any $FR_j \in \mathbf{cFR}$.

Note that the definition of a generalized design parameter requires the identification of a set of common functional requirements that can be done by the low level design parameters $\mathbf{DP}$ as well as its generalization $\widetilde{\mathbf{DP}}$. Also, note that unlike a regular aggegration, specialization does not require constraint in Equation 17.

C. An Illustrative Example

To summarize the discussion on Axiomatic Design for large flexible engineering systems, consider the following example.

Example 3. Consider the manufacturing system depicted in Figure 14. It consists of a drill press and milling machine. The former is able to drill a hole, and the latter is able to do the same and mill surfaces. Each contains its respective fixture. It also has two one-way conveyors between between them.

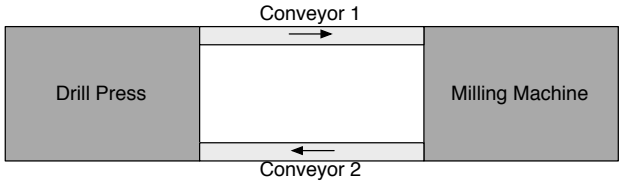


Fig. 14. A Simple Manufacturing System with one drill press, one milling machine and two conveyors

A large fixed engineering system analysis yields:

$\mathbf{FR}$ = {drill hole, drill hole, mill surface, store the part at point A, transport part from point A to point B, transport part from point B to point A, store the part at B}.
$\mathbf{DP}$ = {drill press, milling machine drill, milling machine end mill, drill press fixture, conveyor 1, conveyor 2, milling machine fixture}.

The design matrix $\mathbf{B} = \mathbf{I}^7$. The Independence Axiom is satisfied.

For a large flexible engineering system analysis, the functional requirement classes are viewed instead of their instances.

$\mathbb{FR} = \{$drill hole, mill surface, store the part at A, transport part from point A to point B, transport part from point B to point A, store the part at B$\}$.

An aggregation matrix is applied so that the drill press, milling machine and conveyor system appear as single resources.

$\overline{\mathbf{DP}} = \{$drill press, milling machine, conveyor system$\}$.

$$J = \begin{bmatrix} 1 & 1 & 0 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix} = \left[\begin{array}{c|c} J_M & \mathbf{0} \\ \hline & J_H \end{array}\right] \tag{25}$$

That partitioning of the system knowledge base into J_M and J_H comes from generalization. J_M represents structural degrees of freedom that have a transformational function. J_H represents structural degrees of freedom that have a transportational function.

Resource flexibility: The three resources have flexibilities of 2, 3 and 2 structural degrees of freedom respectively.
Functional redundancy: All the functional requirements have a functional redundancy of 1 except "drill hole".

The failure of the conveyor system would appear as two constraints in the constraints matrix

$$K = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} \tag{26}$$

After the failure of the conveyor system, $DOF = 5$.

VI. Engineering Systems Applications of Axiomatic Design

With a solid Axiomatic Design foundation in place, the chapter can now return to the engineering systems discussion initiated in the introduction. This section highlights the potential applications of Axiomatic Design in the development of engineering with regards to three specific challenges: 1.) a quantitative understanding of life cycle properties 2.) a treatment of cyber-physical systems and 3.) a treatment of hetero-functional networks.

A. Quantitative Understanding of Life Cycle Properties

The subject of life cycle properties in engineering systems is an expansive one [11] with potentially whole text books devoted to a single property (e.g. resilience [79]). Consequently, a detailed discussion can not be provided here. Nevertheless, the Axiomatic Design, MBSE, and engineering systems concepts provided thus far can serve to provide a guiding structure to the subject. A quantitative formulation of life cycle properties first requires a qualitative understanding of which engineering domains in Figure 1 or more generally the engineering systems matrix in Figure 8 pertain to that specific life cycle property. Furthermore, the life cycle property must be classified as a description of system structure or system behavior [70].

TABLE III
A Preliminary Classification of Life Cycle Properties in Engineering Systems

	System Structure	**System Behavior**
Functional Architecture Domain	centrality [80], modularity [81], [82]	stability [49], [50], sustainability [83]–[85]
Physical Architecture Domain	centrality [80], modularity [81], [82]	not applicable
Allocated Architecture Domain	flexibility [53], redundancy [53], reconfigurability [55], [70], static resilience [27]	dynamic resilience [86], [87], stability/synchronization [88], sustainability [83]–[85]

Therefore, Table III presents a first-pass classification of life cycle properties. As mentioned at the end of Section II-B, the central focus of traditional engineering effort is often devoted to understanding system behavior from quantitative models of system function [28]. Sustainability, when viewed in the sense of the provision of a certain level of product or service while limiting the quantities of input resources and byproduct emissions may be similarly classified [83]–[85]. Many life cycle properties, however, depend on an explicit – often graph theoretic – description of system structure. Modularity [81], [82] and centrality [80] are two such life cycle properties that depend on the form of a graph's adjacency matrix; be it in the functional or physical architecture domains. One may argue that perhaps one of the great benefits of MBSE (e.g. through SysML) is that it can abstract details of system behavior to provide a clear view of system structure and its associated life cycle properties.

Still other life cycle properties emerge from the allocated architecture. It is here that the Axiomatic Design design matrix B and knowledge base J, as different types of incidence matrices, are quite valuable in developing a quantitative treatment. Section V-B already provided measures for two relatively simple life cycle properties of system structure: flexibility [53] and redundancy [53]. More complex life cycle properties such as reconfigurability [55], [70] and static resilience [27], often require that a new adjacency matrix A_ρ be constructed from the system's structural degrees of freedom [27], [73].

$$A_\rho = [J \ominus K][J \ominus K]^T \ominus K_\rho \tag{27}$$

where K_ρ is a constraints matrix that imposes continuity relations between the individual structural degrees of freedom. Interestingly, reconfigurability clearly differentiates between large fixed and large flexible engineering systems [27], [53], [54], [73]. As expected, engineering systems that adhere to the Independence Axiom are fundamentally more reconfigurable than systems that do not [55], [70].

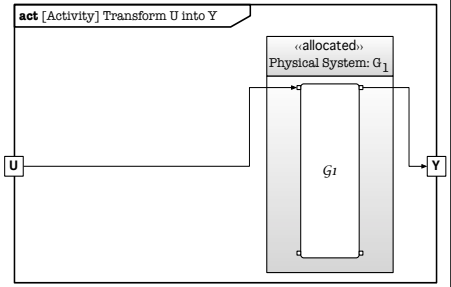
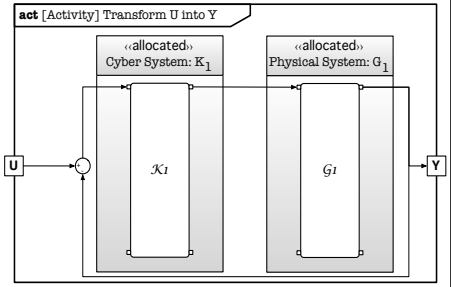
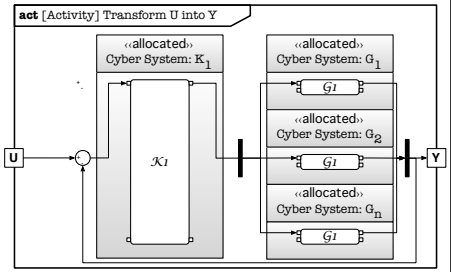
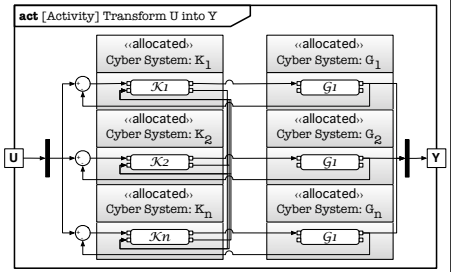
Cyber-Physical System	Activity/Block Diagram	System Behavior	Axiomatic Design
a.) Open-Loop Physical System C0P1 1-Resource		$\mathbf{Y} = \mathcal{G}_1\mathbf{U}$	$\mathbf{Y} = [1] \circledast G_1$
b.) Closed-Loop Cyber-Physical System C1P1 1-Resource 1-Controller		$\mathbf{Y} = \mathcal{G}_{cp}\mathbf{U}$ $\mathbf{Y} = \frac{\mathcal{K}_1\mathcal{G}_1}{\mathbf{I} + \mathcal{K}_1\mathcal{G}_1}\mathbf{U}$	$\mathbf{Y} = [1] \circledast G_{cp}$ $\mathbf{Y} = \begin{bmatrix} 1 & 1 \end{bmatrix} \circledast \begin{bmatrix} K_1 \\ G_1 \end{bmatrix}$
c.) Closed-Loop Cyber-Physical System C1PN n-Resources 1-Controller		$\mathbf{Y} = \mathcal{G}_{cp}\mathbf{U}$ $\mathbf{Y} = \frac{\mathcal{K}_1 \begin{bmatrix} \mathcal{G}_1 & & & \\ & \mathcal{G}_2 & & \\ & & \ddots & \\ & & & \mathcal{G}_n \end{bmatrix}}{\mathbf{I} + \mathcal{K}_1 \begin{bmatrix} \mathcal{G}_1 & & & \\ & \mathcal{G}_2 & & \\ & & \ddots & \\ & & & \mathcal{G}_n \end{bmatrix}}\mathbf{U}$	$\mathbf{Y} = [1] \circledast G_{cp}$ $\mathbf{Y} = \begin{bmatrix} 1 & 1 & 0 & \cdots & 0 \\ 1 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & 0 & 0 & \cdots & 1 \end{bmatrix} \circledast \begin{bmatrix} K_1 \\ G_1 \\ G_2 \\ \vdots \\ G_n \end{bmatrix}$
d.) Closed-Loop Cyber-Physical System CNPN n-Resources n-Controller		$\mathbf{Y} = \mathcal{G}_{cp}\mathbf{U}$ $\mathbf{Y} \approx \frac{\begin{bmatrix} \mathcal{K}_1\mathcal{G}_1 & & & \\ & \mathcal{K}_2\mathcal{G}_2 & & \\ & & \ddots & \\ & & & \mathcal{K}_n\mathcal{G}_n \end{bmatrix}}{\mathbf{I} + \begin{bmatrix} \mathcal{K}_1\mathcal{G}_1 & & & \\ & \mathcal{K}_2\mathcal{G}_2 & & \\ & & \ddots & \\ & & & \mathcal{K}_n\mathcal{G}_n \end{bmatrix}}\mathbf{U}$	$\mathbf{Y} = [1] \circledast G_{cp}$ $\mathbf{Y} = \begin{bmatrix} 1 & & \cdots & 0 \\ 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & 1 \end{bmatrix} \circledast \begin{bmatrix} G_{cp1} \\ G_{cp2} \\ \vdots \\ G_{cpn} \end{bmatrix}$

Fig. 15. Cyber-Physical Systems from the perspective of SysML, transfer functions, and Axiomatic Design

Finally, many life cycle properties require an understanding of the relationship between the allocated architecture and the system behavior. Dynamic resilience – in particular the capacity to "bounce back" to a certain system performance after a disruption – depends equally on the system's constituent device models [75] as on flexibility of its resources and their redundancy [86], [87]. Synchronization of engineering systems with coupled oscillators (e.g. the electric power grid, swarms/fleets of moving vehicles) utilizes many of the techniques required to analyze stability but add further steps that consider the physical architecture's adjacency matrix [88]. Finally, when the prior view of sustainability is expanded to also include cost performance, it must balance the performance of the functional architecture to the cost of the physical architecture via the allocated architecture.

B. Treatment of Cyber-Physical Systems

Axiomatic Design sheds light on many of the architectural questions related to cyber-physical systems. Consider the four simple control theory examples shown in Figure 15. Figure 15a depicts an open-loop physical system C0P1. The second column shows its corresponding SysML activity diagram. The system, as a whole, transforms an input $\mathbf{U}$ into an output $\mathbf{Y}$. A single action $\mathcal{G}_1$ achieves this activity and it is allocated to the physical system G_1. The distinction between the functional element $\mathcal{G}_1$ and the physical element G_1 is critical. The third column shows the corresponding system behavior as a transfer function involving $\mathcal{G}_1$. Meanwhile, the fourth column shows the corresponding allocated architecture as an Axiomatic Design equation involving G_1. A single functional requirement is placed on the output $\mathbf{Y}$ and the single resource single resource G_1 is designed to achieve it and consequently the Independence Axiom is fulfilled with an identity design matrix.

It is important to note that from a mathematical perspective, the two equations in Figure 15a are indeed equivalent; albeit very differently arranged. At this level of abstraction, the functional form of $\mathcal{G}_1$ is hidden away. Similarly, G_1 hides away all of its constituent (design) parameters; the same one would expect to find in $\mathcal{G}_1$. Proving their equivalence requires two steps. First, $\mathcal{G}_1$ is written explicitly and then differentiated with respect each of the design parameters in G_1 so that it takes the form of Equation 7. Similarly, G_1 is decomposed down to an "atomic" level of design parameters represented by real numbers and then differentiated. Although these two equations are equivalent, one focuses the designer on a system's behavior, the other focuses the designer's attention to its allocated architecture.

Figure 15b depicts a closed-loop cyber-physical system. The SysML diagram depicts two components: a cyber component K_1 and a physical component G_1. The former realizes the action (i.e. transfer function) $\mathcal{K}_1$ while the latter realizes the action $\mathcal{G}_1$. An ouptut feedback loop is introduced. The third column shows the overall closed loop transfer function $\mathcal{G}_{cp}$ as a top level of abstraction or equivalently one level of abstraction down in terms of $\mathcal{K}_1$ and $\mathcal{G}_1$. At the highest level of abstraction, the Axiomatic Design equation resembles the open-loop system and the Independence Axiom is fulfilled. However, one level of abstraction down, the design equation reveals a "redundant design"[5]. This coupled design does not adhere to the Independence Axiom and requires the physical system to be fixed first before the controller can be design. Not surprisingly, many feedback control design methods require iterative tuning.

Figure 15c now depicts a closed-loop cyber-physical system with one centralized controller and n resources. As in Figures 15a and 15b, this system may be viewed as an open-loop system fulfilling the Independence Axiom. However, one level of abstraction down, the coupled and redundant design matrix reappears as expected. This is unfortunate from the perspectives of reconfigurability and resilience. Although the four physical systems are mathematically uncoupled, the failure or "hack" of the centralized control affects the performance of all of the functional requirements [53]–[55], [70]. Therefore, from an Axiomatic Design perspective centralized controllers are to be avoided.

Finally, Figure 15d depicts a closed-loop cyber-physical system with n controllers matched to n resources. If the n controllers are entirely independent, the system now fully adheres to the Independence Axiom supporting the case for distributed control. Furthermore, from a reconfigurability and resilience perspective, there exists no single point of failure [53]–[55], [70]. This is a very special and rare case however. Instead, much research on multi-agent control systems [89]–[94] introduces communication between the n agent controllers to achieve greater coordination between the n physical resources. If this inter-agent communication algorithm is either 5x faster or slower than the physical system's dynamics, then the system's transfer function is approximately time-scale separable and the Independence Axiom continues to be supported [78]. Furthermore, the performance of each physical resource can be enhanced with the addition of a single fast but local controller for each physical system. These Axiomatic Design principles have been used to develop multi-agent control system architectures for production [76] and power systems [78].

C. Treatment of Hetero-functional Networks

As engineering systems integrate togehter to form hetero-functional networks, they pose several challenges to existing approaches to engineering design and modeling. As has been previously mentioned in Sections II-B and II-C, adjacency matrices are typically used to provide abstract graph theoretic models of either the functional or the physical architecture. Furthermore, the most common applications of graph theory are homo-functional in nature [27], [80]. Artifacts (of some kind) are transported along edges between physical locations represented as nodes. This is sufficient for individual engineering systems. For example, in transportation systems, the nodes often physically represent intersections and stations while edges/arcs represent roads, rails or transportation routes [95]–[100]. Meanwhile, in power systems, the nodes often

[5]In the Axiomatic Design of large fixed systems, redundant designs have more design parameters than functional requirements [3].

physically represent generators, substations, and loads while the edges represent the power lines [101]–[104]. The integration of two or more engineering systems, however, requires a richer approach because the nodes and edges have completely different physical meanings. Alternatively, bond graphs [46] and linear graphs [105] are promising techniques to quantitatively model continuous-time physical systems across multiple energy domains. Their current level of development, however, lacks the systems thinking abstractions mentioned throughout this chapter. Furthermore, they have limited capability to handle systems with discrete-event dynamics and consequently offer limited support for dynamics and decision-making driven by people; be they individuals or organizations.

In contrast, Axiomatic Design enables the study of engineering systems as they integrate together to form hetero-functional networks. Production systems, due to their hetero-functional nature, have been proven to be an excellent application domain for advancing Axiomatic Design. In Example 3, Axiomatic Design for large flexible engineering systems was used to model the physical part of a production system's allocated architecture at multiple levels of abstraction. Later chapters in this book will demonstrate Axiomatic Design's application to decision-making processes as the cyber-part of production systems. More explicitly, Equation 27 allows the system knowledge base to be converted into a hetero-functional graph with structural degrees of freedom as nodes [27], [73]. Such a graph based upon the Axiomatic Design knowledge based was later used to directly derive a production system's discrete-event dynamics [106]. Similarly derived discrete-event dynamics were demonstrated for transportation systems as an engineering system with no transformation functions [107]. Meanwhile, the Axiomatic Design knowledge base was used with device models to derive the continuous-time dynamics of power systems [78]. With these methodological developments in place, Axiomatic Design has been used to develop full simulations of the energy-water nexus [74], [108], electrified transportation systems [109], and microgrid-enabled production systems [110] as truly hetero-functional and integrated engineering systems. The broad diversity of these applications demonstrates the utility of Axiomatic Design to engineering systems as a field.

VII. Conclusion

In the 21^{st} century, engineers are facing engineering challenges of increasingly greater scope. These include many large complex products and systems described later in this book but even more generally whole engineering systems. This chapter has introduced Axiomatic Design within this larger engineering systems context. It began by identifying Axiomatic Design and MBSE as two engineering design methodologies and theories that when appropriately developed have the potential to address the methodological challenges of engineering systems. The chapter introduced Axiomatic Design and its relationship to MBSE in terms of four domains of engineering design: stakeholder requirements, functional architecture, physical architecture, and process domains. It also discussed a system's allocated architecture with special care given to differentiate its synthesis and analysis. Here, Axiomatic Design's ability to quantify the allocated architecture was highlighted in terms of its Independence & Information Axioms. At that point, the chapter generalized these concepts with several hierarchical techniques to manage system complexity. This allowed the discussion to return to the three methodological challenges mentioned in the introduction: quantification of life cycle properties, design of cyber-physical systems, and design of hetero-functional networks. Taken together, the chapter details the essentials of Axiomatic Design, relates it to MBSE, and highlights its potential applications in the field of engineering systems.

References

[1] N. P. Suh, A. Bell, and D. Gossard, "On an axiomatic approach to manufacturing and manufacturing systems," *Journal of Manufacturing Science and Engineering*, vol. 100, no. 2, pp. 127–130, 1978.

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

[3] ——, *Axiomatic Design: Advances and Applications*. Oxford University Press, 2001.

[4] T. Tomiyama, P. Gu, Y. Jin, D. Lutters, C. Kind, and F. Kimura, "Design methodologies: Industrial and educational applications," {*CIRP*} *Annals - Manufacturing Technology*, vol. 58, no. 2, pp. 543 – 565, 2009. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S000785060900170X

[5] N. P. Suh, *Complexity : theory and applications*. New York: Oxford University Press, 2005.

[6] E. M. Benavides, *Advanced engineering design: an integrated approach*. Elsevier, 2011.

[7] S. J. Kim, N. P. Suh, and S. G. Kim, "Design of software systems based on axiomatic design," *Robotics and Computer-Integrated Manufacturing*, vol. 8, no. 4, 1991.

[8] N. P. Suh, "Design and operation of large systems," *Journal of Manufacturing Systems*, vol. 14, no. 3, 1995.

[9] S. H. Do and N. P. Suh, "Systematic OO programming with axiomatic design," *Computer*, vol. 32, no. 10, pp. 121–124, 1999.

[10] O. L. De Weck, D. Roos, and C. L. Magee, *Engineering systems : meeting human needs in a complex technological world*. Cambridge, Mass.: MIT Press, 2011. [Online]. Available: http://www.knovel.com/knovel2/Toc.jsp?BookID=4611http://mitpress-ebooks.mit.edu/product/engineering-systems

[11] BKCASE Editorial Board, *The Guide to the Systems Engineering Body of Knowledge (SEBoK)*, v1.3 ed. Hoboken, NJ: The Trustees of the Stevens Institute of Technology, 2014.

[12] SE Handbook Working Group, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. International Council on Systems Engineering (INCOSE), 2015.

[13] A. Pyster, D. H. Olwell, T. L. Ferris, N. Hutchison, S. Enck, J. F. Anthony Jr., D. Henry, and A. Squires, "Graduate reference curriculum for systems engineering (grcse) version 1.0," The Trustees of Stevens Institute of Technology, Tech. Rep., 2012.

[14] V. Piuri, A. G. Aghdam, and S. Nahavandi, "Editorial," *Systems Journal, IEEE*, vol. 7, no. 1, pp. 2–3, March 2013.

[15] O. L. de Weck, "A vision for the future of the journal systems engineering," *Systems Engineering*, vol. 16, no. 4, pp. 379–380, 2013.

[16] A. W. Wymore, *Model-based systems engineering*. CRC press, 1993, vol. 3.

[17] MBSE Initiative Working Group, "Model-based systems engineering (mbse) wiki," INCOSE, Tech. Rep., 2015. [Online]. Available: http://www.omgwiki.org/MBSE/doku.php

[18] T. Weilkiens, *Systems engineering with SysML/UML modeling, analysis, design*. Burlington, Mass.: Morgan Kaufmann, 2007.

[19] S. Friedenthal, A. Moore, and R. Steiner, *A Practical Guide to SysML: The Systems Modeling Language*, 2nd ed. Burlington, MA: Morgan Kaufmann, 2011.

[20] M. Amin, "Toward secure and resilient interdependent infrastructures," *Journal of Infrastructure Systems*, vol. 8, no. 3, pp. 67–75, 2002.

[21] ——, "System-of-systems approach," in *Intelligent Monitoring, Control, and Security of Critical Infrastructure Systems*. Springer, 2015, pp. 317–354.

[22] A. M. Annaswamy, M. Amin, C. L. Demarco, T. Samad, J. Aho, G. Arnold, A. Buckspan, A. Cadena, D. Callaway, E. Camacho, M. Caramanis, A. Chakrabortty, A. Chakraborty, J. Chow, M. Dahleh, A. D. Dominguez-Garcia, D. Dotta, A. M. Farid, P. Flikkema, D. Gayme, S. Genc, M. G. i. Fisa, I. Hiskens, P. Houpt, G. Hug, P. Khargonekar, H. Khurana, A. Kiani, S. Low, J. McDonald, E. Mojica-Nava, A. L. Motto, L. Pao, A. Parisio, A. Pinder, M. Polis, M. Roozbehani, Z. Qu, N. Quijano, and J. Stoustrup, *IEEE Vision for Smart Grid Controls: 2030 and Beyond*, A. M. Annaswamy, M. Amin, C. L. Demarco, and T. Samad, Eds. New York NY: IEEE Standards Association, 2013. [Online]. Available: http://www.techstreet.com/ieee/products/1859784

[23] P. Gadonneix, Y. D. Kim, A. Pacific, S. Asia, G. Ward, and C. Frei, *Water for Energy*, 2010.

[24] W. Su, H. Rahimi-eichi, W. Zeng, and M.-y. Chow, "A Survey on the Electrification of Transportation in a Smart Grid Environment," *IEEE Transactions on Industrial Informatics*, vol. 8, no. 1, pp. 1–10, 2012.

[25] Stockholm Environment Institute, "Understanding the Nexus: Background paper for The Water, Energy and Food Security Nexus Conference," Stockholm Environment Institute, Bonn, Tech. Rep. November, 2011.

[26] B. Murgante and G. Borruso, "Smart cities in a smart world," in *Future City Architecture for Optimal Living*. Springer, 2015, pp. 13–35.

[27] A. M. Farid, "Static Resilience of Large Flexible Engineering Systems : Axiomatic Design Model and Measures," *IEEE Systems Journal (in press)*, vol. PP, no. 99, pp. 1–12, 2015. [Online]. Available: http://amfarid.scripts.mit.edu/resources/Journals/IES-J19.pdf

[28] D. M. Buede, *The engineering design of systems : models and methods*, 2nd ed. Hoboken, N.J.: John Wiley & Sons, 2009.

[29] SE Handbook Working Group, *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*. International Council on Systems Engineering (INCOSE), 2011.

[30] D. W. Oliver, T. P. Kelliher, and J. G. Keegan, *Engineering complex systems with models and objects*. New York: McGraw-Hill, 1997.

[31] K. Forsberg and H. Mooz, "The relationship of systems engineering to the project cycle," *Engineering Management Journal*, vol. 4, no. 3, pp. 36–43, 1992.

[32] K. Pohl, *Requirements engineering: fundamentals, principles, and techniques*. Springer Publishing Company, Incorporated, 2010.

[33] B. Berenbach, D. Paulish, J. Kazmeier, and A. Rudorfer, *Software & systems requirements engineering: in practice*. McGraw-Hill, Inc., 2009.

[34] E. Hull, K. Jackson, and J. Dick, *Requirements engineering*. Springer Science & Business Media, 2010.

[35] P. A. Laplante, *Requirements engineering for software and systems*. CRC Press, 2013.

[36] M. K. Thompson, "Improving the requirements process in axiomatic design theory," *CIRP Annals-Manufacturing Technology*, vol. 62, no. 1, pp. 115–118, 2013.

[37] ——, "A classification of procedural errors in the definition of functional requirements in axiomatic design theory," in *Proceedings of the 7th International Conference on Axiomatic Design (ICAD'13), Worcester, MA, June*, 2013.

[38] L.-K. Chan and M.-L. Wu, "A systematic approach to quality function deployment with a full illustrative example," *Omega*, vol. 33, no. 2, pp. 119–139, 2005.

[39] ——, "Quality function deployment: A literature review," *European Journal of Operational Research*, vol. 143, no. 3, pp. 463–497, 2002.

[40] ——, "Quality function deployment: a comprehensive review of its concepts and methods," *Quality Engineering*, vol. 15, no. 1, pp. 23–35, 2002.

[41] D. Dori, *Object-process methodology : a holistics systems paradigm*. Berlin ; New York: Springer, 2002.

[42] ——, *Object-process methodology: A holistic systems paradigm*. Springer Science & Business Media, 2013.

[43] I. Pirbhai and D. Hateley, "Strategies for real-time system specification," *New York: Dorset House*, 1987.

[44] S. M. McMenamin and J. F. Palmer, *Essential systems analysis*. Yourdon Press, 1984.

[45] J. G. Miller, *Living systems*. Mcgraw-Hill, 1978.

[46] D. Karnopp, D. L. Margolis, and R. C. Rosenberg, *System dynamics : a unified approach*, 2nd ed. New York: Wiley, 1990. [Online]. Available: http://www.loc.gov/catdir/enhancements/fy0650/90012110-t.htmlhttp://www.loc.gov/catdir/enhancements/fy0650/90012110-b.htmlhttp://www.loc.gov/catdir/enhancements/fy0650/90012110-d.html

[47] M. van Steen, *Graph Theory and Complex Networks: An Introduction*. Maarten van Steen, 2010, no. January.

[48] S. D. Eppinger and T. R. Browning, *Design structure matrix methods and applications*. Cambridge, Mass.: MIT Press, 2012.

[49] B. Friedland, *Control system design : an introduction to state-space methods*. New York: McGraw-Hill, 1986.

[50] K. Ogata, *Discrete-time control systems*, 2nd ed. Englewood Cliffs, N.J.: Prentice Hall, 1994.

[51] J. Rumbaugh, I. Jacobson, and G. Booch, *The Unified Modeling Language Reference Manual*. Addison-Wesley, 2005.

[52] C. A. Brown, "Axiomatic design of manufacturing processes considering coupling," in *Proceedings of ICAD2014 The Eighth International Conference on Axiomatic Design*, 2014.

[53] A. M. Farid and D. C. McFarlane, "Production degrees of freedom as manufacturing system reconfiguration potential measures," *Proceedings of the Institution of Mechanical Engineers, Part B (Journal of Engineering Manufacture) – invited paper*, vol. 222, no. B10, pp. 1301–1314, Oct. 2008. [Online]. Available: http://amfarid.scripts.mit.edu/resources/Journals/IEM-J05.pdf

[54] A. M. Farid, "Product Degrees of Freedom as Manufacturing System Reconfiguration Potential Measures," *International Transactions on Systems Science and Applications – invited paper*, vol. 4, no. 3, pp. 227–242, 2008. [Online]. Available: http://amfarid.scripts.mit.edu/resources/Journals/IEM-J04.PDF

[55] ——, "Measures of Reconfigurability & Its Key Characteristics in Intelligent Manufacturing Systems," *Journal of Intelligent Manufacturing*, vol. 1, no. 1, pp. 1–26, 2014. [Online]. Available: http://dx.doi.org/10.1007/s10845-014-0983-7

[56] J. E. Bartolomei, "Qualitative knowledge construction for engineering systems: extending the design structure matrix methodology in scope and procedure," Massachuseets Institute of Technology Engineering Systems Division, Tech. Rep., 2007.

[57] J. E. Bartolomei, D. E. Hastings, R. de Neufville, and D. H. Rhodes, "Engineering systems multiple-domain matrix: An organizing framework for modeling large-scale complex systems," *Systems Engineering*, vol. 15, no. 1, pp. 41–61, 2012.

[58] S. Ullmann, "{Semantics: An Introduction to the Science of Meaning}," 1979.

[59] C. K. Ogden, I. A. Richards, B. Malinowski, and F. G. Crookshank, *The meaning of meaning*. Kegan Paul London, 1923.

[60] F. De Saussure, W. Baskin, and R. Harris (trans.), *Course in general linguistics*. Open Court Publishing Company, 1986 (original 1916).

[61] G. Guizzardi, *Ontological foundations for structural conceptual models*. CTIT, Centre for Telematics and Information Technology, 2005.

[62] J. E. Shigley, C. R. Mischke, and T. H. Brown, *Standard handbook of machine design*, 3rd ed. New York: McGraw-Hill, 2004.

[63] J. D. Sterman, *Business dynamics: systems thinking and modeling for a complex world*. Irwin/McGraw-Hill Boston, 2000, vol. 19.

[64] G. Guizzardi, "On ontology, ontologies, conceptualizations, modeling languages, and (meta) models," *Frontiers in artificial intelligence and applications*, vol. 155, p. 18, 2007.

[65] H. P. Grice, "Logic and conversation," in *Syntax and Semantics*. New York: Academic Press, 1970, vol. 3, pp. 43–58.

[66] M. Blaha, J. Rumbaugh, and M. BIBlaha, "Object-oriented modeling and design with UML," pp. xvii, 477 p., 2005.

[67] Anonymous, "Synthesis," Dictionary.com, Tech. Rep., 2015. [Online]. Available: http://dictionary.reference.com/browse/synthesis?s=t

[68] A. Kossiakoff, W. N. Sweet, and Knovel (Firm), *Systems engineering principles and practice*. Hoboken, N.J.: Wiley-Interscience, 2003. [Online]. Available: http://www.knovel.com/knovel2/Toc.jsp?BookID=1430

[69] G. A. Miller, "The magical number seven, plus or minus two: Some limits on our capacity for processing information." *Psychological review*, vol. 63, no. 2, p. 81, 1956.

[70] A. M. Farid, "Reconfigurability Measurement in Automated Manufacturing Systems," Ph.D. Dissertation, University of Cambridge Engineering Department Institute for Manufacturing, 2007. [Online]. Available: http://amfarid.scripts.mit.edu/resources/Theses/IEM-TP00.pdf

[71] I. S. Khayal and A. M. Farid, "Axiomatic Design Based Volatility Assessment of the Abu Dhabi Healthcare Labor Market," *Journal of Enterprise Transformation (in press)*, vol. 1, no. 1, pp. 1–20, 2015.

[72] A. M. Farid, "An Axiomatic Design Approach to Non-Assembled Production Path Enumeration in Reconfigurable Manufacturing Systems," in *2013 IEEE International Conference on Systems Man and Cybernetics*, Manchester, UK, 2013, pp. 1–8. [Online]. Available: http://dx.doi.org/10.1109/SMC.2013.659

[73] A. Viswanath, E. E. S. Baca, and A. M. Farid, "An Axiomatic Design Approach to Passenger Itinerary Enumeration in Reconfigurable Transportation Systems," *IEEE Transactions on Intelligent Transportation Systems*, vol. 15, no. 3, pp. 915 – 924, 2014. [Online]. Available: http://amfarid.scripts.mit.edu/resources/Journals/TES-J08.pdf

[74] W. N. Lubega and A. M. Farid, "A Reference System Architecture for the Energy-Water Nexus," *IEEE Systems Journal*, vol. PP, no. 99, pp. 1–11, 2014. [Online]. Available: http://amfarid.scripts.mit.edu/resources/Journals/EWN-J11.pdf

[75] S. Rivera, A. M. Farid, and K. Youcef-Toumi, "Chapter 15 - a multi-agent system coordination approach for resilient self-healing operations in multiple microgrids," in *Industrial Agents*, P. L. Karnouskos, Ed. Boston: Morgan Kaufmann, 2015, pp. 269 – 285. [Online]. Available: http://amfarid.scripts.mit.edu/resources/Books/SPG-B03.pdf

[76] A. M. Farid and L. Ribeiro, "An Axiomatic Design of a Multi-Agent Reconfigurable Mechatronic System Architecture," *IEEE Transactions on Industrial Informatics*, vol. 11, no. 5, pp. 1142–1155, 2015. [Online]. Available: http://dx.doi.org/10.1109/TII.2015.2470528

[77] A. M. Farid and W. Covanich, "Measuring the Effort of a Reconfiguration Process," in *Emerging Technologies and Factory Automation, 2008. ETFA 2008. IEEE International Conference on*, Hamburg, Germany, 2008, pp. 1137–1144. [Online]. Available: http://dx.doi.org/10.1109/ETFA.2008.4638540

[78] A. M. Farid, "Multi-Agent System Design Principles for Resilient Coordination & Control of Future Power Systems," *Intelligent Industrial Systems (in press)*, vol. 1, no. 1, pp. 1–9, 2015. [Online]. Available: http://amfarid.scripts.mit.edu/resources/Journals/SPG-J17.pdf

[79] E. Hollnagel, D. D. Woods, and N. Leveson, *Resilience Engineering: Concepts and Precepts*, kindle edi ed. Aldershot, U.K.: Ashgate Publishing Limited, 2006.

[80] M. Newman, *Networks: An Introduction*. Oxford, United Kingdom: Oxford University Press, 2009. [Online]. Available: http://books.google.ae/books?id=LrFaU4XCsUoC

[81] J. K. GERSHENSON, G.J.PRASAD, and Y. ZHANG, "Product modularity: definitions and benefits," *Journal of Engineering Design*, vol. 14, no. 3, pp. 295–313, 2003.

[82] ——, "Product modularity: measures and design methods," *Journal of Engineering Design*, vol. 15, no. 1, pp. 33–51, 2004. [Online]. Available: http://www.tandf.co.uk/journals

[83] D. Mebratu, "Sustainability and sustainable development: historical and conceptual review," *Environmental impact assessment review*, vol. 18, no. 6, pp. 493–520, 1998.

[84] P. Glavič and R. Lukman, "Review of sustainability terms and their definitions," *Journal of cleaner production*, vol. 15, no. 18, pp. 1875–1885, 2007.

[85] C. Böhringer and P. E. Jochem, "Measuring the immeasurable—a survey of sustainability indices," *Ecological economics*, vol. 63, no. 1, pp. 1–8, 2007.

[86] A. Madni and S. Jackson, "Towards a conceptual framework for resilience engineering," *IEEE Systems Journal*, vol. 3, no. 2, pp. 181–191, 2009.

[87] R. Bhamra, S. Dani, and K. Burnard, "Resilience: the concept, a literature review and future directions," *International Journal of Production Research*, vol. 49, no. 18, pp. 5375–5393, Sep 2011.

[88] A. Arenas, A. Diaz-Guilera, J. Kurths, Y. Moreno, and C. Zhou, "Synchronization in complex networks," *Physics Reports*, vol. 469, no. 3, pp. 93–153, Dec. 2008. [Online]. Available: http://dx.doi.org/10.1016/j.physrep.2008.09.002

[89] W. Shen and D. Norrie, "Agent-Based Systems for Intelligent Manufacturing: A State-of-the-Art Survey," *Knowledge and Information Systems: An International Journal*, vol. 1, no. 2, pp. 129–156, 1999.

[90] P. Leitao, "Agent-based distributed manufacturing control: a state-of-the-art survey," *Engineering Applications of Artificial Intelligence*, vol. 22, no. 7, pp. 979–991, Oct. 2009. [Online]. Available: http://dx.doi.org/10.1016/j.engappai.2008.09.005

[91] R. Babiceanu and F. Chen, "Development and applications of holonic manufacturing systems: A survey," *Journal of Intelligent Manufacturing*, vol. 17, pp. 111–131, 2006.

[92] V. Marik, M. Fletcher, M. Pechoucek, O. Stepankova, H. Krautwurmova, and M. Luck, "Holons and Agents: Recent Developments and Mutual Impacts," in *Multi-Agent Systems and Applications II: Lecture Notes in Artificial Intelligence*. Springer Verlag, 2002, pp. 233–267.

[93] D. C. McFarlane and S. Bussmann, "Developments in holonic production planning and control," *Production Planning & Control*, vol. 11, no. 6, pp. 522–536, Jan 2000.

[94] D. McFarlane, S. Bussmann, and S. M. Deen, "Holonic Manufacturing Control: Rationales, Developments and Open Issues," in *Agent-Based Manufacturing*. Berlin: Springer-Verlag, 2003, pp. 303–326.

[95] W. H. Ip and D. Wang, "Resilience and friability of transportation networks: Evaluation, analysis and optimization," *IEEE Systems Journal*, vol. 5, no. 2, pp. 189–198, Jun 2011.

[96] V. Pillac, M. Gendreau, C. Gueret, and A. L. Medaglia, "A review of dynamic vehicle routing problems," *European Journal of Operational Research*, vol. 225, no. 1, pp. 1–11, 2013. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0377221712006388

[97] K. G. Zografos and K. N. Androutsopoulos, "Algorithms for Itinerary Planning in Multimodal Transportation Networks," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 9, no. 1, pp. 175–184, 2008.

[98] K. G. Zografos, K. N. Androutsopoulos, and V. Spitadakis, "Design and Assessment of an Online Passenger Information System for Integrated Multimodal Trip Planning," *Intelligent Transportation Systems, IEEE Transactions on*, vol. 10, no. 2, pp. 311–323, 2009.

[99] L. Hame and H. Hakula, "Dynamic journeying in scheduled networks," *IEEE Transactions on Intelligent Transportation Systems*, vol. 14, no. 1, pp. 360–369, Mar 2013.

[100] L. Häme and H. Hakula, "Dynamic journeying under uncertainty," *European Journal of Operational Research*, vol. 225, no. 3, pp. 455–471, Mar 2013.

[101] R. D. Zimmerman, C. E. Murillo-Sanchez, and R. J. Thomas, "MATPOWER: Steady-State Operations, Planning, and Analysis Tools for Power Systems Research and Education," *IEEE Transactions on Power Systems*, vol. 26, no. 1, pp. 12–19, 2011. [Online]. Available: http://dx.doi.org/10.1109/TPWRS.2010.2051168

[102] J. Ash and D. Newth, "Optimizing complex networks for resilience against cascading failure," *Physica A: Statistical Mechanics and its Applications*, vol. 380, pp. 673–683, Jul. 2007. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0378437107002543

[103] P. Holme, B. Kim, C. Yoon, and S. Han, "Attack vulnerability of complex networks," *Phys. Rev. E*, vol. 65, no. 5, pp. 56 101–56 109, May 2002.

[104] R. Albert, H. Jeong, and A.-L. Barabási, "Error and attack tolerance of complex networks," *Nature*, vol. 406, no. 6794, pp. 378–382, Jul 2000.

[105] D. Rowell and D. N. Wormley, *System dynamics : an introduction*. Upper Saddle River, NJ: Prentice Hall, 1997.

[106] W. Schoonenberg and A. M. Farid, "A dynamic production model for industrial systems energy management," in *2015 IEEE International Conference on Systems Man and Cybernetics*. Hong Kong, 2015, pp. 1–7.

[107] A. Viswanath and A. M. Farid, "A Hybrid Dynamic System Model for the Assessment of Transportation Electrification," in *American Control Conference 2014*, Portland, Oregon, 2014, pp. 1–7. [Online]. Available: http://dx.doi.org/10.1109/ACC.2014.6858810

[108] W. N. Lubega and A. M. Farid, "Quantitative Engineering Systems Model & Analysis of the Energy-Water Nexus," *Applied Energy*, vol. 135, no. 1, pp. 142–157, 2014. [Online]. Available: http://dx.doi.org/10.1016/j.apenergy.2014.07.101

[109] A. M. Farid, "Electrified transportation system performance: Conventional vs. online electric vehicles," in *Electrification of Ground Transportation Systems for Environment and Energy Conservation*, N. P. Suh and D. H. Cho, Eds. MIT Press, 2015, ch. 22, pp. 1–25.

[110] W. C. Schoonenberg and A. M. Farid, "A Dynamic Energy Management Model for Microgrid-Enabled Production Systems," *submitted to: IEEE Transactions on Industrial Informatics*, vol. 1, no. 1, pp. 1–7, 2015.

Amro M. Farid received his Sc.B and Sc.M degrees from MIT and completed his Ph.D. degree at the Institute for Manufacturing within the University of Cambridge Engineering Department in 2007. He is currently an assistant professor of Engineering Systems and Management and leads the Laboratory for Intelligent Integrated Networks of Engineering Systems (LIINES) at Masdar Institute of Science and Technology, Abu Dhabi, U.A.E. He is also a visiting scientist at the MIT Mechanical Engineering Department. His research interests address the systems engineering of intelligent energy systems including smart power grids, energy-water nexus, transportation electrification, and industrial production. He is a senior member of the IEEE and is actively involved in the Control Systems Society, the Systems, Man & Cybernetics Society, and the Industrial Electronics Society.