Chapter 14 Design of Large Engineering Systems

Gyung-Jin Park, Amro M. Farid

Summary

One defining characteristic of twenty-first century engineering challenges is the breadth of their scope. The National Academy of Engineering (NAE) has identified 14 "game-changing goals". At first glance, each of these aspirational engineering goals is so large and complex in its own right that it might seem entirely intractable. Fortunately, design science provides a continually advancing perspective built upon a meta-problem-solving skill set. This chapter introduces the design engineer to the world of large complex systems from an Axiomatic Design perspective. In particular, the chapter focuses on two critical "systems-thinking" design skills that help the designer manage the inherent and abstract complexity of large systems. They are 1.) system decomposition and 2.) the allocation of function to form. The chapter also practically demonstrates these design skills in several design stories and case studies. The chapter discusses why and how these design skills are used differently when the system has a fixed versus flexible structure. Finally, the chapter concludes with several avenues for further investigation.

14.1 Introduction

There are two kinds of large systems. The first kind is a tree-like large system. It starts with a limited number of FRs and DPs at the highest-level but requires many layers of decomposition for actual implementation, involving a large number of lower-level FRs to satisfy the limited number of the highest level FRs. For example, the original idea for making a dispersion-strengthened copper alloy had only a limited number of FRs. It consisted of a pure copper matrix and a plethora of nano-scale titanium di-boride particles as dispersoids. The process devised was the high-speed, isothermal impingement mixing of Cu/Ti solution with Cu/B solution, followed by rapid cooling and shredding of solid ribbon of the alloy, which was compacted through hydrostatic compaction/extrusion of the alloy. The number of FRs at the highest level was relatively small. However, when the process design was completed to manufacture the alloy commercially, the manufacturing system became quite complicated because many lower-level FRs were introduced to implement the highest-level FRs.

The second kind of large system has a large number of FRs at the highest level. For example, if we are designing an airport for a major hub, there are many highest-level FRs. These FRs, in turn, generate many lower-level FRs, making the system very large. This chapter deals with this second kind of large systems, although the methodology and theory are equally applicable to the first kind of large system.

14.1.1 Motivation

One defining characteristic of twenty-first century engineering challenges is the breadth of their scope. The National Academy of Engineering (NAE) has identified 14 “game-changing goals”:

1. Advance personalized learning
2. Make solar energy economical
3. Enhance virtual reality
4. Reverse-engineer the brain
5. Engineer better medicines
6. Advance health informatics
7. Restore and improve urban infrastructure
8. Secure cyber-space
9. Provide access to clean water
10. Provide energy from fusion
11. Prevent nuclear terror
12. Manage the nitrogen cycle
13. Develop carbon sequestration methods
14. Engineer the tools of scientific discovery

At first glance, each of these aspirational engineering goals is so large and complex in its own right that each might seem entirely intractable. Furthermore, each goal might appear so different from the next that an aspiring engineer might easily conclude that the skills needed to solve one challenge are entirely distinct from those of another. Consequently, our engineering education system would have to turn "on a dime", orient itself towards each of these 14 challenges, and ask our first-year engineering students to commit themselves to one of these challenges; never to change direction again. And in the unlikely event that we are successful on such a course, the engineering education system would have to pivot again years later to address the newly cropped-up grand challenges.

Fortunately, design science provides an alternative and continually advancing perspective. While each of these aspirational NAE goals might seem entirely different, in reality, they exhibit many common characteristics which can be integrated into a consistent design framework. In time, this design framework increasingly spans individual engineering disciplines and real-life problem domains. It also increasingly sets aside limiting paradigms and assumptions in its quest towards a refined meta-problem-solving skill set.

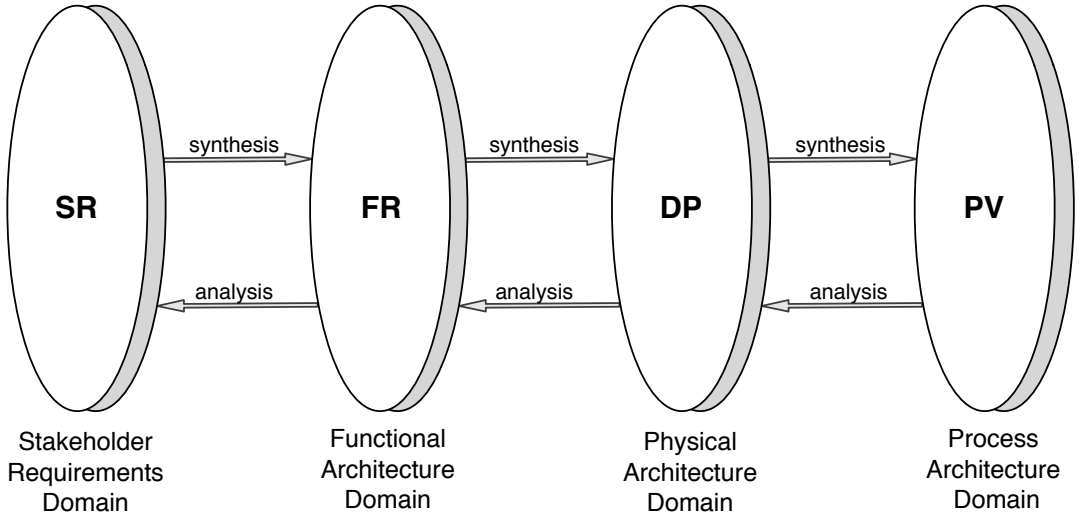


Figure 14.1. Four Domains in the Engineering Design of Systems – An Axiomatic Design Perspective

A Design Story: (Note: We should put chapter number in front of the figure and table numbers to minimize confusion.)

The design engineer reconsiders the 14 NAE challenges in the context of the four Axiomatic Design engineering domains shown in Figure 14.1.

- First, they recognize that the "design solution" to each of the grand challenges can be viewed as a newly designed ***large complex system*** that is very much a refined version of the large complex system that exist in its place today.
- Second, they realize that unlike "traditional products", these systems have not just one "customer" in the stakeholder requirements (SR) domain but rather a ***diversity of internal and external stakeholders***. These stakeholders impose a wide variety of hard and soft requirements which must ultimately be resolved into the functional requirements (FRs) and constraints (Cs) of the functional architecture domain.
- There, the aspiring design engineer finds that large complex systems are characterized by a large ***number*** of FRs. Managing such a large number is a psychological and organizational challenge in its own right.

- Beyond just number, the design solution is complicated by the ***heterogeneity*** (or diversity) of the FRs. Functions are closely tied to siloed engineering disciplines. At their simplest, an electrical engineer knows electricity, a chemical engineer knows chemical reactions, and a mechanical engineer knows classical mechanics. A diversity of function means that the design engineer must either absorb the functions associated with other disciplines themselves or work effectively with other design engineers who have.

- It is at this point, when the design engineer crosses the synthesis path from the functional architecture domain to the physical architecture domain and its associated design parameters that they face their greatest challenge: ***creativity***. The word strikes fear in the hearts of many young designers. As young designers ourselves, we recall being perpetually in awe of how our design professors seemed to magically come up with design solutions from "thin air". And yet, creativity is not a binary characteristic encoded in DNA. Rather, it is cultivated by a willingness to loosen and expand one's established mental constructs; whether by picking up new books or immersing oneself amongst and appreciating the perspectives of a broad diversity of people. Each person, their field, and personal set of experiences can be characterized by a set of "mental constructs" that serve as "meta-design-parameters" that the design engineer deploys creatively in the moment of synthesis. The design engineer soon recognizes that the 14 NAE goals are indeed challenges for the specific reason that they require a broad number and diversity of such "meta-design-parameters". No single design engineer will have them all, but design teams can produce maximally effective large complex systems by fostering an environment where new ideas are encouraged and easily communicated.

- As the design engineer implements their solution and crosses over to the process architecture domain, they find that the large complex system has already been implemented –- at a cost of millions, billions, or even trillions of dollars -- poorly no less -- in the form of the ***legacy system***! The likelihood of implementing their "forward-engineering design" is low. Despair can set in. But with newfound energy, the design engineer sets off in the reverse direction; now along the "analytical path", "reverse-engineering" the legacy system. They task themselves with modeling the PVs, DPs, FRs, and SRs of the legacy system; knowing full well that they will have to make a sequence of meaningful piecewise transformations towards an "ideal" design solution that reconciles their forward design with the existing system all while it remains operational.

14.1.2 Chapter Contribution

This chapter introduces the design engineer to the world of large complex systems from an Axiomatic Design perspective. It would be entirely intractable to try to address ***all*** of the specific challenges mentioned in the large complex system design story relayed above. Rather, this chapter focuses on two critical "systems-thinking" design skills that help the designer manage the inherent and abstract complexity of large systems. They are 1.) system decomposition and 2.) the allocation of function to form. The chapter also practically demonstrates these design skills in several tractable design case studies. The chapter also discusses why and how these design skills are used differently when the system has a fixed versus a flexible structure. The chapter concludes with a discussion of several open challenges in the design of large complex systems.

14.1.3 Chapter Outline

The remainder of the chapter is organized as follows. Section 14.2 treats the Axiomatic Design of large fixed systems. Section 14.3 then provides several design stories of large fixed systems. Finally, Section 14.4 discusses large flexible systems and contrasts them with the large fixed systems discussed earlier in the chapter.

14.2 Axiomatic Design of Large Fixed Engineering Systems

14.2.1 What are Large Fixed Engineering Systems?

The previous section used the examples of the 14 NAE challenges to motivate the topic of large complex systems. This section addresses an important subset of these called "large fixed engineering systems". To gain insight, the term must be deconstructed into its constituent words.

There is no shortage of definitions in the literature for the term "system". This chapter adopts the definition below:

Definition 14.1. System: A set of components (subsystems, segments) acting together to achieve a set of common objectives via the accomplishment of a set of tasks.

Note that the mere definition of the term requires the definition of three more abstract systems thinking concepts:

Definition 14.2. System Boundary: The delineation between the system and its environment or context.

Definition 14.3. System Form: What a system "is". It includes a description of:

1. All the system's components (or design parameters).
2. How the components are interconnected .
3. What portion of the total system behavior/function is carried out by each component.

Definition 14.4. System Function/Behavior: What a system "does". It is its reason for existence. A set of subfunctions (or functional requirements) that must be performed to achieve a specific objective.

Returning to the Axiomatic Design framework shown in Figure 1, the system form constitutes the physical architecture domain and the system function constitutes the functional architecture domain.

"Large systems" are referred to as such because they have a "large" number of system elements; be they FRs or DPs. How large is a "large number"? For all practical purposes, the answer is driven by the psychological limitations of the human mind as it attempts to design the system. In 1956, Miller recognized that human beings can typically recall 7 +/- 2 numerical digits in short term memory. Consequently, as a rule of thumb, the literature refers to systems with approximately 7 elements as "small-sized", $7^2 \approx 50$ elements as "medium-sized", and $7^3 \approx 300$ elements as "large-size". Interestingly, at $7^4 \approx 2500$ elements or greater, the system can no longer be designed practically by a single designer (or system architect). Instead, multiple designers or design teams with their respective responsibilities must cooperate to produce a well-functioning large system.

The mere existence of multiple designers implies that they must manage the interdependencies between the systems' elements. The term "complex system" is often used to refer to systems with a large number of interdependencies between its constituent elements. For clarity, it is useful to distinguish between two types of such interdependencies:

Definition 14.5. Interactions: An interdependency caused by the sequence of one functional requirement followed by another.

Definition 14.6. Interfaces: An interdependency caused by the relationship between one design parameter and another. For example, there may exist a flow of matter between two components.

One can imagine that a system with N elements can have up to N^2 interdependencies. Consequently, a large system by virtue of its number of elements is likely to be complex as well by virtue of the interdependencies between these elements.

As the large complex system grows in size, potentially over many years, it is likely that its elements and their interdependencies will change. Consequently, Axiomatic Design distinguishes between "large fixed engineering systems" and "large flexible engineering systems".

Definition 14.7. Large fixed engineering system: An engineering system with a large set of functional requirements which do not evolve over time and whose components (DPs) also do not change over time.

Definition 14.8. Large flexible engineering system: An engineering system with many functional requirements that not only evolve over time, but also can be fulfilled by one or more design parameters.

Finally, this book is devoted to "engineering systems" rather than "systems" broadly; in that the latter includes (natural) systems (e.g. the solar system, the human body) that are not engineered but still adhere to the three minimal requirements of being a system stated above.

14.2.2 Divide and Conquer: Decomposition of System Hierarchy

The first critical "systems-thinking" design skill is decomposition of the system hierarchy. Here, the engineer must use a "divide and conquer" mentality to manage the design of the large system. In brief, the system's elements; be they functional requirements or design parameters must be decomposed into their constituent parts.

For the moment, let's assume a large fixed engineering system. By Definition 14.7, it has many functional requirements. Following Figure 14.1, these were identified in a requirement engineering process where the requirements of the stakeholder requirements domain were transformed into the high-level functional requirements of the functional architecture domain.

To recall, the functional requirements (FRs) describe what the system must do and sometimes they are simply referred to as the systems' functions to describe what the system does. Said differently, they are a mutually exclusive and collectively exhaustive set that describe the system functionally. Each FR must be defined in a solution-neutral way that doesn't presuppose the technologies of the design solution. By convention, each function is defined as a transitive verb stated in the third person singular followed by its associated object/operand. For example, in a home design, one functional requirement may be:

FR1: Protect Internal Climate.

There is a lot packed into such a "high-level" functional requirement. For this reason, it is often necessary to ***decompose*** each function into a number of lower-level of functions that ***aggregate*** to achieve the same end. In decomposing a given functional requirement, the designer must be careful to keep the lower-level FRs mutually exclusive and collectively exhaustive. For example, FR1 can be decomposed into:

FR1.1 Keep out Moisture
FR 1.2 Damp out Hot/Cold Fluctuations in the External Environment
FR 1.3 Heat and Cool Interior Area to desired temperature

FR 1.4 Redirect falling rain away from the house
FR 1.5 Admit natural light
FR 1.6 Protect from Insects
FR 1.7 Allow entry/exit of inhabitants
FR 1.8 Protect from Intruders

The designer can further decompose each of these FRs into even lower-level FRs until the large system has sufficient functional detail to be fully realized. In an abstract and general sense, this iterative process generates a functional hierarchy as shown in Figure 14.2. Furthermore, in many cases, it is useful to explicitly identify the functional interactions between each system function at each level of functional decomposition. At such a point, the functional hierarchy becomes a complete functional architecture.

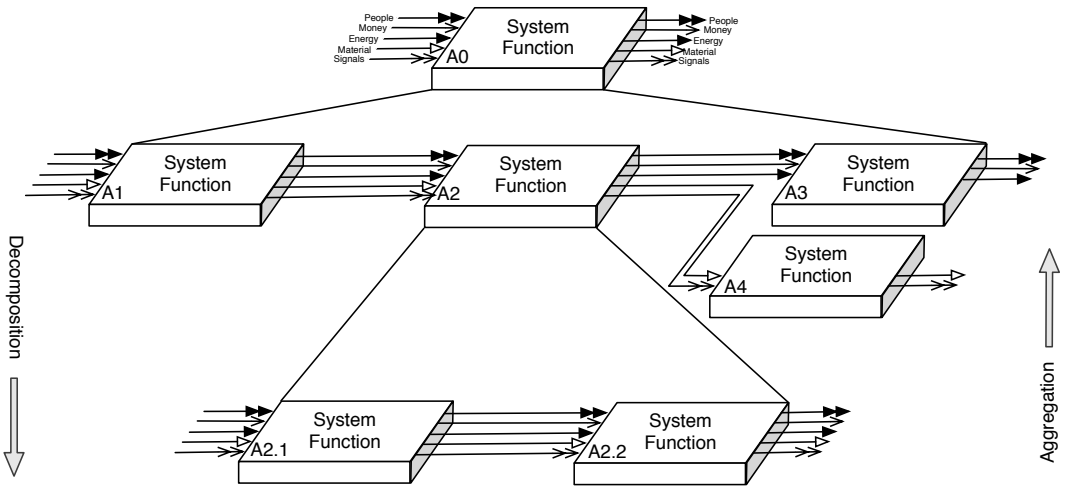


Figure 14.2. A functional architecture that has been decomposed two levels. Parallel and serial interactions are shown between each function.

In an analogous fashion, a large system also has a physical hierarchy composed of multiple decompositions of design parameters (DPs); be they systems, subsystems, components or single numerical parameters. At the highest level of hierarchy, the large system is a single design parameter. For example, in home design,

DP1: House's External Barrier

This high-level design parameter can be decomposed into several lower-level design parameters (as subsystems):

DP1.1 Waterproof Shell
DP1.2 Insulation layer
DP 1.3 Air-source heat pump
DP 1.4 Gutter system
DP 1.5 Window
DP 1.6 Window screens
DP 1.7 Doors
DP 1.8 Door locks

Finally, when the physical hierarchy is depicted with the physical interactions between each design parameter, it becomes the physical architecture. At such a point, it resembles the functional architecture in Figure 14.2; but with the functions replaced with components (or DPs).

Functional and physical decomposition create the Axiomatic Design dual hierarchy shown in Figure 14.3. It serves as the primary means by which a designer tackles the inherently large number of elements (i.e. FRs and DPs) in the system. Rather than view the large system as a loose collection of functions and components, these sets are organized methodically in a tree-like structure. The primary advantage is to reduce the "mental-load" of the designer so that they are only thinking of the elements that are directly related within the associated hierarchy.

One question that often arises is how to organize the functional and physical hierarchies. One may imagine that a given FR or DP can be decomposed all at once into many (potentially hundreds) of elements. Alternatively, one may imagine that the designer organizes these elements into many decomposition layers. In theory, the choice of the number of decomposition steps is arbitrary. In practice, it matters tremendously. For example, the physical hierarchy may be organized into assemblies and subassemblies that have real manufacturing significance. Similarly, design teams may have certain functional expertise that drives how we may think of the functional requirements. Finally, the 7±2 rule mentioned above provides a practical "design rule of thumb" from which designers should not depart too far.

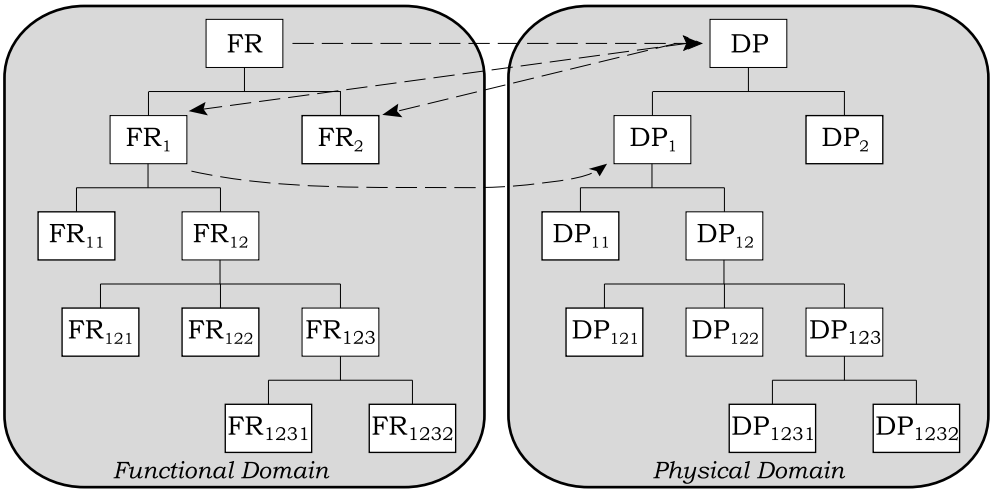


Figure 14.3. The Axiomatic Design Dual Hierarchy: Functional decomposition, Physical Decomposition and the Allocation of Function to Form in a "Zigzagging" Process.

14.2.3 Allocation of Function to Form – the Zigzagging Process

The second critical "systems-thinking" design skill is the allocation of function to form in what is called the "zigzagging process". In the previous section, functional and physical decomposition were presented independently. In reality, a given functional requirement can rarely be decomposed without first assuming some technological solution that fulfills it. In the meantime, decomposing a given design parameter has little meaning without understanding the functional requirements that the decomposed design parameters fulfill. Consequently, in forward-design, the designer considers a given functional requirement, conceives a design parameter to fulfill it, and then allocates the function to this new element of form. At that point, the designer decomposes the functional requirement while assuming the newly conceived design parameter. In such a way, the two critical design skills of decomposition and function-to-form allocation are used in an alternating fashion. A single designer can handle several zig-zigging processes to accommodate several hundred design parameters. Beyond that, a design team can work together to accommodate a system of potentially arbitrary size.

As part of the zigzagging process, Axiomatic Design keeps track of the allocation of function to form using a design equation:

$$FR\$f(DP)$$

where FR is the set of functional requirements on a given level of decomposition, DP is the set of design parameters on the same level of decomposition, and f() is a function representing the laws of physics that govern the design and the relatively new symbol $ means "satisfies" when read from right to left. When the $ symbol is replaced with an = and the first derivative is taken it yields a linear equation:

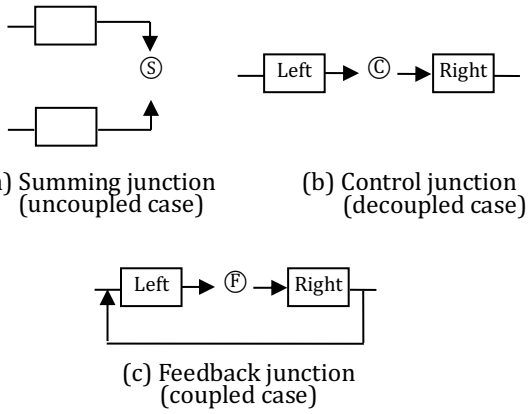
$$\Delta FR = B\,\Delta DP$$

where B is the design matrix. The Independence Axiom instructs designers to ensure that this design matrix is square and diagonal to reach an uncoupled design, and if not then square and lower triangular to reach a decoupled design. When the design matrix is anything else, the design is said to be coupled and it violates the Independence Axiom. For example, the decomposed FRs and DPs mentioned in the previous section yield the design matrix below.

	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8
1.1	X							
1.2	X	X						
1.3	X	X	X					
1.4	X			X				
1.5	X				X			
1.6					X	X		
1.7	X						X	
1.8							X	X

Figure 14.4. Axiomatic Design Matrix of a House's External Barrier

One important consequence of the Independence Axiom is that it affects the project management of the design, or the design workflow. A designer's task is to complete the selection of all the DPs that fulfill the FRs. The design matrix shows the selection of DPs must be taken in groups. The group of DPs that fulfill a given FR is called a ***module***. The design matrix also shows that there is a specific order in which to design each module. In the case of an uncoupled design, the modules can be designed entirely independently in parallel. As shown in Figure 14.5, in a design flow diagram, such a case is defined by a summation (Ⓢ) junction of two or modules. In the case of decoupled design, Figure 5 shows a control (Ⓒ) junction to indicate the design of one module must follow another sequentially. Finally, in the case of a coupled design, Figure 5 shows a feedback (Ⓕ) junction to indicate that design of one module is in an iterative feedback loop with another module. The problem with such feedback loops is that it is not always clear how many design iterations will be required to escape the design feedback! In the design of large systems, such design feedback loops can be entirely debilitating and ultimately bring design progress to a screeching halt! In short, the Independence Axiom does not just facilitate good designs, it also facilitates efficient design workflows.



(a) Summing junction (uncoupled case)

(b) Control junction (decoupled case)

(c) Feedback junction (coupled case)

Figure 14.5. Elements and junctions in a design workflow diagram. Summing, control and feedback junctions correspond to uncoupled, decoupled, and coupled designs respectively.

To summarize, the forward-design zigzagging process follows several rules. The designer must:

1. Conceive the DPs associated with the FRs in the same level.
2. Respect the Independence Axiom when allocating function to form. The design is ideally uncoupled, and otherwise decoupled. If necessary, draw a design workflow diagram to operationalize the sequence of design task amongst the design team.
3. Decompose the FRs of the subsequent level based upon the choice of DPs in the level immediately above.
4. Ensure that the FRs and DPs in a lower level are a mutually exclusive and collectively representation of the FRs and DPs in the level immediately above.
5. Continue the zigzagging process until the Axiomatic Design dual hierarchy includes sufficient detail to implement the design solution.

Zigzagging in reverse-engineering simply runs in reverse. The designer must start from the individual components (or DPs) and then deduce the associated FRs. From there, the designer proceeds to a higher level of aggregation in the DPs and then deduces the associated FRs on that level. Such a reverse-engineering approach is necessary when a part or a whole of the system has already been built and the development of the functional architecture is required to determine how to "evolve" the system forwards to a more advanced stage of development. In some cases, the deduction of the associated functional requirement is straight forward. Many tried-and-true physical solutions have well-known functions. For example, I-beams in buildings support weight, and railways transport trains.

14.3 Examples

The two critical "systems-thinking" design skills of system decomposition and the allocation of function to form are surprisingly powerful tools in the engineer's armament towards the design of large complex systems. This section presents case studies where these two skills are applied practically. The provided examples have been chosen for their manageable size so as to facilitate learning.

14.3.1 Axiomatic Design of the Mount Type Air Conditioning System

The mount type air conditioning system is a type of heating, ventilating and air conditioning (HVAC)

control system that is installed between a ceiling and the ceiling boards of a room to control room temperature. Such a product is investigated from Axiomatic Design point of view. The overall decomposition of the system hierarchy is established down to the "atomic" level components. Some coupled aspects are found; but ultimately kept at the request of the design sponsor. The sponsor wants to keep some of existing component level parts, and so the coupled aspects of the design are reported as warnings in the design process.

The basic operating principle of the mount type air conditioning system is to absorb heat in a room and emit it to the exterior. Figure 6 illustrates the components of the mount type air conditioning system which consists of indoor and outdoor machines. Most of the air conditioning systems employ the refrigeration cycle using refrigerant to generate cool air in a room. Figure 14.7 presents the refrigeration cycle using the refrigerant. In the outdoor part, a compressor generates the flow of refrigerant and a condenser releases heat. The indoor part consists of a capillary tube to control the flow of the refrigerant and an evaporator to absorb the heat in the room. The refrigerant periodically circulates between the two sides so that the indoor heat is absorbed and released outside. Currently, the design is carried out based on the conventional process illustrated in Figure 14.8. Since the machine is not a new one, most of the design components are already known. Therefore, the designer selects appropriate components and sets the values of the associated design parameters. Performance tests are then conducted to meet the requirements. The designer iteratively redesigns the components until the performance requirements are satisfied. The associated work flow is illustrated in Figure 14.9.

(a) Indoor machine (b) Outdoor machine

Figure 14.6. Indoor and Outdoor Machines

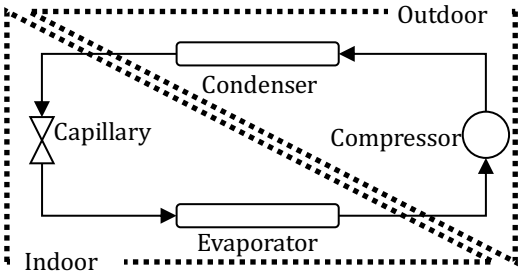


Figure 14.7. Composition of an Air Conditioning System

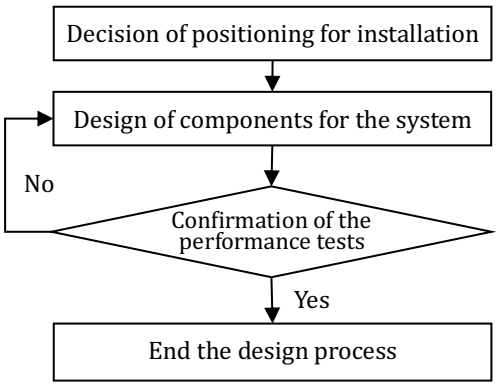


Figure 14.8. The Conventional Design Process for an Overall Air Conditioning System

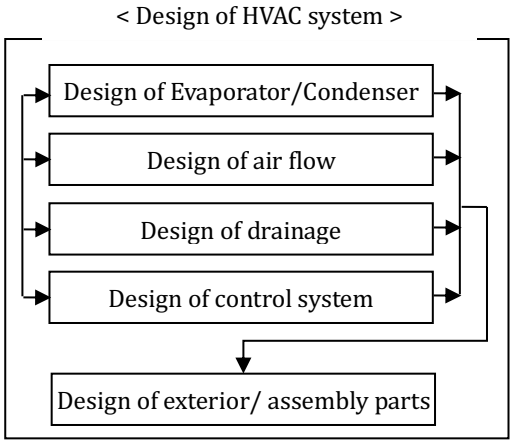


Figure 14.9. The Conventional Design Process for the Design of a Part of an Air Conditioning System

Each customer presents a new set of performance requirements. In such a case, the product should be re-designed and re-manufactured each time to satisfy the new conditions. The conventional design method relies on a trial and error approach to meet the new set of requirements because it is not easy to identify the relationships between the components and their characteristic design parameters. Furthermore, the designer may find it difficult to follow a rational design process that converges towards satisfied functional requirements. Along the way, the designer may exercise engineering judgement that relies on tacit knowledge. Such knowledge is neither written nor systematic and consequently is difficult to transfer to junior designers. All of these challenges make the conventional design process costly and slow.

In contrast, a new Axiomatic Design approach is demonstrated. Customer needs (*CN*s) – a subset of the stakeholder requirements described in Section 14.1 -- are gathered first based on customer surveys and customer interviews with design engineers. The *CN*s are shown in Table 14.1. *FR*s are then identified based on the *CN*s, and *DP*s are selected to satisfy the independence of the *FR*s. The *FR-DP* relationship is made by the design matrix and the dual-hierarchy of the design process is established. Again, the decomposition of the functional and physical hierarchies is made by the zigzagging approach. *CN*s are defined from the customer survey and interviews with design

engineers.

Table 14.1. Customer Needs for a Mount Type HVAC Control System

Customer Needs from General Users	Customer Needs from Design Engineers
People should feel cool.	Dust or bad smell should be eliminated
The room should be cooled as soon as possible.	The maintenance and management of the system should be easy
The cool air must sufficiently circulate.	The system should need a small space and have an aesthetic shape.
Uniform temperature is required.	Regulations must be satisfied.
Each vane should be controlled separately.	Production costs should be minimized.
Temperature control should be easy.	
Control from a distance is required.	
The machine should operate quietly.	
Energy consumption should be minimized	

Based on the *CN*s in Table 1, the *FR*s and constraints (*C*s) are defined at the top level.

FR1= Minimize a possessed space of the mount type air conditioning system.
FR2 = Generate appropriate air current in the room.
FR3 = Make enough cold air.
FR4 = Minimize the vibration/noise of the mount type air conditioning system.
FR5 = Maintain purity of the air quality in the room.
FR6 = Control the temperature under user's directions.

Note that the FRs are stated in a solution-neutral language. All of the nouns in the FRs either 1.) refer to the higher-level DP of the "mount-type air conditioning system" or 2.) refer to nouns in the design context (e.g. air space, air current, cold air, noise, air purity, temperature). Constraints are defined as well. They provide bounds on acceptable design solutions and differ from the *FR*s in that they do not have to be independent.

C1 = Satisfy the first-grade for energy efficiency.
C2 = Satisfy related standards.
C3 = Satisfy the product size to sufficiently insert the air conditioning system between the ceiling and ceiling board.
C4 = Minimize production cost.
C*5* = Make maintenance and repair of the system easy.

To meet the *FR*s, an appropriate set of *DP*s are defined.

DP1 = Ceiling type structure
DP2 = Air current formation system
DP3 = Mutual assistance system
DP4 = Vibration/noise reduction system
DP5 = Air cleaner system
DP6 = Temperature control system

Then the allocation of function to form is captured in the design matrix.

	DP1	*DP2*	*DP3*	*DP4*	*DP5*	*DP6*
FR1	*X*	*0*	*0*	*0*	*0*	*0*
FR2	*X*	*X*	*0*	*0*	*0*	*0*
FR3	*0*	*X*	*X*	*0*	*0*	*0*
FR4	*0*	*X*	*X*	*X*	*0*	*0*
FR5	*0*	*0*	*0*	*0*	*X*	*0*

FR6	*0*	*X*	*X*	*0*	*X*	*X*

where X represents where the FR is being fulfilled by one or more DPs and O represents otherwise. At the top level of decomposition, the design is a decoupled, and the design matrix is lower triangular. Therefore, the independence of *FR*s (in the Independence Axiom) is guaranteed if the *DP*s are determined in increasing numerical order from FR1 to FR6.

The *FR*s and *DP*s at the top level are then iteratively decomposed using the zigzagging process described in the previous section until the bottom or "atomic" level components are reached. For the sake of brevity, a full explanation of the system decomposition is omitted here. The complete design matrix, however, is presented in Figure 10. The left column of the table represents the *FR*s and the upper row includes the corresponding *DP*s. The mount type air conditioning system has a decoupled design at the top level as shown in Equation (14.1). The entire design matrix at its lowest level of decomposition is, however, non-square and the resulting product design is classified as redundant and coupled. This situation is the direct result of the sponsor's request to keep the existing component-level parts.

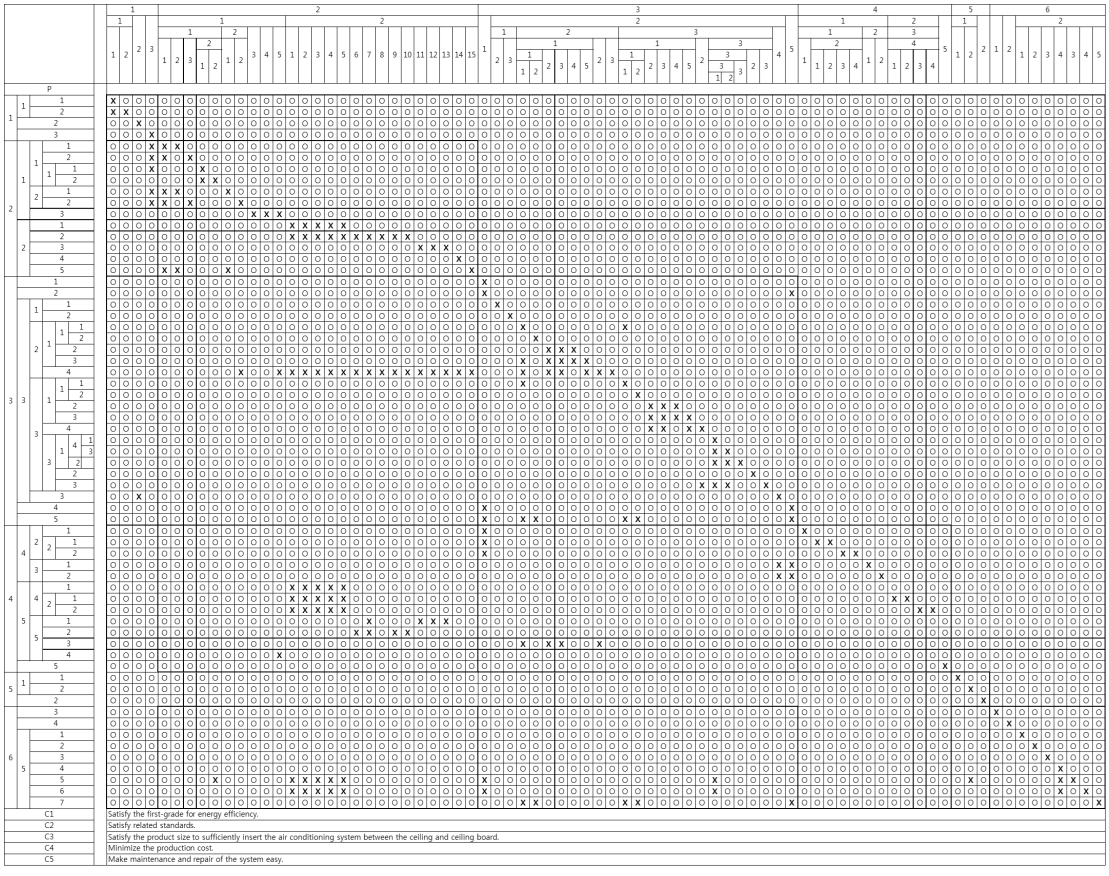


Figure 14.10. Complete Design Matrix for Mount Type Air Conditioning System

The coupled aspects are explained. The entire matrix in Figure 11 is a non-square and consequently some *FR-DP* relationships are coupled at some points of the hierarchy. For example, consider

FR222.

FR2.2.2 = Generate the air-flow in the perpendicular direction using a rotary motion.

The existing system has five corresponding *DP*s that make up the components of the turbofan.

DP2.2.6 = The number of blades
DP2.2.7 = Ratio of the inside/external diameters
DP2.2.8 = The angle among blades
DP2.2.9 = Shape of the blade
DP2.2.10 = Ratio of pitch/chord

Consequently, the relationship between *FR222* and its *DP*s is

	DP2.2.6	*DP2.2.7*	*DP2.2.8*	*DP2.2.9*	*DP2.2.10*
FR2.2.2	*X*	*X*	*X*	*X*	*X*

Obviously, it is a redundant design and a lot of feedbacks are required. The redundancy is also found in other components such as the motor, the orifice, the fin of the evaporator, and the condenser.

Several subsystems within this detailed design warrant further discussion. The *FR*s and *DP*s for the compressor system are defined as:

FR3.1 = Generate the pressure to change the state of the refrigerant.
FR3.2 = Generate the air-flow for the ability of air conditioning to sufficiently cool the room.
DP3.1 = Compressor system

The associate portion of the design matrix is

	DP3.1
FR3.1	*X*
FR3.2	*X*

In this case, the compressor system has an insufficient number of DPs, and its design matrix is full and therefore the compressor subsystem constitutes a coupled design. The capillary tube subsystem suffers from the same problem.

The heat exchanger system has an equal number of FRs and DPs but they are fully coupled.

FR3.3.2 = Change refrigerant from gas to liquid.
FR3.3.3 = Change refrigerant from liquid to gas.
DP3.2.2 = Evaporator system
DP3.2.3 = Condenser system

	DP3.2.2	*DP3.2.3*
FR3.3.2	*X*	*X*
FR3.3.3	*X*	*X*

Because the refrigerator cycle consists of the compressor system, the capillary tube, and the heat exchanger system, the refrigerator cycle as a whole also constitutes a coupled design. Note that only one coupled subsystem is required for the system as a whole to be classified as coupled. In an ideal situation, the DPs of the coupled design should be modified or replaced with a set that yields either an uncoupled or decoupled design. In this case, the firm decided to flag the coupled characteristics to alert designers to the potential for iterative feedback loops in the design process.

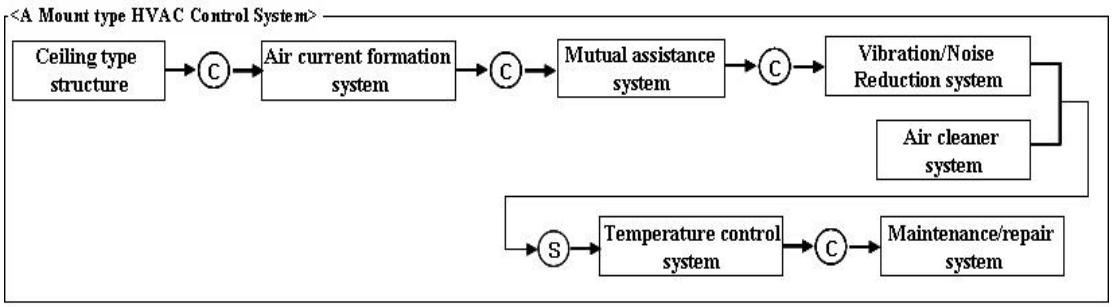


Figure 14.11. Design Flow of a Mount Type Air Conditioning System

Returning back up to the top level of design decomposition, the associated design flow diagram is illustrated in Figure 11. Because it is a decoupled design at the top level of hierarchy, the design flow links the modules sequentially. That said, if the design workflow were detailed further, feedback loops would appear within each of the system's modules.

In summary, this example has served to apply Axiomatic Design for large fixed systems to a mount type air conditioning system. Axiomatic Design demonstrates a rational and effective design process that decomposes the FRs and DPs at each level of hierarchy and then allocates function to form using the design matrix. These two systems thinking techniques allow the design team to manage the complexity of the design. Finally, the design flow diagram organizes the design's project management and is established from the design matrix.

The Axiomatic Design approach highlighted some differences between the existing conventional approach. In the conventional approach, each module was designed in parallel and then later integrated into a larger system; resolving inconsistencies in the design all at once. In contrast, Axiomatic Design highlighted the need for a sequential approach to the design of the high-level modules; thus avoiding the need to resolve downstream design inconsistencies. The improved design process is more effective for designers to not only understand the overall design process but also to meet the diversity of customer demands. Finally, the Axiomatic Design approach highlights the presence of coupled design feedback in lower levels of the dual-hierarchy. These are fairly inefficient and time-consuming and have the potential to derail the design of the system as a whole. These coupled aspects are identified and flagged for management attention without eliminating them from the design as a whole.

14.3.2 Automobile Cooling System

We consider the design of a cooling system for a hybrid vehicle as illustrated in Figure 14.12. Coolant material is circulated in a cooling system to expel the generated heat from the engine. The circulation is illustrated in Figure 14.13. Some of the heat is reused for heating the inside of the car.

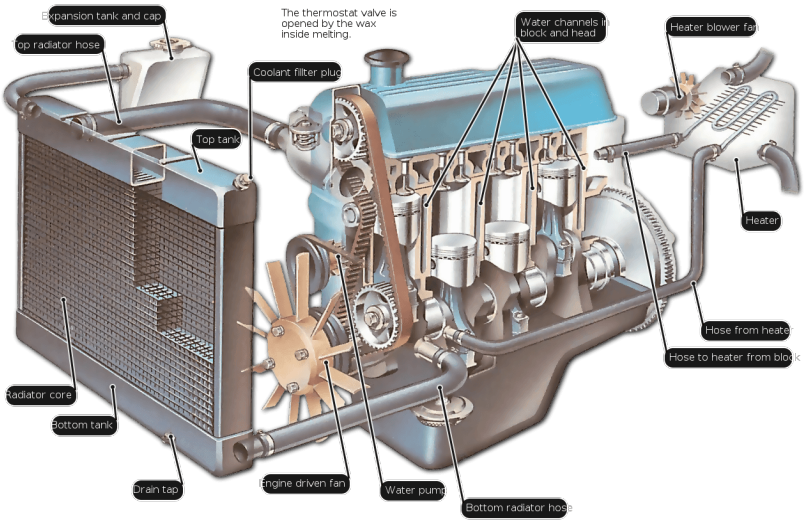


Figure 14.**12. An Automobile Cooling System**

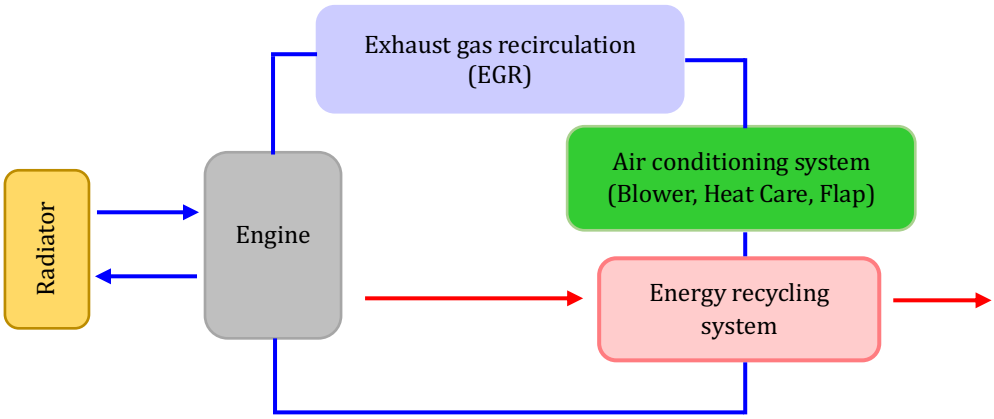


Figure 14.**13. Circuit of Coolant and Air in the Cooling System**

The customer needs (*CN*s) of the system are defined as:

CN1: The inside temperature should be kept within a certain range.
CN2: Environmental pollution is undesired.
CN3: The vehicle fuel economy should be high.

The top-level *FR* is defined based on the *CN*s: Increase the energy efficiency of a hybrid car while maintaining the inside temperature.

From the top-level *FR,* the next level *FR*s, *DP*s, and design matrix are

FR1 = Maintain the temperature of the air inside the vehicle.
FR2 = Reduce the emission of hazardous substances during operation.
FR3 = Control fuel consumption within acceptable bounds.

DP1 = Air conditioning system

DP2 = Hazardous material reduction system
DP3 = Energy management system

	DP1	*DP2*	*DP3*
FR1	*X*	*x*	*x*
FR2	*0*	*X*	*0*
FR3	*x*	*0*	*X*

Note that FR1 and FR3 have a coupled design with respect to DP1 and DP3. This coupling is discussed in greater detail at lower levels of decomposition. *DP1* and *FR1* are decomposed to:

FR1.1 = Decrease room temperature.
FR1.2 = Increase room temperature.
DP1.1 = Air cooling system
DP1.2 = Air heating system

	DP1.1	*DP1.2*
FR1.1	*X*	*0*
FR1.2	*0*	*X*

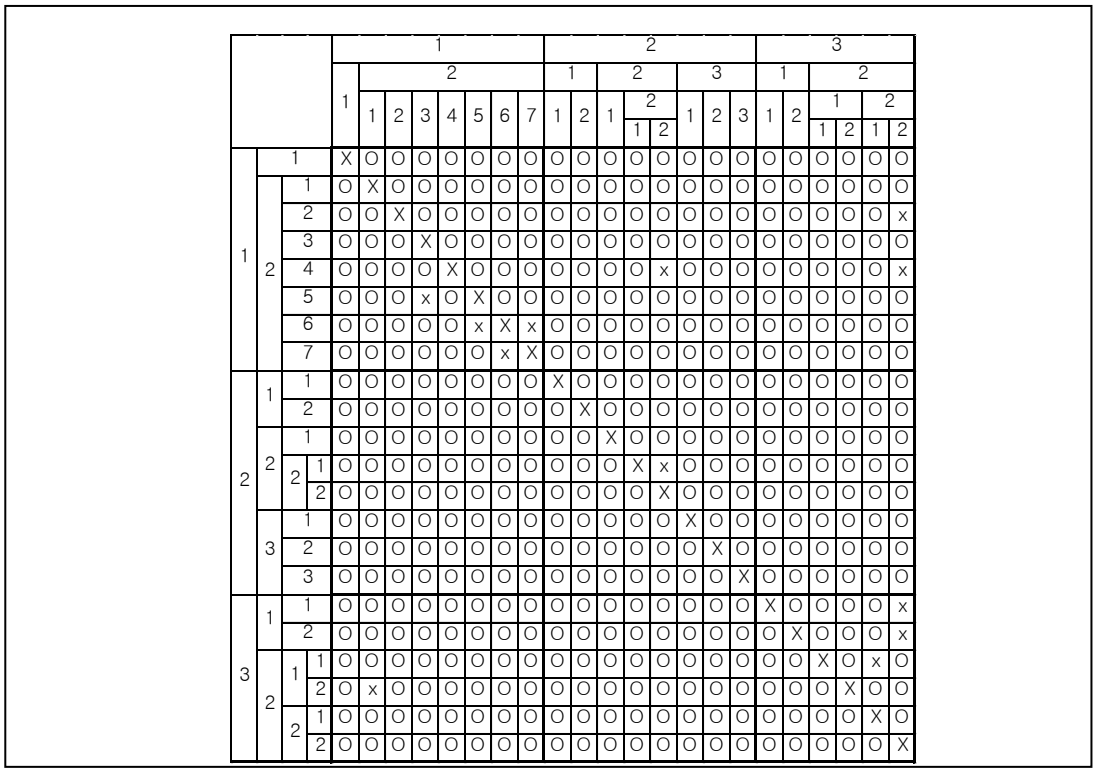
The air cooling system is not considered further in this example. *FR1.2* and *DP1.2* are then decomposed to:

FR1.2.1 = Generate a heat source.
FR1.2.2 = Generate additional heat.
FR1.2.3 = Absorb the generated heat.
FR1.2.4 = Deliver the heat.
FR1.2.5 = Heat the air.
FR1.2.6 = Move the heated air.
FR1.2.7 = Adjust the air direction.
DP1.2.1 = Engine operation request logic
DP1.2.2 = Positive temperature coefficient (PTC) system
DP1.2.3 = Coolant
DP1.2.4 = Coolant circuit
DP1.2.5 = Heat core
DP1.2.6 = Blower
DP1.2.7 = Flap

The associated design matrix is:

	DP1.2.1	*DP1.2.2*	*DP1.2.3*	*DP1.2.4*	*DP1.2.5*	*DP1.2.6*	*DP1.2.7*
FR1.2.1	*X*	*0*	*0*	*0*	*0*	*0*	*0*
FR1.2.2	*0*	*X*	*0*	*0*	*0*	*0*	*0*
FR1.2.3	*0*	*0*	*X*	*0*	*0*	*0*	*0*
FR1.2.4	*0*	*0*	*0*	*X*	*0*	*0*	*0*
FR1.2.5	*0*	*0*	*x*	*0*	*X*	*0*	*0*
FR1.2.6	*0*	*0*	*0*	*0*	*x*	*X*	*x*
FR1.2.7	*0*	*0*	*0*	*0*	*0*	*x*	*X*

FR2 and *DP2* can be decomposed to:



FR21 = Reduce the hazardous material before the combustion.
FR22 = Reduce the hazardous material after the combustion.
FR23 = Reduce the hazardous material during the combustion.
DP21 = Evaporative gas reduction system
DP22 = Engine emission reduction system
DP23 = Catalyst system

The associated design matrix is:

	DP21	*DP22*	*DP23*
FR21	*X*	*0*	*0*
FR22	*0*	*X*	*0*
FR23	*0*	*0*	*X*

FR3 and *DP3* are decomposed as well. Figure 14.14 presents the entire design matrix after several more decomposition and zigzagging steps.

Figure 14.**14. Complete Design Matrix of an Automobile Cooling System Prior to Resorting FRs and DPs.**

At first glance, the design of the automobile cooling system ***appears*** to be highly coupled. There are filled X elements in both the upper and lower triangles of the design matrix. Recall that FR1 and FR3 have a coupled design by virtue of DP1 and DP3 at the highest level of decomposition. These coupled elements manifest themselves in lower levels decomposition as well. The situation, however, is not as bleak as one might think. The high-level coupling between FR1, FR3, DP1, and DP3 does not mean that ***all*** of the low-level FRs and DPs are coupled. Quite a bit of sparsity is introduced into the design matrix with each subsequent decomposition. After careful inspection, the designer finds that many of the filled elements in the upper triangle of the design matrix do not have their associated filled elements in the lower triangle. Indeed, a minimum condition of a truly

coupled design is that there exists two DPs that are coupled to two FRs. Or mathematically, there exists at least one pair of elements in the design matrix DM such that DM(i,j)=DM(j,i) where i≠j. Such a condition occurs in exactly one place above: FR 1.2.6 and FR1.2.7 are coupled with DP 1.2.6 and DP1.2.7. All of the other filled elements in the design matrix are asymmetric and indicate a decoupled design for the remainder of the automobile cooling system. To emphasize this point, the rows and the columns of the design matrix can be sorted in such a way as to bring all of the off-diagonal terms to either the matrix's upper or lower triangle. Figure 15 shows the complete design matrix of the same automobile cooling system after the FRs and DPs have been sorted. It reveals clearly the original conclusion that only FR 1.2.6 and FR 1.2.7 are coupled. This example illustrates that a high-level coupled design can be "rescued" with respect to the Independence Axiom to become a decoupled design in lower levels of decomposition if the nature of the coupling is reflected in an asymmetric design matrix! Fortunately, a creative designer can often find clever ways to introduce such asymmetry into the design.

	1.1	1.2.2	1.2.4	1.2.7	1.2.6	1.2.5	1.2.3	2.1.1	2.1.2	2.2.1	2.2.2.1	2.2.2.2	2.3.1	2.3.2	2.3.3	3.1.1	3.1.2	3.2.1.1	3.2.1.2	1.2.1	3.2.2.1	3.2.2.2
1.1	X																					
1.2.2		X																				X
1.2.4			X									X										X
1.2.7				X	X																	
1.2.6				X	X	X																
1.2.5						X	X															
1.2.3							X															
2.1.1								X														
2.1.2									X													
2.2.1										X												
2.2.2.1											X	X										
2.2.2.2												X										
2.3.1													X									
2.3.2														X								
2.3.3															X							
3.1.1																X						X
3.1.2																	X					X
3.2.1.1																		X			X	
3.2.1.2																			X	X		
1.2.1																				X		
3.2.2.1																					X	
3.2.2.2																						X

Figure 14.**15. Complete Design Matrix of Automobile Cooling System after Resorting FRs & DPs.**

14.3.3 Automobile Suspension System

The Independence Axiom is used to design the automobile suspension system shown in Figure 16. It connects the wheels and the body to make the automobile move. There are two suspension subsystems; one for the front wheels and the other for the real wheels.

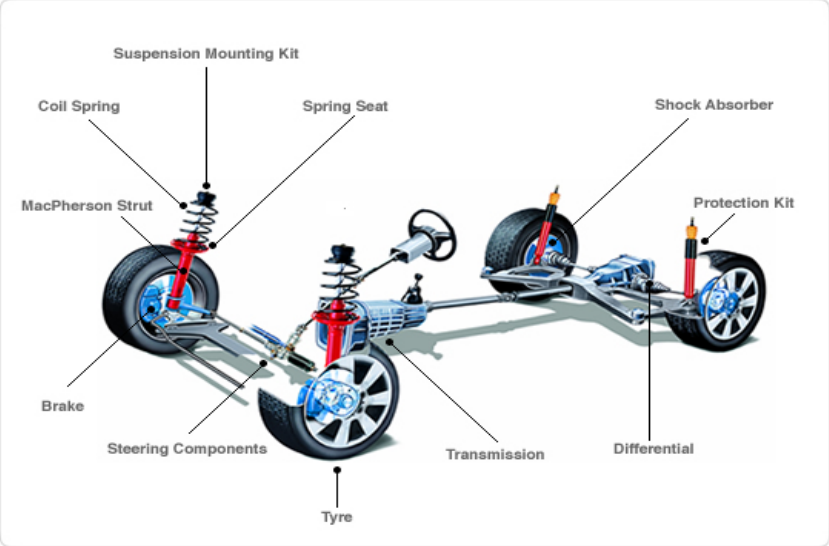


Figure 14.**16. An Automobile Suspension System**

The customer needs (*CN*s) are:

CN1: The vehicle must have a driving capability.
CN2: Safety must be provided to the vehicle when driving.

From the *CN*s, the top-level *FR* is:

FR: Suspend the vehicle with stability while driving (*CN1* + *CN2*)

which is in turn decomposed into six *FR*s:

FR1 = Generate forward thrust from transmission system torque.
FR2 = Permit wheel rotation.
FR3 = Secure the car body.
FR4 = Suspend the vehicle with stability while driving on rugged roads
FR5 = Suspend the vehicle with stability while turning
FR6 = Provide turning ability.

To meet *FR*s, an appropriate set of *DP*s are defined as follows:

DP1 = Wheel system
DP2 = Knuckle & bearing
DP3 = Mounting system
DP4 = Independent link system
DP5 = Stabilizer bar & shock absorber
DP6 = Tie rod & ball joint

The associated design matrix shows a decoupled design at the highest level of decomposition.

	DP1	*DP2*	*DP3*	*DP4*	*DP5*	*DP6*
FR1	*X*	*0*	*0*	*0*	*0*	*0*
FR2	*x*	*X*	*0*	*0*	*0*	*0*
FR3	*0*	*0*	*X*	*X*	*0*	*0*

FR4	*0*	*x*	*x*	*X*	*x*	*0*
FR5	*x*	*0*	*x*	*x*	*X*	*0*
FR6	*0*	*x*	*0*	*0*	*x*	*X*

FR1 and *DP1* are then decomposed.

FR11 = Generate friction force from transmission system torque.
FR12 = Deliver forward thrust from friction force.
DP11 = Tire
DP12 = Wheel

The associated design matrix shows a decoupled subsystem.

	DP11	*DP12*
FR11	*X*	*0*
FR12	*x*	*X*

FR2 and *DP2* are then decomposed and the corresponding design matrix is defined.

FR21 = Impede the motion of the wheel system through five coordinates (3 translation and 2 rotation)
FR22 = Permit the rotation of the wheel system about its axis.
DP21 = Knuckle
DP22 = Bearing

	DP21	*DP22*
FR21	*X*	*x*
FR22	*x*	*X*

FR3 and *DP3* are then decomposed and the corresponding design matrix is defined.

FR31 = Absorb vibration in mounting.
FR32 = Connect the car body.
DP31 = Mounting bush
DP32 = Mounting bolt

	DP31	*DP32*
FR31	*X*	*0*
FR32	*x*	*X*

FR4 and *DP4* are then decomposed and the corresponding design matrix is defined.

FR41 = Absorb vibration and noise at the junctions.
FR42 = Connect the mounting and wheel system.
FR43 = Allow the wheel system to move independently.
DP41 = A,G bush
DP42 = Sub-frame
DP43 = Lower arm

	DP41	*DP42*	*DP43*
FR41	*X*	*0*	*0*
FR42	*x*	*X*	*x*

FR43	*x*	*x*	*X*

FR5 and *DP5* are then decomposed and the corresponding design matrix is defined.

FR51 = Provide rigidity for roll mode.
FR52 = Provide rigidity for the ride mode.
FR53 = Decrease in vertical energy.
FR54 = Limit the vertical distance of motion.
DP51 = Stabilizer bar
DP52 = Spring
DP53 = Damper
DP54 = Bump stopper

	DP51	*DP52*	*DP53*	*DP54*
FR51	*X*	*x*	*0*	*0*
FR52	*x*	*X*	*0*	*0*
FR53	*0*	*x*	*X*	*0*
FR54	*0*	*x*	*x*	*X*

FR6 and DP6 are then decomposed and the corresponding design matrix is defined.

FR61 = Restrain the wheel system to turn.
FR62 = Provide a turning input.
DP61 = Ball joint
DP62 = Tie rod

	DP61	*DP62*
FR61	*X*	*0*
FR62	*x*	*X*

	1.1	1.2	2.1	2.2	3.1	3.2	4.1	4.2	4.3	5.1	5.2	5.3	5.4	6.1	6.2
1.1	X														
1.2	X	X													
2.1		X	X	X											
2.2			X	X											
3.1					X										
3.2					X	X		X					X		
4.1			X		X		X								
4.2							X	X	X						
4.3							X	X	X	X					
5.1							X			X	X				
5.2	X				X		X			X	X				
5.3							X					X			
5.4											X	X	X		
6.1			X										X	X	
6.2														X	X

Figure 14.**17. Complete Design Matrix of an Automobile Suspension System**

The entire design matrix is illustrated in Figure 14.17. Although the suspension system has received many design iterations over a long period of time, there are many and more importantly non-negligible points of coupling between the FRs and the DPs. In the early stage of product development, the suspension system had relatively few and simple *FRs.* Over time, more *FRs* were added, and the design became quite complex with many non-zero off-diagonal terms in the design matrix. Despite this large number, only three pairs of DPs (shown in red above) have coupled designs. They are DP 2.1 and DP2.2, DP 4.2 and 4.3, and DP5.1 and 5.2. Although these pairs of DPs formally violate the Independence Axiom, the situation is quite manageable. These three pairs of DPs can be treated as three DPs; with each pair being treated as a single entity. Furthermore, the DPs appear at the bottom of the Axiomatic Design dual hierarchy, and so their coupled nature does not "ripple" across the design of the rest of the suspension system. Finally, product design companies can find it beneficial to have highly coupled components at the bottom of the dual hierarchy because they effectively embody proprietary and potentially secret design know-how that is not easily reproduced. That said, this special condition should not be interpreted as a license to ignore the Independence Axiom. The product's overall dual-hierarchy should remain uncoupled or decoupled (as above) and only if necessary introduce coupled DPs at the very bottom of the hierarchy.

14.3.4 Mobile Harbor

As the world-wide volume container shipments increases and very large container ships emerge as a dominant player in the maritime cargo transport market, the functional capabilities of container ports need to be greatly enhanced. For example, large container ships with streamline hulls lack maneuverability in tight and highly suggested maritime ports and often require tug boats which may in and of themselves be otherwise occupied. To address this problem, the Korea Advanced Institute of Science and Technology (KAIST) is undertaking a project to design a novel container transport system called Mobile Harbor (MH). A conceptual illustration of the mobile harbor is presented in Figures 14.18 and 14.19. Mobile Harbor refers to a system that can go out to a large container ship, anchor in the open sea, load and unload containers between itself and the container ship, and transport them to their destination. It has a flat bottom design so that it is maneuverable in narrow shipping lanes and stable enough to handle a wide variety of ocean-going cargo. Like many other large-scale engineering projects, the design of Mobile Harbor presents a number of challenges at the beginning stages of the project. In the conceptual design phase of the project, the design team must properly define and disseminate functional requirements, clarify interface requirements between its subsystems, identify and reconcile potential design conflicts like functional coupling.

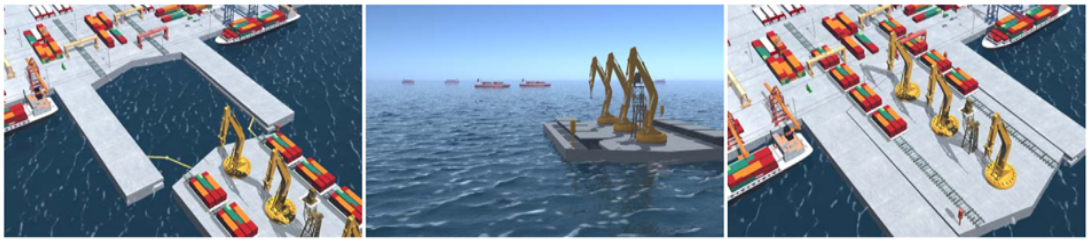


Figure 14.**18. Illustration of a Revised Mobile Harbor Concept**
(*approved for funding by a national R&D program***)**

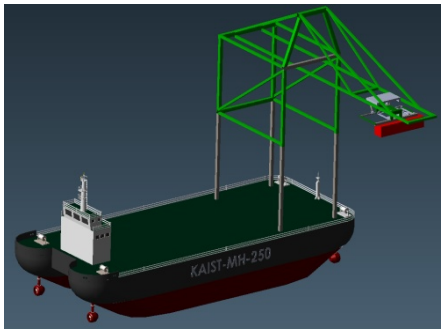


Figure 14.**19. The Mobile Harbor as Conceptualized by Axiomatic Design**

To begin the Axiomatic Design, the top-level *FR*s, *DP*s, and constraints are defined.

FR: Transfer containers from ships anchoring in the sea to a harbor.
FR1 = Load/unload the containers.
FR2 = Keep the containers in MH (Mobile Harbor).
FR3 = Let MH float on the sea.
FR4 = Let MH navigate on the sea.
FR5 = Dock MH against the container ship in the open sea.

DP: Mobile harbor (MH)
DP1: Crane system
DP2: Deck system for loading the containers
DP3: Floating body structure
DP4: Driving system <propulsion + navigation>
DP5: Mooring system + Docking system

Constraints

C1: The capacity of the power generator should be large enough to cover the needed energy. (Generate a certain power for operating.)
C2: The total weight should be less than a specified value.
C3: The MH carries the containers up to 250 TEU.
C4: Safety, reliability, efficiency, cost-effective, environmentally acceptable, automation, the satisfaction of standard and shipping registration or other rules.

According to the relationship between *FR*s and *DP*s, the design matrix is defined as follows:

	DP1	*DP2*	*DP3*	*DP4*	*DP5*
FR1	*X*	*x*	*0*	*0*	*0*
FR2	*x*	*X*	*X*	*0*	*0*
FR3	*x*	*x*	*X*	*0*	*0*
FR4	*x*	*0*	*x*	*X*	*0*
FR5	*x*	*0*	*x*	*x*	*X*

For illustrative purposes, the Axiomatic Design of DP1 is presented in detail and the completed design matrix of the Mobile Harbor is shown in Figure 19.

Decomposition of *FR1* (*DP1*: Crane system)

FR11: Translocate the containers between MH and a container ship in the open sea or a harbor.

FR12: Load and unload the containers while MH is on the sea.

DP11: Container crane

DP12: Dynamic control system of MH movement

	DP11	*DP12*
FR11	*X*	*0*
FR12	*x*	*X*

Decomposition of *FR11* (*DP11*: Container crane)

FR111: Let the container crane of MH approach to the containers in a container ship or open sea.

FR112: Hold the container.

FR113: Move the container.

DP111: The size and shape of the container crane and moving system of the container

DP112: An appropriate configuration of the crane structure to have sufficient strength

DP113: The hoist device including a trolley

Decomposition of *FR12* (*DP12*: Dynamic control system of MH movement)

FR121: Minimize the movement of the container due to the vibration of the boom while working on the sea.

FR122: Minimize the movement of the container due to the vibration of the hoist while working on the sea.

DP121: Zero moment point (ZMP) system for stability control

DP122: Dynamic control system of the trolley

Cs: The condition of sea state 3 (a wave level) should be satisfied.

	DP121	*DP122*
FR121	*X*	*0*
FR122	*x*	*X*

The mobile harbor constitutes an entirely novel large complex product. It is able to go out to a ship anchored in the open sea to load and unload cargo from it without occupying a pier of a land-based harbor. Such a functionality relieves the cargo ship from having to come into a stationary harbor to unload and load its cargo. Furthermore, the increasing size of container ships means that fewer and fewer harbors have a sufficiently large pier and are sufficiently deep. The Mobile Harbor is also capable of loading and unloading cargo to and from a large ship to the stationary land-based harbor without a container quay by pre-loading the cargo in freight cars placed on the mobile harbor which themselves can be moved onto and removed from the Mobile Harbor. The Mobile Harbor was designed so that it can approach both sides of a cargo ship and attach itself securely; further reducing the time to load and unload cargo. In addition to these uses cases, the Mobile Harbor can be used inland within a canal to allow the crossing of transport containers and goods. One integral part of the design is positioning. Not only must the Mobile Harbor control horizontal translation on the sea and vertical height alignment; it also uses a large gyroscopic apparatus to stabilize rotation even in the presence of large waves.

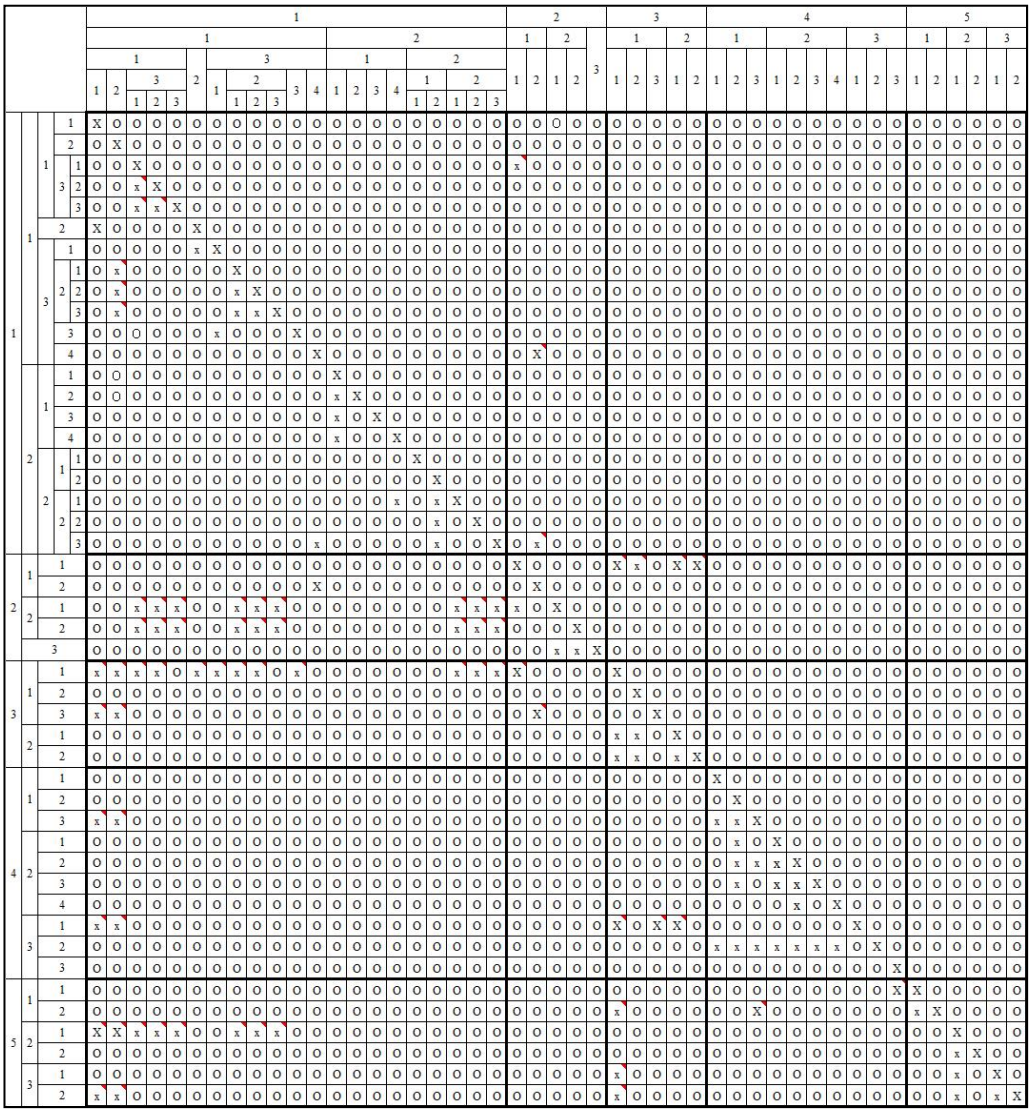


Figure 14.**20. Complete Design Matrix of the Mobile Harbor**

The final design matrix is illustrated in Figure 14.20. It is almost entirely decoupled. FR2 is weakly coupled with FR1 and FR3. Keeping the containers in the mobile harbor has the potential to impact their loading/unloading as well as the overall floating functionality of the mobile harbor. Over most conditions, these coupling are quite weak. They only represent a few design parameters and can be given sufficient design attention up front so as to allow the remainder of the design to proceed. Nevertheless, one must recognize that as the designers pursue ever-more "aggressive" mobile-harbor designs with an ever-larger number of increasingly heavy containers, the coupling will strengthen and eventually lead to design infeasibility. Indeed, the presence of such coupling is entirely inherent to the mobile application and is predicted by Axiomatic Design Theory. The "Design Range and Coupling" theorem in Axiomatic Design Theory states:

> Theorem 20. Design Range and Coupling: If the design ranges of uncoupled or decoupled designs are tightened, they may become coupled designs. Conversely, if the design ranges of some coupled designs are relaxed, the designs may become either uncoupled or decoupled.

Many mobile applications ultimately suffer from such coupling because the mobility of the

application effectively places hard constraints on the design range for space, energy, and functionality. The energy and power capabilities of many robots and drones are limited by their size and weight. Sea-faring vessels must also ensure flotation, and aerospace applications must ensure flight. Finally, the most recent smart city research demonstrates that as an urban population expands, and the demands for water, energy, mobility and other infrastructure services grow, the design and planning of the city's infrastructure systems becomes increasingly interdependent within the city's confined geography. Whereas such couplings can provide the integrated delivery of infrastructure services, they are also susceptible to failures propagating from one subsystem to another. Alternatively, the city's confined geography can be loosened to avoid such a situation; at which point urban sprawl becomes the concern.

14.3.5 The On-Line Electrical Vehicle

The On-Line Electrical Vehicle (OLEV) is a type of electric vehicle that was developed at KAIST to overcome many of the problems posed by conventional electric vehicles: limited driving range on a single charge, the battery's heavy weight, and its high cost. The conceptual configuration is illustrated in Figure 14.21. The OLEV design team is composed of many designers from many engineering disciplines including automobile engineering and electro-magnetics.

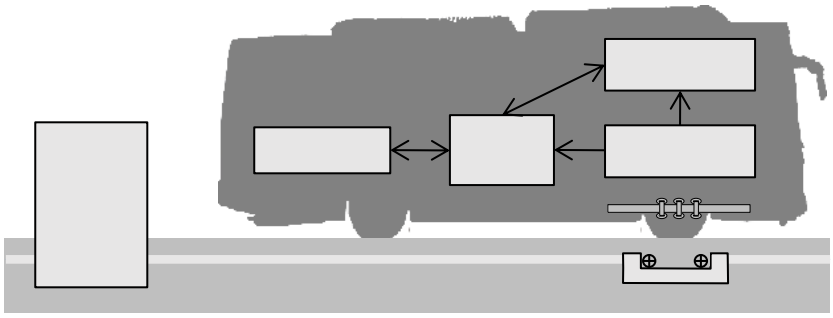


Figure 14.**21. Conceptual Organization of the OLEV**

The integration of these multiple fields presents a significant design challenge. Consequently, the design was conducted by a "design team" and an "integration team". Their roles and workflows are illustrated in Figure 14.22. The integration team defines the FRs, at which point, the design team defines the DPs, at which point the integration team defines the design matrix and defines the new set of decomposed FRs. This alternation between the two teams realizes the zigzagging process on the fly and prevents the inadvertent introduction of design coupling.

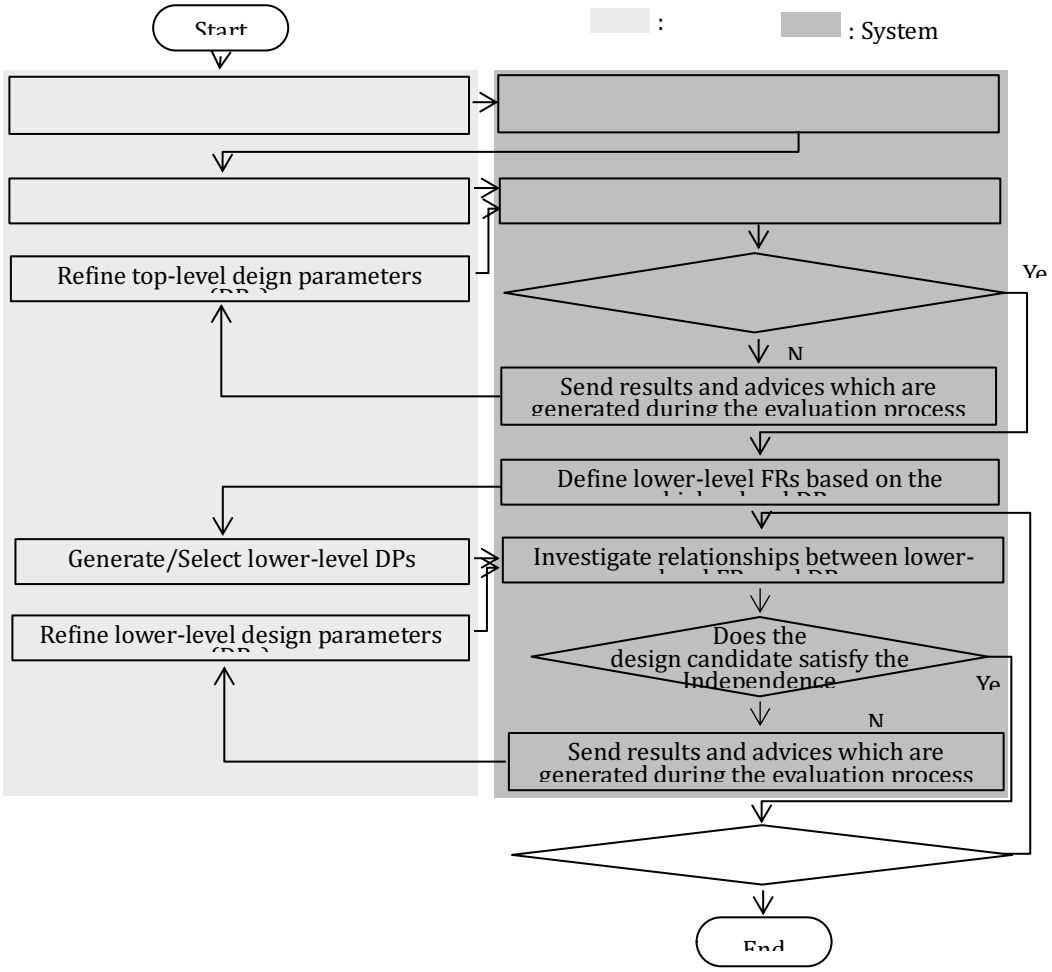


Figure 14.**21. Design Process of the OLEV**

The *FR*s and *DP*s at the top level are

FR1 = Import electric power
FR2 = Transmit electric power to the vehicle.
FR3 = Operate the vehicle using electric power.
FR4 = Protect systems from external loads.

DP1 = Three phase AC 370-440V
DP2 = Induction Coupling System
DP3 = Electric Vehicle
DP4 = System protection devices

From the *FR-DP* relationship, the design matrix is defined.

	DP1	*DP2*	*DP3*	*DP4*
FR1	*X*	*0*	*0*	*0*
FR2	*X*	*X*	*0*	*X*
FR3	*0*	*X*	*X*	*0*
FR4	*0*	*X*	*0*	*X*

Throughout the design process, much effort was exerted to maintained a decoupled design. Nevertheless, the design matrix above shows a high-level coupling of by virtue of the FR2-DP4

coupling. Investigating further, this coupling arises because the system protection devices are coupled to the transmission of electric power to the vehicle. This coupling is ultimately inevitable because the safety considerations of transferring many kilowatts of electricity inductively through an air gap requires a highly integrated design between the induction coupling and the protection devices. From this high-level design matrix, the zigzagging process is carried out until the bottom level is reached. The final design matrix is presented in Figure 14.22. The FR2-DP4 coupling is highlighted to alert the design and integration teams to the potential for design iterations.

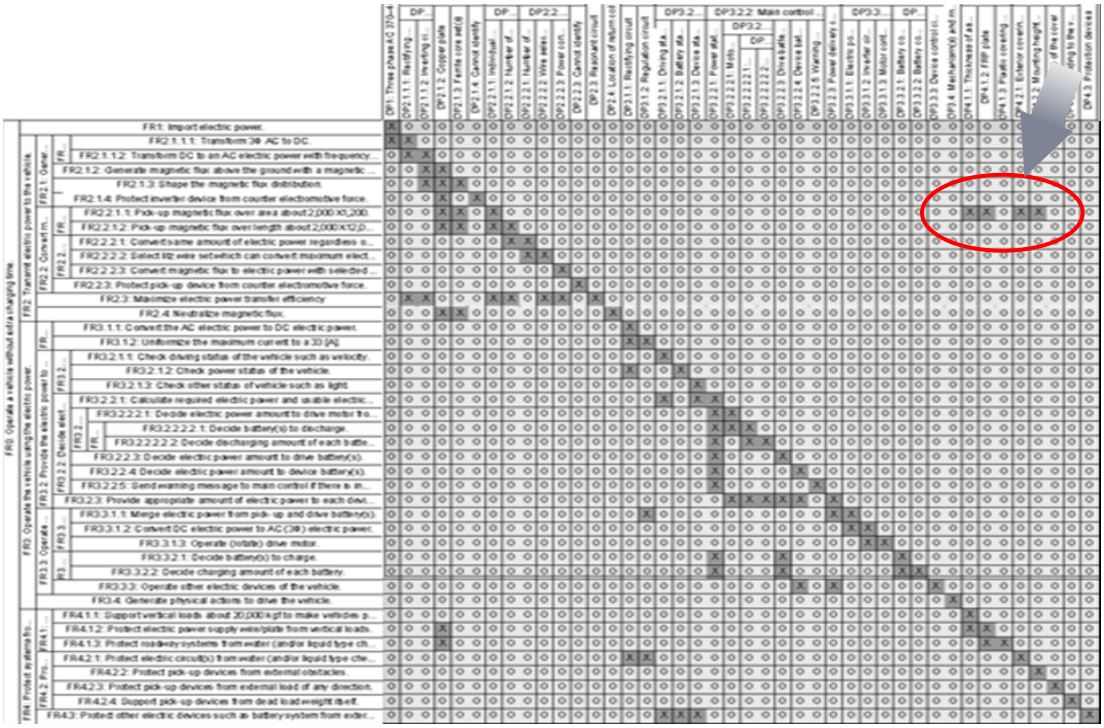


Figure 14.**22. Complete Design Matrix of the OLEV**

This design example, much like the Mobile Harbor example before it, serve to demonstrate the utility of Axiomatic Design in large systems of unprecedented function. The engineers on the project cannot rely on other products or large systems for inspiration. Rather, they must deeply reflect on the what the system must achieve and consequently synthesize the associated FRs and DPs. From there, the two critical systems thinking skills detailed at the beginning of the chapter become the guiding principles of forward-design: decomposition of the functional and physical hierarchy and the allocation of function to form through the zigzagging process. When used in concert, they serve as the primary means by which a designer or design team can tackle the design of large fixed engineering systems.

14.4 Axiomatic Design of Large Flexible Engineering Systems

Thus far, the chapter has motivated the role of design science as a means by which engineers can tackle the 14 grand challenges identified by the National Academy of Engineers. It recognized that design science provides engineers with a meta-problem-solving skill set composed of several systems-thinking design skills. The chapter focused on two of these: decomposition of the system function and form as a means to "divide-and-conquer" and the allocation of function to form in a manner consistent with the Independence Axiom. These two design skills were demonstrated on a half-dozen design examples and in so doing demonstrated their salience to a wide variety of large fixed engineering systems in many application domains.

A curious engineering design student may naturally ask: "What if we wished to design a large flexible engineering system?" (See Definition 14.8 at the beginning of the chapter.). Indeed, many of the NAE grand challenges identified at the beginning of the chapter will require enduring engineering solutions that respond to the changing needs of society. Furthermore, they are likely to require such large investments that it would be entirely cost-prohibitive to design systems from scratch. These design solutions will need to build upon existing legacy systems and will need to continually evolve their functionality over and over again. Unfortunately, the answer to the curious student's question cannot be answered comprehensively in a single book chapter, or perhaps even book, because in many ways it presents a relatively open frontier for the development of design science as an engineering discipline. The literature on the Axiomatic Design of Large Flexible Engineering Systems has expanded over the years and has developed to now be called Hetero-functional Graph Theory; given its heavy reliance on graph theoretic concepts. Nevertheless, this chapter can serve to open the door to this fascinating realm of design science and systems engineering.

The Axiomatic Design of large flexible engineering systems was first mentioned by Suh in his 2001 text. He presented the following abstract example of a large flexible engineering system ***knowledge base***:

$$FR_1\$(DP^a_1, DP^b_{1,\ldots,} DP^r_1)$$
$$FR_2\$(DP^a_2, DP^b_{2,\ldots,} DP^q_2)$$
$$FR_3\$(DP^a_3, DP^b_{3,\ldots,} DP^w_3)$$
$$\ldots$$
$$FR_m\$(DP^a_m, DP^b_{m,\ldots,} DP^s_m)$$

where the first line means that FR1 (as indicated by the \$) can be satisfied by DP^a_1 ***or*** DP^b_1 ***or*** DP^r_1 etc. These few lines are remarkably profound. First, note that the knowledge base does ***not*** say that DP^a_1 ***and*** DP^b_1 ***and*** … ***and*** DP^r_1 satisfy FR1. If it did, it would become the design matrix used throughout the earlier parts of this chapter, and DP^a_1 ***and*** DP^b_1 ***and*** … ***and*** DP^r_1 would form either a decoupled or coupled engineering system. Instead, the focus is on the word "or". In other words, at a given moment in time, DP^a_1 ***alone*** satisfies FR_1. Such a statement is simply another way of stating an adherence to the Independence Axiom. Consequently, the knowledge base above (as presented in the Suh 2001 text) did not allow for decoupled and uncoupled designs. In so doing, it implicitly reached a profound conclusion: large flexible engineering systems ***must*** adhere to the Independence Axiom if they are to maintain their ability to evolve their FRs and DPs over time.

We offer a logical proof by contradiction. In the case of an ideal uncoupled design (with an identity design matrix), the engineer would remove a DP from a functioning system, and an entire FR would be removed at the same time. The system would then continue to function with reduced functionality. In contrast, if a given FR demonstrated a decoupled design and was coupled to multiple DPs, or if the FR demonstrated a couple design and was coupled to other FRs, then when one of the associated DPs were removed, the system would demonstrate broken functionality! Furthermore, removing additional DPs to remove an entire module and its associated FR would lead to downstream effects on other FRs. By Definition 14.8, such a condition contradicts a functioning large flexible engineering system. Indeed, large flexible engineering systems demonstrate a "plug-and-play" functionality similar to that found in modern computers where functional and physical elements can be added or removed at will.

The knowledge base above also serves the design of large flexible engineering systems for two other reasons. First, by Definition 14.8, 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[1], the set of

[1] Note that functional redundancy refers to having the same functional requirement repeated. For example, FR1=generate electric power and FR2=generate (backup) electric power.

functional requirements ***instances*** **FR** must be distinguished from the set of functional requirement ***classes*** $\mathbb{FR}$[2]. Second, the knowledge base allows a single DP to fulfill more than functional requirement class $\mathbb{FR}$. The importance for doing so is discussed later in the context of hierarchy in large flexible engineering systems. In brief, the knowledge base becomes the single most important concept in the design of large flexible engineering systems, in much the same way that the design matrix is important to the design of large fixed engineering systems.

As the Axiomatic Design of large flexible engineering systems developed into hetero-functional graph theory, the knowledge base was given an explicitly quantitative definition.

Definition 14.9 System Knowledge Base: A binary matrix **J** of size σ($\mathbb{FR}$)x σ(**DP**) whose element J(w,v)∈ {0,1} is equal to one when an action e_{wv} (in the SysML sense) exists as a functional requirement class $\mathbb{FR}_w$ being executed by a design parameter DP_v. These actions represent the "capabilities" in the engineering system.

Consequently, the design equation of the large flexible engineering system can be written in terms of the system knowledge base.

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

where $\odot$ represents matrix Boolean multiplication.

Definition 14.10 Matrix Boolean Multiplication $\odot$: Given sets or Boolean matrices **B** and **C** and Boolean matrix **A**, **C=A**$\odot$**B** is equivalent to:

$$C(i,k) = \bigvee_j A(i,j) \wedge B(j,k)$$

where ^ refers to the scalar "AND" operation and $\bigvee$ is an "OR" operation over j elements much like the well-known sigma sum Σ.

The design equation of the large flexible engineering system stated above in no way replaces the design matrix B. In fact, the design equation stated at the beginning the chapter can be written in a set-theoretic expression as well.

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

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

Definition 14.11 Aggregation Operator $\circledast$**:** Given Boolean matrix **A** and sets **B** and **C**, **C=A**$\circledast$**B** is equivalent to:

$$C(i) = \bigcup_j a(i,j) \wedge b(j)$$

Where the $\bigcup$ operation is a union operation of j elements much like the well-known sigma sum Σ.

Functional redundancy should not be confused with "redundant designs" in the Axiomatic Design of Large Fixed Systems were the number of design parameters exceeds the number of functional requirements.

[2] The terms class and instance are drawn from the fields of software/systems engineering. 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.

The engineering design student must recognize that these two statements of the design equation are equivalent in meaning, but must absolutely be distinguished from each other. They simply express the allocation of function to form in terms of the system knowledge base **J** or the design matrix **B**. The fine mathematical distinctions between the aggregation operator $\circledast$ and matrix Boolean multiplication $\odot$ and the set of functional requirement instances FR and the set of functional requirement classes $\mathbb{FR}$ facilitates the analysis of large flexible engineering systems in complementary ways.

Example: To solidify these large flexible engineering systems, consider the example of the simple manufacturing system depicted in Figure 23. 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. The manufacturing system also has two one-way conveyors between them.

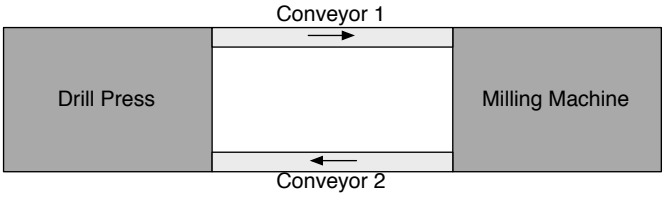


Figure 14.**23. A simple manufacturing system that consists of one drill press, one milling machine and two conveyors.**

A quick analysis of the system yields:

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.

DP={drill press, milling machine drill, milling machine end mill, drill press fixture, conveyor 1, conveyor 2, milling machine fixture}.

Consequently, the design matrix $\mathbf{B=I^{7x7}}$. As expected, the system is uncoupled, and the Independence Axiom is satisfied. To continue the 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}. Rather than viewing the design parameters at the very lowest level of aggregation, it is often useful to aggregate the DPs to a higher level of abstraction. In this case, a logical aggregation yields:
$\overline{DP}$ ={drill press, milling machine, conveyor system}

At this level of physical aggregation, the system knowledge base J is defined.

$$J = \begin{bmatrix} 1\ 1\ 0 \\ 0\ 1\ 0 \\ 1\ 0\ 0 \\ 0\ 0\ 1 \\ 0\ 0\ 1 \\ 0\ 1\ 0 \end{bmatrix}$$

Note that the first column of the knowledge denotes that the drill press is cable of both drilling holes as well as storing the part. The second column shows that the milling machine is cable of doing both of those functions as well as also milling surfaces. The third column shows that the conveyor system is capable of transporting parts back and forth between the two machines. In all

three cases, the columns assigned to each of these aggregated DPs, (often called resources in hetero-functional graph theory), was greater than one because each was capable of more than one function. Conversely, the first row has a sum greater than one to reflect that the function "drill hole" has two instances in the system. In short, the design matrix and the system knowledge base give complementary insights into how function is allocated to form in large flexible engineering systems.

Returning to the original exposition of large flexible engineering systems in the 2001 Suh text, the ***flexible*** nature was emphasized by a set of functional requirements that changed in time.

@t=0	$\{FR\}_0 = \{FR_1, FR_5, FR_7, FR_m\}$
@t=T_1	$\{FR\}_1 = \{FR_3, FR_5, FR_8, FR_z\}$
@t=T_2	$\{FR\}_2 = \{FR_3, FR_9, FR_{10}, FR_m\}$

As the Axiomatic Design of large flexible engineering systems developed into hetero-functional graph theory, this time-dependent system functionality was quantified using a system constraints matrix.

Definition 14.12 System Constraints Matrix: A binary matrix **K** of size $\mathbb{FR}$x**DP** whose element $B(w,v) \in \{0,1\}$ 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. 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***. Quantitatively keeping track of these capabilities is done via the system's structural degrees of freedom as a quantitative measure.

Definition 14.13 Structural Degrees of Freedom[3]: The set of independent actions E_S that completely defines the available capabilities in a large flexible engineering system. Their number is given by:

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

Perhaps the last necessary topic in this introductory discussion of large flexible engineering systems is that of the Axiomatic Design dual hierarchy. When a functional or physical element is added or removed, it has the potential to disrupt their respective hierarchies as well. Simply a speaking, a plane ceases to be one if it were to lose a wing. And its high-level function of flight would be impossible if it loses propulsion. Nevertheless, the designer can proceed cautiously.

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. It is particularly common in bottom-up design to aggregate only the physical hierarchy into higher-level design parameters (or resources). A

[3] The term structural degrees of freedom is appropriately named. Previous works on hetero-functional graph theory have shown that it is a generalization of the well-known concept of degrees of freedom in mechanical systems.

corresponding functional aggregation does not need to occur. Such was the case of the manufacturing system example above. This is because, in bottom-up design synthesis and analysis, physical aggregation and functional aggregation do not have the same meaning and do not necessarily imply each other. 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 many capabilities. An addition or removal of a design parameter yields the corresponding change in a resource's capabilities. In contrast, the functional aggregation of a large flexible engineering system may result in aa 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.

14.5 Conclusion

This chapter has introduced the design engineer to the world of large complex systems from an Axiomatic Design perspective and motivated the pressing need for such design skills in terms of the 14 grand challenges of the National Academy of Engineers. The chapter focused on two critical systems-thinking design skills that help the designer manage the inherent and abstract complexity of large systems. They are 1.) system decomposition as a means to "divide-and-conquer", and the allocation of function to form in what is often called the "zigzagging" process. These two skills were demonstrated in several tractable design case studies of large fixed engineering systems. The chapter also discusses why and how these design skills are used differently when the system has a fixed versus a flexible structure.

In distinguishing between large fixed and flexible engineering systems, this chapter has also opened the door to a fascinating real of design science and systems engineering; one that remains very open frontier for methodological development. Three broad directions of enquiry are worthy of note here.

- A Quantitative Understanding of Life Cycle Properties
- The Treatment of Cyber-Physical Systems
- Hetero-functional Networks in Large Flexible Engineering Systems

First, the field of systems engineering is increasingly concerned with developing a quantitative understanding of life cycle properties which can be deployed within engineering design. Many life cycle properties are called "ilities" because they end with that suffix. Flexibility, sustainability, reconfigurability, maintainability, and interoperability are but a few that follow this grammatical pattern. In the meantime, research into safety, quality, and even stability are well-established in many engineering curriculums as independent design methods. Still other life cycle properties like resilience are relatively new and garner active research interest. These "ilities" are usually classified as non-functional requirements; which are further classified as constraints in Axiomatic Design Theory. Nevertheless, most of these life-cycle properties have a highly integrative nature and consequently their integration into formal engineering design approaches like Axiomatic Design Theory presents a significant intellectual challenge.

Second, most large complex systems have a cyber-physical nature. In addition to the underlying engineering physics, the system as a whole deploys control, automation, and decision-making components that make the system operate efficiently, reliably, and stably. These components can be entirely automated as in the case of PID controllers, or entirely manual as in the case of aircraft pilots. Furthermore, the control, automation, or decision-making system can have centralized, distributed, or decentralized algorithms. Figure 23 contrasts four types of cyber-physical

systems: a.) open-loop physical systems b.) closed-loop cyber-physical systems c.) closed-loop cyber-physical systems a centralized controller and d.) closed-loop cyber-physical systems with a distributed control architecture. Each of these can be modeled as a SysML block diagram, analyzed for its system behavior, or inspected in terms of its underlying system structure. The challenge, here, is that many closed-loop control systems end up creating cyber-physical systems with coupled designs. Consequently, the design workflows develop feedback loops that are not easily managed and prone to design error; particularly for large complex systems with demanding engineering physics. One can perhaps speculate whether the recent news about the Boeing 737 MAX airplanes is the result of the feedback design loops caused by the plane's underlying cyber-physical nature. In any case, we must recognize that as the modern world continues to automate its technologies, it designs cyber-physical systems that are increasingly safety critical. Developing engineering design approaches that target the closed loop nature of cyber-physical systems is imperative.

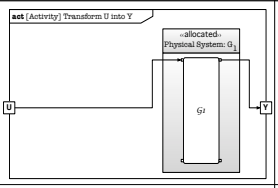
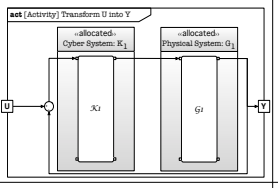
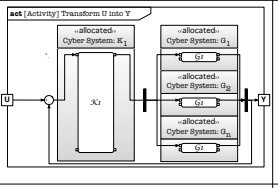
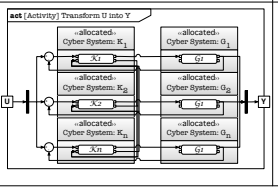
Cyber-Physical System	Activity/Block Diagram	System Behavior	Axiomatic Design
a.) Open-Loop Physical System C0P1 1-Resource		$\mathbf{Y} = \mathcal{G}_1\mathbf{U}$	$\mathbf{Y} = [1] \circledast G_1$
b.) Closed-Loop Cyber-Physical System C1P1 1-Resource 1-Controller		$\mathbf{Y} = \mathcal{G}_{cp}\mathbf{U}$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

Figure 14.**23. Cyber-physical systems from the Perspectives of SysML, Transfer Functions, and Axiomatic Design**

Finally, the study of large flexible engineering systems must continue to develop through hetero-functional graph theory. Three broad communities are actively working to develop the methods to tackle the types of systems that sit at the heart of the NAE's grand challenges. The network science community has deployed graph theory as a means to analyze the ***form*** of large flexible engineering systems. These works, however, are relatively divorced from the established understanding of function within the engineering design field. Furthermore, many of the large-scale architectural transformations of the 21st century appear within the functional architecture or in the system knowledge base, but not in the physical architecture alone. In the meantime, the model-based systems engineering has established graphical modeling techniques like SysML to approach systems of arbitrary size and complexity. However, the quantitative understanding of the underlying system structure has remained elusive. In the meantime, much of the Axiomatic Design community focuses on the quantification of such a system structure, but most investigations have been limited to large fixed engineering systems. To the network science community, hetero-functional networks present a new view as to how to construct graphs with fundamentally different

meaning and insight. To the systems engineering community, hetero-functional graph theory may come to be viewed as a quantification of many of the structural concepts in model-based systems engineering. Finally, to the engineering design student interested in large flexible engineering systems, hetero-functional graph theory presents itself as a natural extension of Axiomatic Design Theory where the Independence Axiom is applied to study systems of flexible structure.

Further Reading

14.1 National Academy of Engineering, "NAE Grand Challenges for Engineering," http://www.engineeringchallenges.org/challenges.aspx, National Academy of Engineering, 2019.

14.2 N. P. Suh, Axiomatic Design: Advances and Applications. Oxford University Press, 2001.

14.3 A. M. Farid and N. P. Suh, Axiomatic Design in Large Systems: Complex Products, Buildings and Manufacturing Systems. Berlin, Heidelberg: Springer, 2016.

14.4 W. C. Schoonenberg, I. S. Khayal, and A. M. Farid, A Hetero-functional Graph Theory for Modeling Interdependent Smart City Infrastructure. Berlin, Heidelberg: Springer, 2018.

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

14..6 E. Crawley, B. Cameron, and D. Selva, System Architecture: Strategy and Product Development for Complex Systems. Upper Saddle River, N.J.: Prentice Hall Press, 2015.