CENTER OF EXCELLENCE
IN SYSTEMS ENGINEERING

INDIANA UNIVERSITY–PURDUE UNIVERSITY FORT WAYNE
AN IPFW CENTER OF EXCELLENCE

A System Design Language Primer

Prof. D.S. Cochran

Revision 1.51
February 19, 2018

Introduction

Axiomatic Design is a design approach used to ensure that the resulting design (system) is both functional and process capable.[1] This methodology utilizes two axioms to guide the design process, and each one serves a key role in the development of a design. The first axiom, the Independence Axiom, guides the decomposition of the design, while the second axiom, the Information Axiom, gives a method to ensure the most capable design is chosen.

- **Design Axiom 1: The Independence Axiom**
  Maintain the independence of the functional requirements (FRs).
- **Design Axiom 2: The Information Axiom**
  Minimize the information content of the design.

A System Design Language

A **System Design Language** provides a means of communicating a design and an approach to determine the effectiveness of a design before it is built. This design language helps to guide the decomposition of the design by asking the right questions at the right time. See Fig. 6 for a visual representation. The process of defining system requirements, communicating these requirements and their chosen solutions, and determining the effectiveness of the system follows the following format:

1. System Boundary

The first step to the system design process is to understand who/what the designer can and cannot control. Fig. 1 provides an example of system boundary diagram. The diagram shows that the product/system being developed has a mechanical and electrical subsystem. The mechanical subsystem must interface with a preexisting mount and customer tooling. The mount and tooling cannot be altered which puts them outside of the system boundary. The electrical subsystem must interface with the mechanical subsystem in addition to a Programmable Logic Unit (PLC) and a user/operator. Therefore, the PLC and the operator are outside of the system boundary.

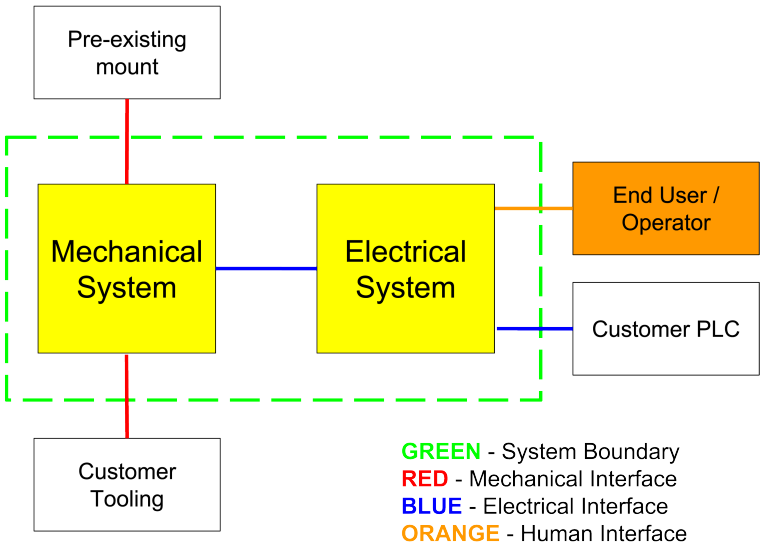


Figure 1: Example of a system boundary diagram

[1] Suh, Nam P. Complexity: Theory and Applications. Oxford: Oxford UP, 2005. Print.

2. Domains

Before mapping the system design, the needs of the customer must be understood. Understanding the needs of the customer and translating them into the functional domain is the first step in defining the requirements of the system. From here, the physical/solution domain is mapped to the functional domain. See Fig. 2 for an overview of the various domains and how they interface with one another. With any system design, the customers and their needs should be clearly defined before defining the requirements of the system.

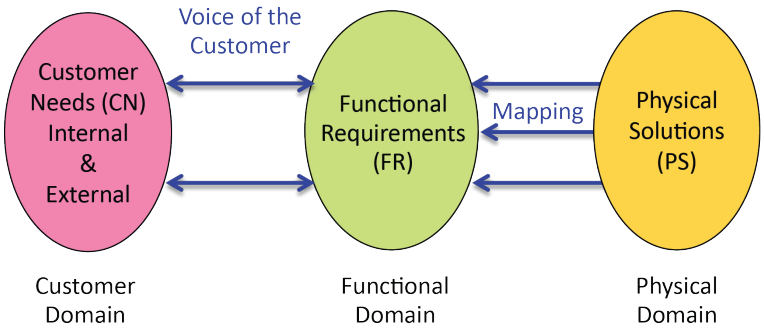


Figure 2: The system design domain

3. Key Questions

- **"What"** must be achieved? $\Longrightarrow$ Functional Requirement **(FR)**
  The FR ALWAYS begins with a **verb**; improve, develop, reduce, provide, etc... FRs represent what the system designers determine the system **must** achieve. Think of an "action" or "transformation" that must take place.
- **"How"** will we achieve the FR? $\Longrightarrow$ Physical Solution **(PS)**
  The PS ALWAYS begins with a **noun**; System, Component Selection, Color Choice, etc...

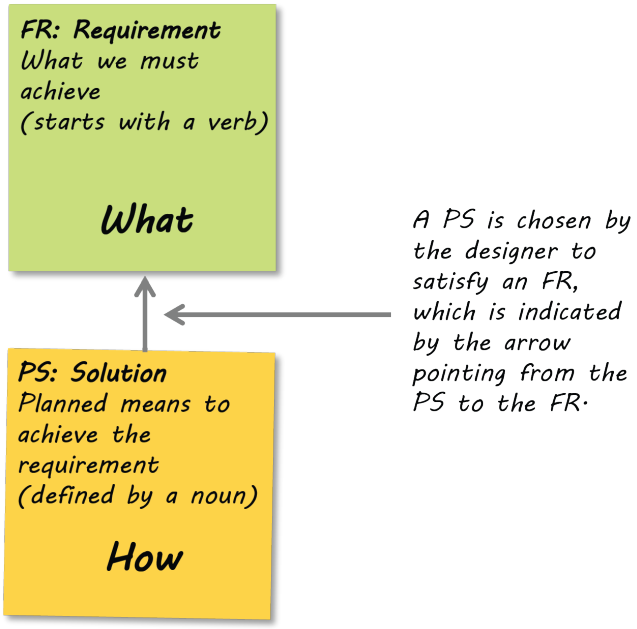


Figure 3: Example of a design relationship

4. Performance Measurement

One aspect of developing FRs is to include a measure or a way to quantify the performance of achieving an FR. The FRm is used to state the expected performance (including unit of measure) to indicate when an FR is achieved. The designer must develop a test plan that ensures that the desired performance can be tested.

- **"FRm"** Performance Measurement tied to the Functional Requirements (as needed).A FRm measures whether or not the design requirement is achieved.
- **"PSm"** Performance Measurement tied to the Physical Solutions (as needed). A PSm measures whether or not the PS is implemented correctly.

5. Alternative solutions

When the designer chooses a PS to satisfy an FR, chances are, reasonable alternatives exist. These alternatives can be presented following the format shown in Fig. 5. Measures for each of these alternatives can be included if necessary. Based on the chosen PS, the resulting design may be different.This idea is explored in more detail in Fig. 8.

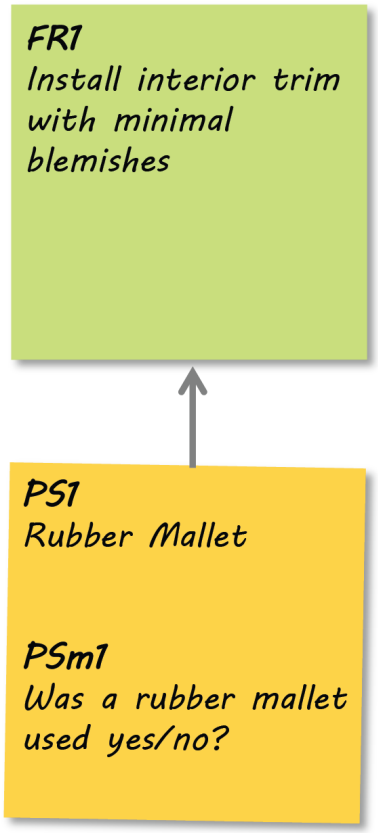


Figure 4: Selection of FR_m

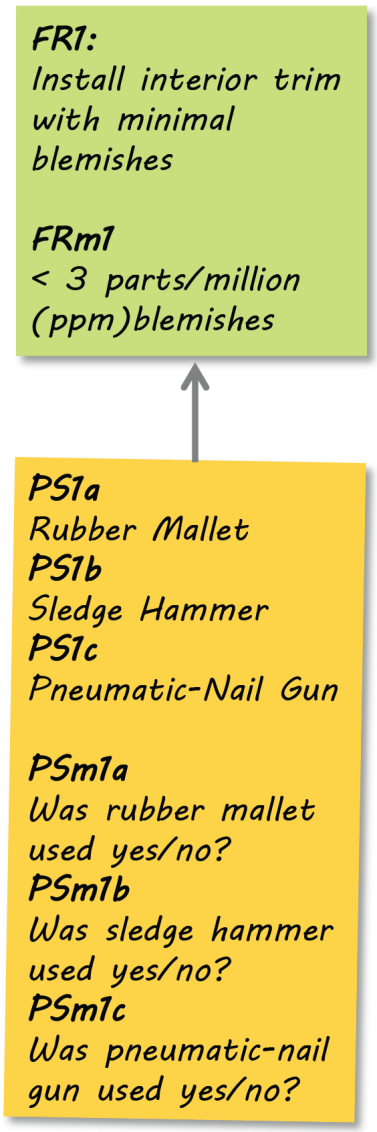


Figure 5: Design relationship with alternative PSs

System Design Decomposition

The **System Design Decomposition** (or map) is a method to visually represent a design(system) by linking the Physical Solutions (PSs) to their corresponding Functional Requirements (FRs). This method provides an easy way to show coupling or interactions of how a set of PSs affects the achievement of a set of FRs at a specific level of the design decomposition. To determine interactions, the designer must ask the question, **"does the choice of PSj affect the achievement of FRi within a branch at a specific level?"**

Steps to the Decomposition Process

1. Map Customer Needs to Functional Requirements
2. Choose Physical Solutions to satisfy Functional Requirements
3. Define a performance measure for the FRs (abbreviated FRm) when applicable.
4. Define a performance measure for the PSs (abbreviated PSm) when applicable.
5. Check for interactions/coupling among the PSs and FRs at the current level. Coupling is found by asking the question, **"does the choice of PSj affect the achievement of FRi within a branch at a specific level?"** (See "Possible Design Cases" below for a better understanding of coupling.) Arrows are drawn from physical solutions to functional requirements within a branch at a single level in order to indicate coupling. **NOTE:** Level I is considered a one FR design which means coupling is not possible.
6. Decompose the design to the next level. **NOTE:** A design must be checked for interactions/coupling before the design can be decomposed to the next level. The decomposition can only proceed if the design is uncoupled or partially coupled at that level. In addition, a design can only be decomposed to the next level if at least **two** or more FRs expand on the preceding higher level PS. Stop decomposing when the PS expresses the design well enough that it can be implemented.

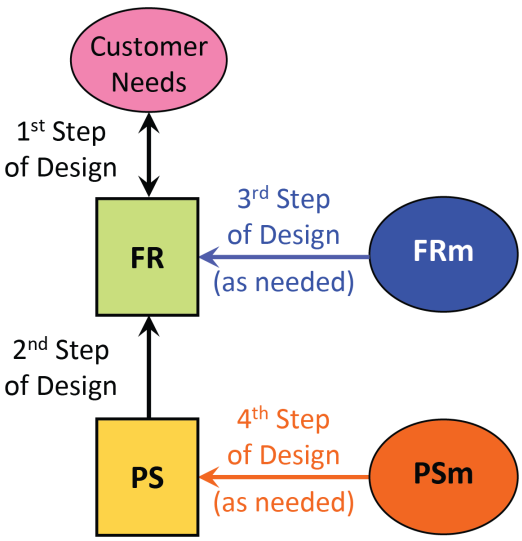


Figure 6: System Design Language

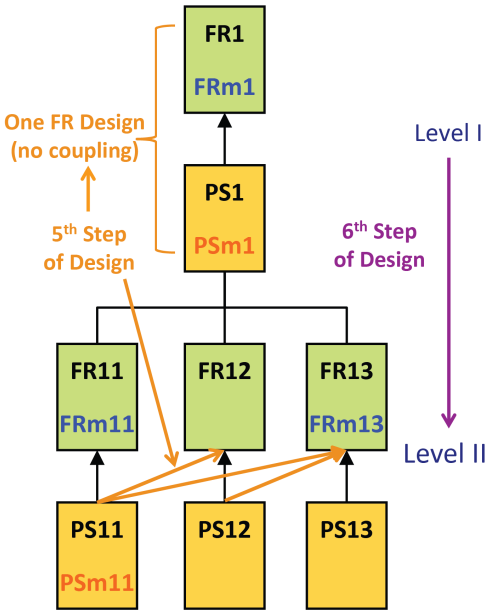


Figure 7: Design Decomposition Format

Decomposing Alternative Solutions

For each FR defined, there exists the possibility for alternative solutions. Based on the solution, the resulting branches might vary. The way to represent these solutions with their differing branches is shown in Fig. 8.

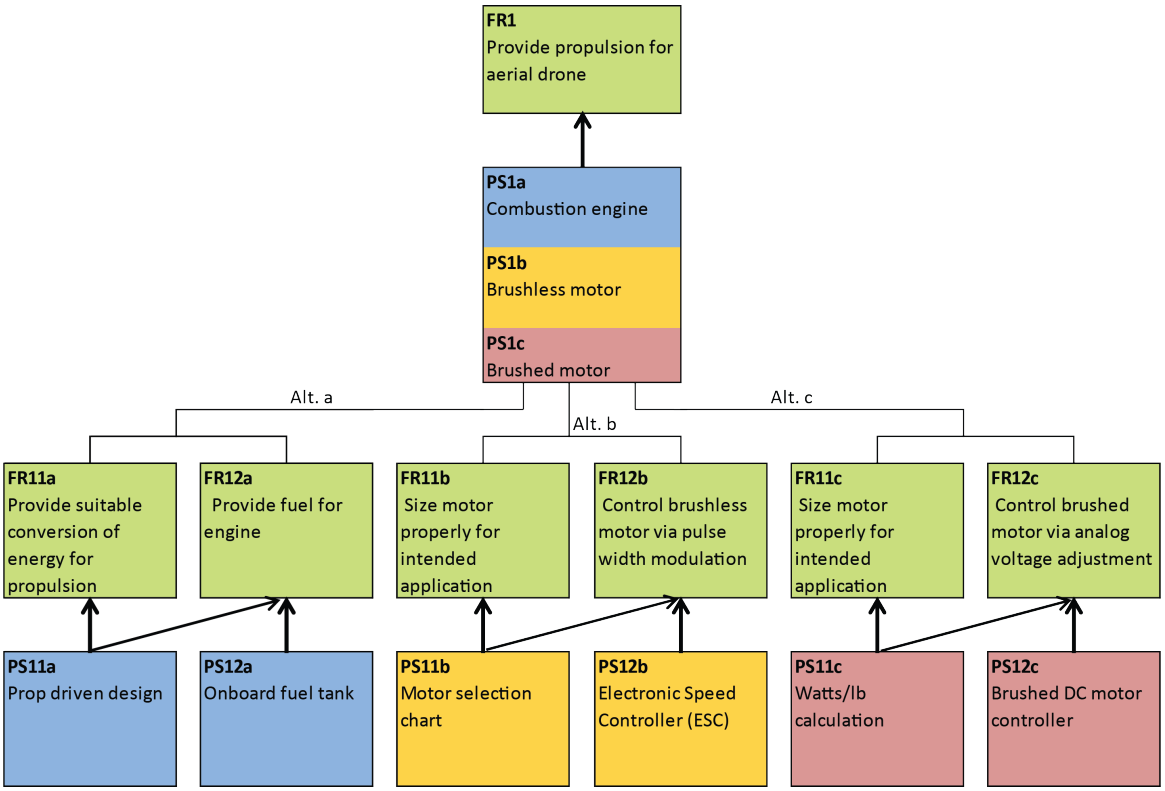


Figure 8: Decomposition showing break down of alternative PSs

Motivation

The motivation to utilize a System Design Language is detailed in Fig. 9. Using a System Design Language approach provides a means for understanding the implementation sequence of the physical solutions. This approach results in eliminating the iterations that commonly occur when a design is coupled. Overall, the time and expenses involved in the design phase are reduced.

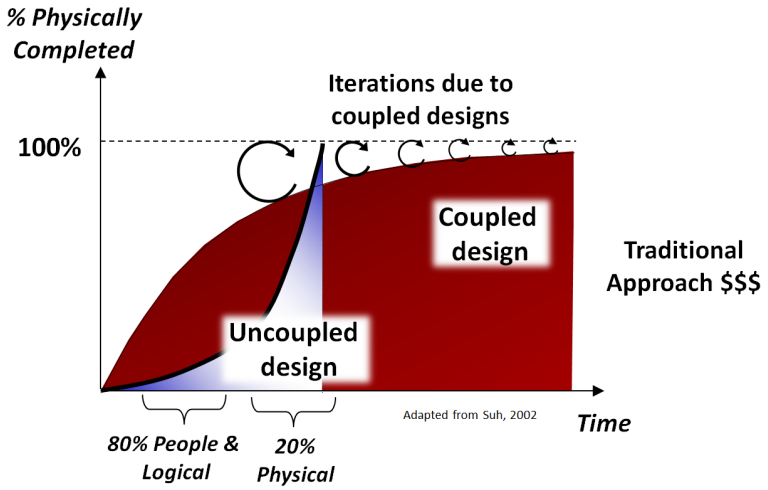


Figure 9: Comparison between the traditional approach and the System Design Approach

Possible Design Cases

The possible design cases are listed below. Remember that a design is acceptable when it is either uncoupled or path dependent. The four cases that are unacceptable are listed below. As stated before, interactions (X's in the below matrices) are determined by asking the question, **"does the choice of PSj affect the achievement of FRi within a branch at a specific level?"** Equation 1 represents the design equation for the relationship between FRs and PS within a branch at a specific level.

$$FRi = \left[Aij\right] PSj \tag{1}$$

Where Aij is the design's relationship matrix for a branch at a specific level.

Acceptable Cases

- Uncoupled

$$\begin{bmatrix} FR1 \\ FR2 \end{bmatrix} = \begin{bmatrix} X & O \\ O & X \end{bmatrix} \begin{bmatrix} PS1 \\ PS2 \end{bmatrix}$$

- Path Dependent (Partially Coupled)

$$\begin{bmatrix} FR1 \\ FR2 \end{bmatrix} = \begin{bmatrix} X & O \\ X & X \end{bmatrix} \begin{bmatrix} PS1 \\ PS2 \end{bmatrix}$$

Unacceptable Cases

$\Longrightarrow$ Related to **Axiom 1:** Maintain the independence of the functional requirements (FRs).

1. Coupled

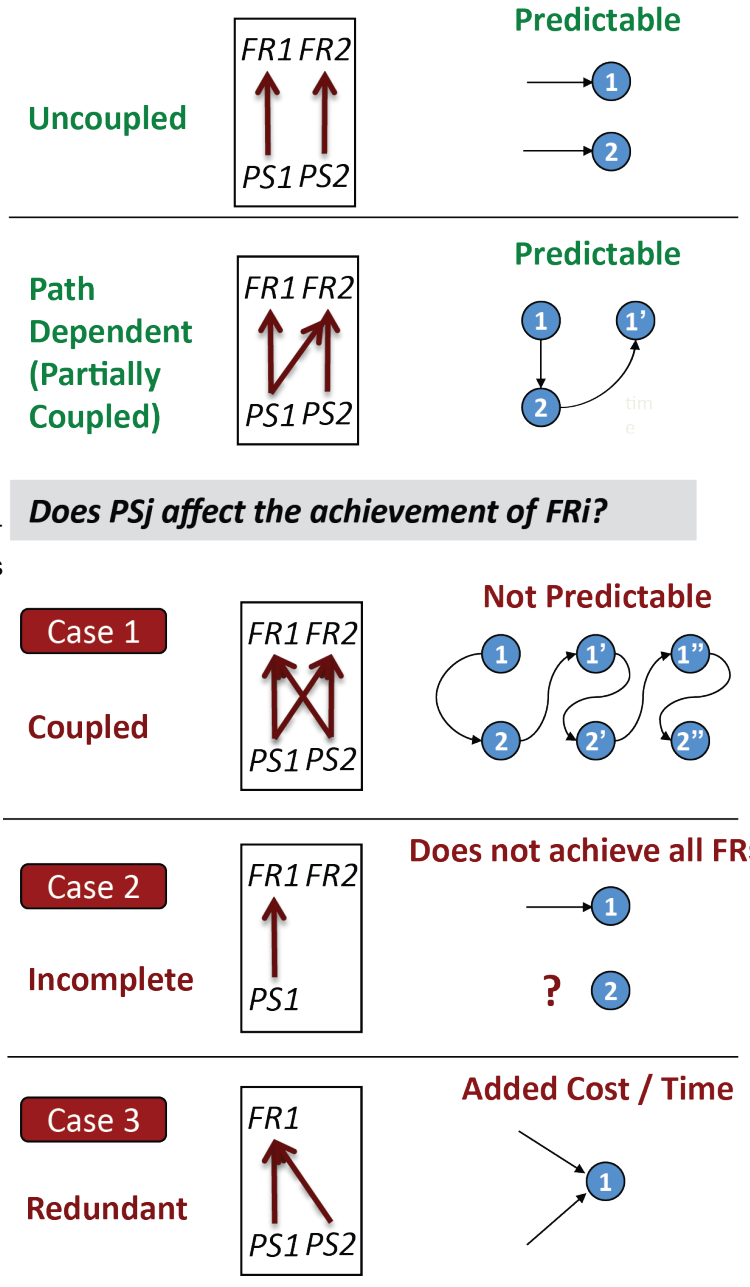
$$\begin{bmatrix} FR1 \\ FR2 \end{bmatrix} = \begin{bmatrix} X & X \\ X & X \end{bmatrix} \begin{bmatrix} PS1 \\ PS2 \end{bmatrix}$$

2. Incomplete

$$\begin{bmatrix} FR1 \\ FR2 \end{bmatrix} = \begin{bmatrix} X & O \\ O & O \end{bmatrix} \begin{bmatrix} PS1 \\ PS2 \end{bmatrix}$$

3. Redundant

$$\begin{bmatrix} FR1 \end{bmatrix} = \begin{bmatrix} X & O \\ X & O \end{bmatrix} \begin{bmatrix} PS1 \\ PS2 \end{bmatrix}$$



$\Longrightarrow$ Related to **Axiom 2:** Minimize the information content of the design.

4. Not Process Capable A design that is not process capable is one in which a chosen PS does not achieve the corresponding FR. In other words, the Physical Solution did not achieve the Functional Requirement Therefore, so it is said to be not process capable.

Implementation Sequence of Physical Solutions

A graphical representation of the implementation sequence of the PSs for three possible design cases is presented in the figure on the right.

In Fig.10, points A and B on the graphs represent the desired level of achievement of FR1 and FR2. In addition, point B has a combined higher level of FR1 and FR2 achievement, than A. In the case of a predictable (uncoupled) design, implementing PS1 affects only FR1, and implementing PS2 affects only FR2. Therefore, this design is the most robust in reference to a change in PS1 or PS2, since PS1 only affects the achievement of FR1 and likewise, PS2 only affects FR2. This independence leads to the most robust design and thus defines the least waste condition.

A path dependent design (Fig.11) is not ideal, but it is considered to be an acceptable design. As the name implies, a path must be followed during the PS implementation process. PS1 is implemented first, as it effects both FR1 and FR2. Once PS1 is implemented, PS2 can be implemented without fear of affecting FR1. Thus, when the implementation of PSs follows the correct sequence, an acceptable design will be the consequence.

With the coupled design (Fig.12), the result is considered to be an unacceptable design. The implementation sequence of the PSs represents a loop that may never allow point B to be reached. In this design case, PS1 affects both FR1 and FR2, and PS2 affects both FR1 and FR2 causing the implementation of one PS to alter the effectiveness of the other PS to meet its requirement. Long implementation time and the risk of never reaching target B with absolute certainty is the consequence of a coupled design.

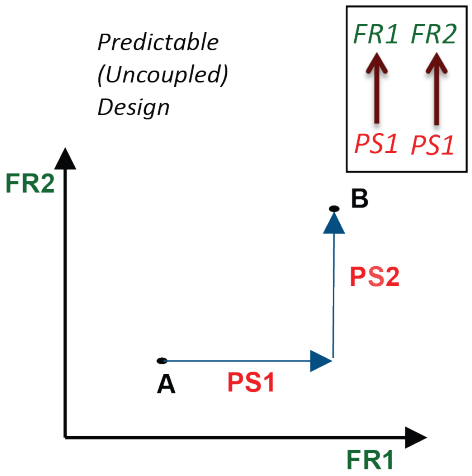


Figure 10: Predictable Design

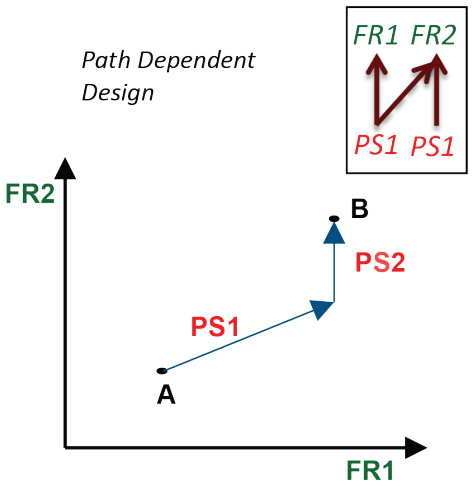


Figure 11: Path Dependent Design

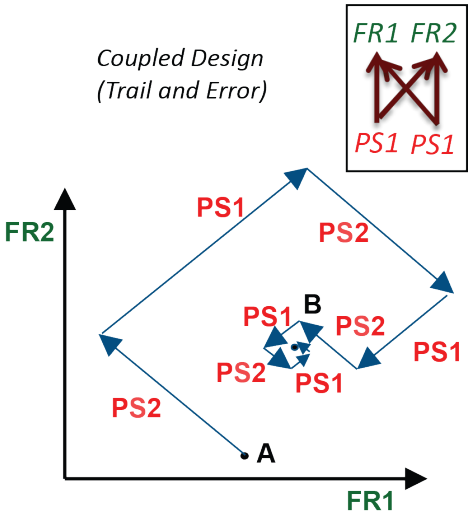


Figure 12: Coupled Design

Process Capability - A way to choose among alternatives.

Determining **Process Capability** is a way to determine whether alternative PS1a, PS1b... PS1n is the best solution to FR1. The drawback is that it requires time to build a detailed model (i.e., Computational Fluid Dynamics (CFD), Finite Element Analysis (FEA), solid model, etc.)[2] Process Capability can be determined for each individual FR, or for the whole design. The process capability of an PS is used to determine the information content associated with that particular PS.

Information Content (I_i) for a given FRi is defined in terms of the probability of success (P_i) of satisfying FRi with PSi.[2] P_i is calculated by finding the probability that the resulting outcome of a physical solution falls within the Common Range (CR). The CR represents the overlapping area between the Design Range and the System Range (SR).

$$I_i = \log_2 \frac{1}{P_i} \quad \text{where} \quad P_i = P\left(\frac{\text{CR}}{\text{SR}}\right) \quad \text{thus} \quad I_i = \log_2 \frac{\text{SR}}{\text{CR}}$$

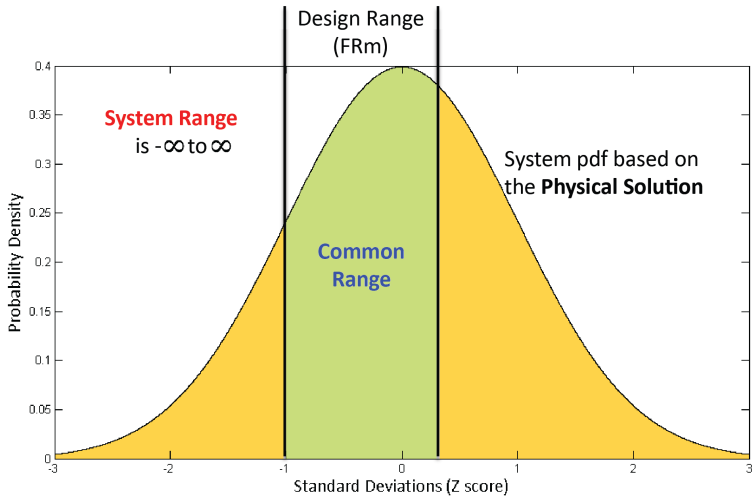


Figure 13: Sample Probability Density Function indicating the Common Range

Recall: $log_2(1) = 0$, therefore, when the System Range is completely within the Design Range, the Information Content is ZERO. This case is commonly referred to as a "robust" solution, meaning that the PS always meets the FR. Please refer to Fig. 14 for an example of a robust design.

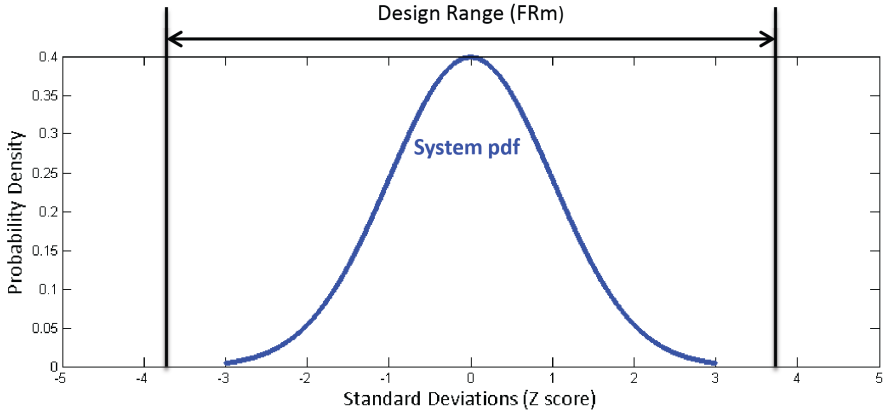


Figure 14: Robust design example (i.e., system range is within design range)

[2]Suh, Nam P. Complexity: Theory and Applications. Oxford: Oxford UP, 2005. Print.

Example of Using Axiom 2

Please refer to Fig. 8 for an example of alternative physicals solutions for one functional requirement (FR1 and PS1a,b,c). In addition, refer to Fig.13 for a graphical representation of the System Range and Design Range for the alternative PSs. In the below calculations, data from Fig.15 is used to calculate the probability and Information Content of the three proposed Physical Solutions. Eq.2 represents the probability density function. Eq.3 represents the probability of an outcome in a specific range given a probability density function (cumulative distribution function).

$$f(x,\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2} \tag{2}$$

Where

$e =$ Eulers's number, $\quad \sigma =$ standard deviation, $\quad \mu =$ mean

$$\phi(z) = \frac{1}{\sigma\sqrt{2\pi}} \int_0^z e^{\frac{-1}{2}(\frac{x-\mu}{\sigma})^2} dx \tag{3}$$

PS1a:

$$f_{1a}(x,-2,1) = \frac{1}{\sqrt{2\pi}} e^{\frac{-1}{2}(x+2)^2}$$

$$\phi_{1a} = \frac{1}{\sqrt{2\pi}} \int_0^2 e^{\frac{-1}{2}(x+2)^2} dx$$

$$\phi_{1a} = 0.0227 P_{1a} = 2.27\%$$

$$I_{1a} = log_2 \frac{1}{0.0227}$$

$$\boldsymbol{I_{1a} = 5.461}$$

PS1b:

$$f_{1b}(x,0,1) = \frac{1}{\sqrt{2\pi}} e^{\frac{-x^2}{2}}$$

$$\phi_{1b} = \frac{1}{\sqrt{2\pi}} \int_0^2 e^{\frac{-x^2}{2}} dx$$

$$\phi_{1b} = 0.4773$$

$$P_{1b} = 47.73\%$$

$$I_{1b} = log_2 \frac{1}{0.4773}$$

$$\boldsymbol{I_{1b} = 1.067}$$

PS1c:

$$f_{1c}(x,2,2) = \frac{1}{2\sqrt{2\pi}} e^{\frac{-1}{2}(\frac{x-2}{2})^2}$$

$$\phi_{1c} = \frac{1}{2\sqrt{2\pi}} \int_0^2 e^{\frac{-1}{2}(\frac{x-2}{2})^2} dx$$

$$\phi_{1c} = 0.3413$$

$$P_{1c} = 34.13\%$$

$$I_{1c} = log_2 \frac{1}{0.3413}$$

$$\boldsymbol{I_{1c} = 1.551}$$

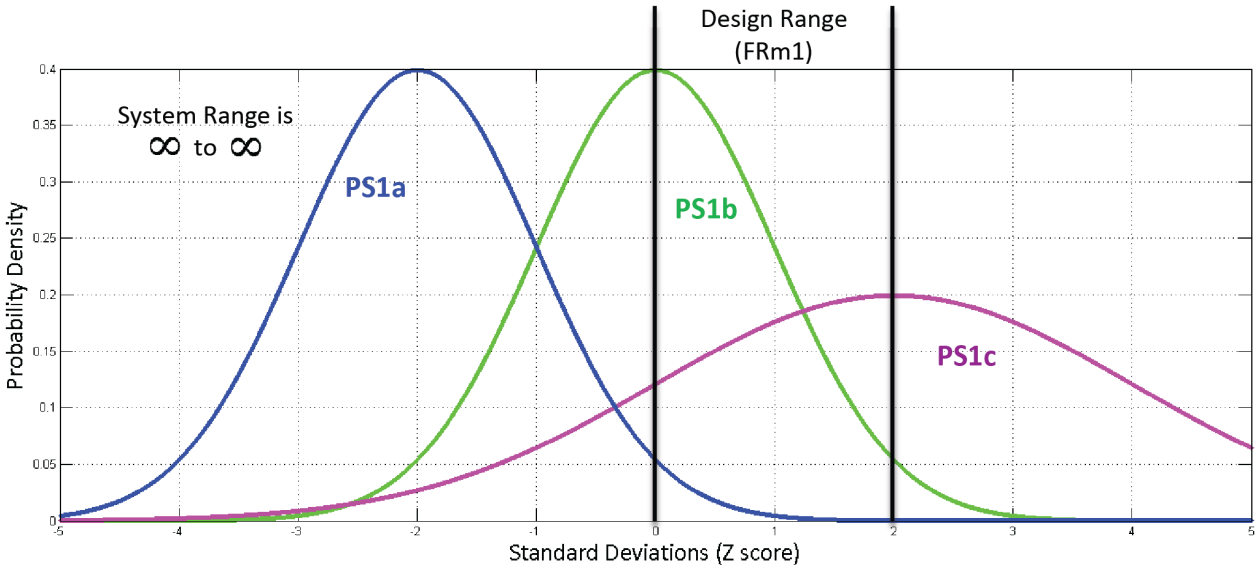


Figure 15: Example of System Range and Design Range for Detailed FR $\Longleftarrow$ PS in Fig. 4.

Conclusion: From the above calculations, **PS1b** has the least amount of information content. According to Design Axiom 2, **PS1b** would be the chosen physical solution to meet FR1. **PS1b** is considered the most capable process (solution) to meet the specified functional requirement.

Important System Design Errors to Avoid

1. **Incorrect wording of Functional Requirements:** A Functional Requirement begins with a Verb. Think about an "action" or "transformation" that must occur.
2. **Incorrect wording of Physical Solutions:** A Physical Solution begins with a Noun. A PS is something observable or quantifiable.
3. **Unacceptable cases:** Check for the four unacceptable cases (coupled, incomplete, redundant and not process capable) prior to decomposing to the next level.
4. **Inadequate branching:** A design can only be decomposed to the next level if at least **two** or more FRs expand on the preceding higher level PS.
5. **Incorrect coupling expressed between levels:** Arrows indicating coupling should not be seen branching from one level to another, or from one branch to another.
6. **Incorrect coupling expressed between branches:** Coupling is only expressed visually between PSs and FRs within the same branch. Coupling elsewhere is already indicated by showing coupling at a higher levels.
7. **Incorrect branch Formating** Arrows are drawn from the PS to the FR. Lines indicating the branches should not be arrows.
8. **Incorrect numbering of FRs and PSs:** As a physical solution is decomposed into further functional requirements, the numbering should follow the specified format. For example, PS1 is decomposed into FR11, FR12...FR1n with their corresponding PSs of PS11, PS12...PS1n.

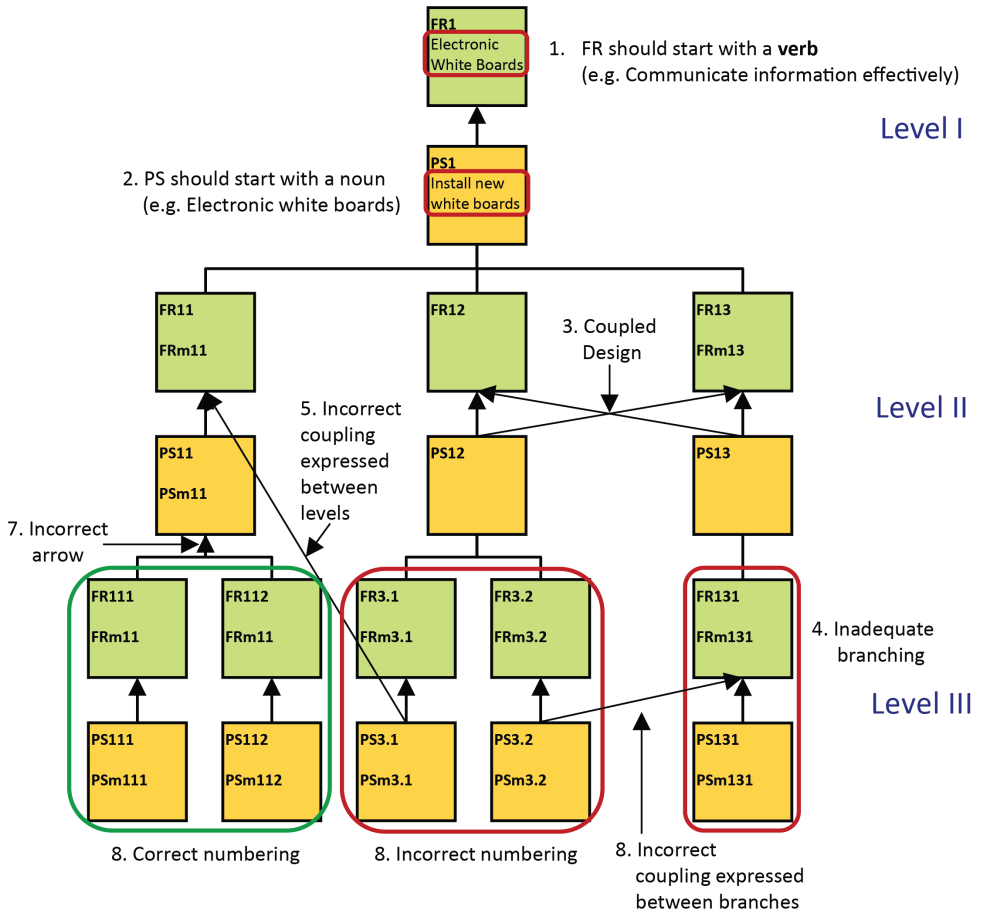


Figure 16: Common System Design Errors (See list above for details)

Application

The System Design Language expressed in this primer represents a tool that guides the design of a system. The design decomposition provides the user with a map that indicates the order in which to implement solutions and also an order in which to improve the system. In Fig.17, a simple design decomposition of a manufacturing system is presented. The arrows that indicate coupling provide the flow of the implementation and improvement process.

From this decomposition, the user can conclude things such as:

- "The quality may be perfect, but if the product design doesn't meet the customers' needs, the system failed."
- "Implementing single piece flow (to improve delay reduction) should only occur after the product design is correct, the quality standard is met, problems are identified and resolved, and the output is predictable."

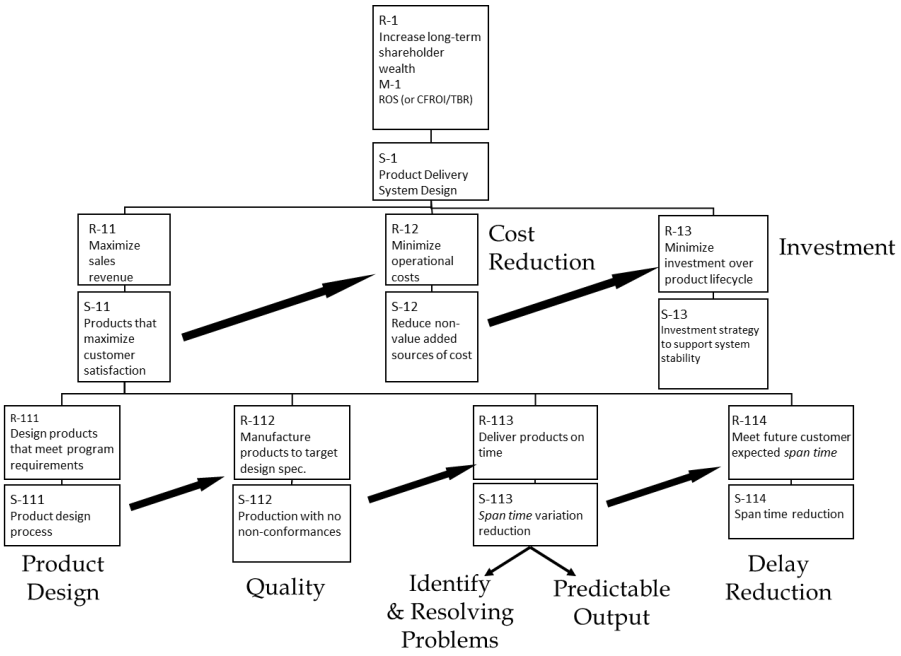


Figure 17: System Design Decomposition indicating path dependency of a manufacturing system design.