A Case Study on an Attribute-Based Design Method Selection Framework

By

Ephraim Chen

Master of Science, Aerospace Engineering
Georgia Institute of Technology, 2013

Bachelor of Science in Engineering, Mechanical and Aerospace Engineering
Princeton University, 2009

Submitted to the System Design and Management Program
in Partial Fulfillment of the Requirements for the Degree of

Master of Science in Engineering and Management
at the
MASSACHUSETTS INSTITUTE OF TECHNOLOGY

February 2018

Signature redacted

Signature of Author: .. ..
Ephraim Chen
System Design and Management Program
January 3, 2018

Signature redacted

Certified By: . ...
Warren Seering, PhD
Weber-Shaughness Professor of Mechanical Engineering and Engineering Systems
Co-Director, System Design and Management Program
Thesis Supervisor

Signature redacted

Accepted By: ....
Joan Rubin
Executive Director and Senior Lecturer, System Design and Management Program

(THIS PAGE INTENTIONALLY LEFT BLANK)

A Case Study on an Attribute-Based Design Method Selection Framework

By

Ephraim Chen

Submitted to the System Design and Management Program on January 3, 2018
in Partial Fulfillment of the Requirements for the Degree of
Master of Science in Engineering and Management

Abstract

This work demonstrates the effectiveness of using the concept of attributes or properties to identify, compare, and select methods for the design of aerospace systems. Growing product development cost trends for these complex systems have been alarming to the aerospace industry. To curb rising development costs by improving the product design actions that drive them, the issue is viewed from a "design system" perspective that distinguishes between the product design process – the set of tasks or problems to solve to produce a design – and the set of problem-solving techniques or methods used to complete the tasks. Support of a structured approach for designers to select their methods would help ensure that methods meeting the specific quality, budget and schedule needs of each unique design situation are utilized in a manner that is transparent to the whole design team.

This thesis develops a conceptual framework for method selection decision support that combines a multi-form design process-methodology model with a general collection of attributes for characterizing any method. The model and attribute framework enable the discovery and comparison of alternative design methods relative to a given design task's requirements. The framework was validated by a case study on the early system-level conceptual design phase of a recent industry flight vehicle development program. By employing graphical and matrix modeling techniques, primary research and interviews with members of the industry design team empirically substantiated the overall efficacy of the framework, indicated four particular attributes that are especially important for comparing methods, and revealed six contextual factors that influence the attribute-based characterization of a method.

Thesis Supervisor: Warren Seering, PhD
Title: Weber-Shaughness Professor of Mechanical Engineering and Engineering Systems

(THIS PAGE INTENTIONALLY LEFT BLANK)

Acknowledgements

To my advisor and comrade-in-arms in the quest for continuous improvement and holistic thinking, Professor Warren Seering, I thank you. Your thoughtful and sincere guidance, wealth of experience, and patient encouragement throughout the research process in all of our memorable conversations made all the difference in the world.

To my colleagues at The Boeing Company for your time and support as I conducted this case study on a real program, I thank you: Roger Lacy, Dan Newman, Eric Hathaway, Perry Ziegenbein, Dave Mason, Mark Spano, Joe Nickerson, Jamie Dryfoos.

To the members of the SDM program and the MIT community who have made this a truly cutting-edge and accommodating educational experience for me, I thank you.

To my family, for your steadfast love, I thank you.

I dedicate this work to the Creator God, the Giver of Life and the Greatest Designer.

(THIS PAGE INTENTIONALLY LEFT BLANK)

Table of Contents

List of Figures

List of Tables

Nomenclature

AIDA	Analysis of Interconnected Decision Areas
DBD	Document-Based Design
DMM	Domain Mapping Matrix
DSM	Design Structure Matrix
IPT	Integrated Product Team
MAU	Multi-Attribute Utility
MBD	Model-Based Design
MDM	Multi-Domain Matrix
MMM	Munich Model of Methods
OPM	Object-Process Methodology
PBD	Point-Based Design
PDP	Product Development Process
PoMM	Process-oriented Method Model
QFD	Quality Function Deployment
RCDSA	Rapid Custom Domain-Specific Analysis
SBD	Set-Based Design
SME	Subject Matter Expert
TRIZ	*Teoriya Resheniya Izobretatelskikh Zadach* (Theory of Inventive Problem-Solving)

1 Improving Aerospace Product Development by Supporting Design Method Selection

This chapter provides an executive summary of the contents of this thesis, including the rationale and background behind the work – roadmapped at a high level in Figure 1 – and a simplified illustration of the concept underpinning the thesis statement. The organization of the rest of the thesis is also described.

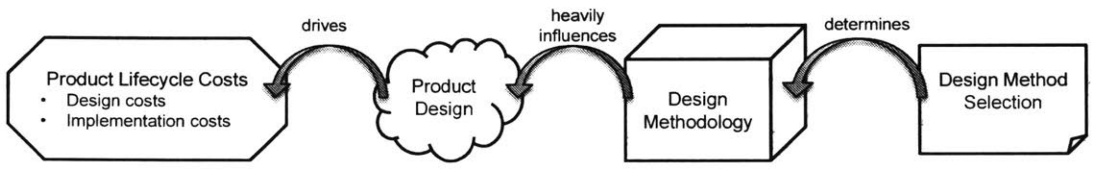


Figure 1: Thesis motivation "chain"

1.1 Impact of Design on Aerospace Development Costs

U.S. aerospace industry development costs have been steadily rising over the past several decades, as explored in more detail in section 2.1. Such trends have serious ramifications for the armed forces' readiness and ability to procure new aircraft, rotorcraft, spacecraft and other defense systems, for commercial airlines to predictably perform aircraft fleet planning to serve new routes and markets, and for the economic outlook of aerospace manufacturers.

While several causal factors for development cost growth – many of which are dynamically coupled and interdependent – have been identified by multiple sources, the inadequacy of legacy design practices to cope with the growing complexity of modern aerospace systems is a prominent recurring theme. As portrayed in Figure 2, most of a system's total lifecycle costs are determined during the design phase; after that, it becomes increasingly expensive to address design flaws. Therefore, the design organization's capabilities have profound leverage on controlling development costs and handling mounting system complexity.

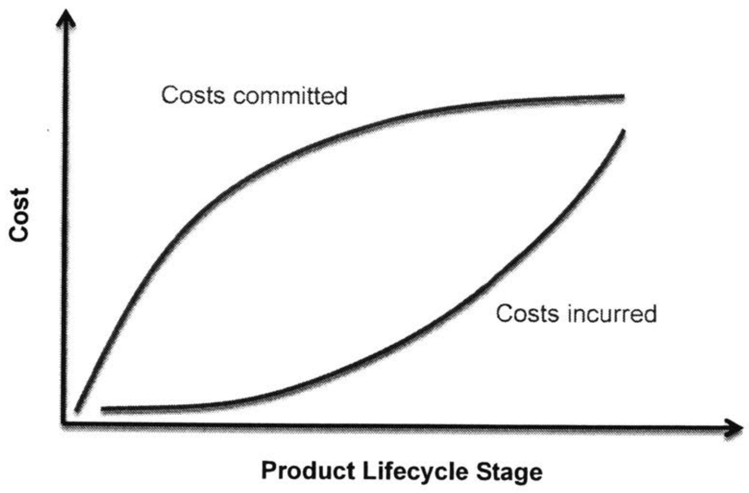


Figure 2: Approximate trends of product lifecycle costs

1.2 Defining Methodology within the Design System

A holistic approach to evaluating legacy design practices for improvement begins with the realization that an entire system is at work during the design of a new aerospace product. As section 2.2 explains, this system – the design system – consists of five domains: the design goals, the product design, the design process, the design organization, and the design resources, as depicted in Figure 3. The design process is the set of tasks undertaken by the design organization to produce the product design. To accomplish this, the design organization utilizes design resources, a domain that includes three elements: design methods, design tools, and design-enabling platforms.

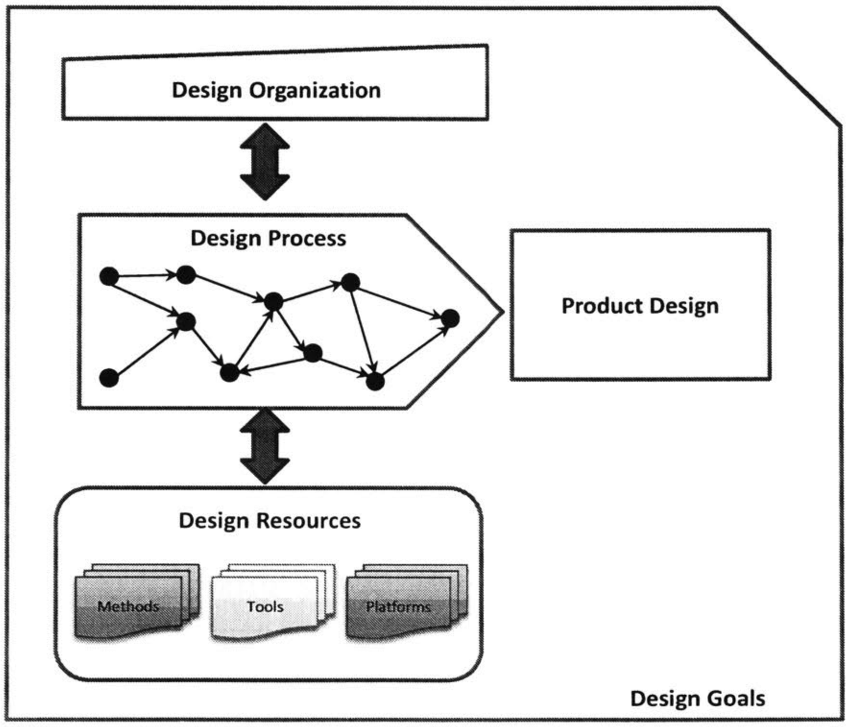


Figure 3: Design system domains

A design method is the fundamental problem-solving technique used to tackle a design task, while a design tool is a particular implementation of the method that the designer interacts with. If an enterprise is developing a novel aircraft configuration to meet new customer needs, a design tool will be handicapped by an underlying design method that is ill-equipped to deal with the new physics or complexity. A deficient design tool will in turn inhibit the designer who wields it from fulfilling the design task with a deliverable of satisfactory quality or at acceptable cost. A flawed task deliverable can impact downstream tasks and propagate design errors that are expensive to correct if not discovered early enough.

A design methodology is a set of methods collectively used to execute a design process and form the overall product design. A methodology dominated by inadequate legacy design methods causes the design process to deliver a deficient product design and results in poor quality, cost and schedule outcomes. This assertion is born from a systemic view of design and its various facets.

1.3 Choosing the Right Design Methods

Enormous potential for improving design quality and development project performance is found in a structured approach to design methodology and, in particular, a framework for designers to select which design methods to deploy to solve tasks. The designer always chooses their methods, consciously or unconsciously. Even if the design methodology is openly formulated and not relegated to implicit decisions, many design organizations succumb to the inertia of legacy methods. It is easy to default to the existing methodology without basing that choice on the actual requirements of the design situation in a transparent decision-making process involving all relevant parties of the design team.

What is needed is a framework to support the designer's consideration of a broad range of applicable design methods and selection of methods that meet each unique design context. Method selection decision support that helps the designer cope with and efficiently search and assess the plethora of methods in existence can be accomplished through an attribute-based approach. By rigorously characterizing methods with relevant attributes while accounting for contextual factors, methods can be systematically evaluated for compatibility with design tasks, and the attribute differences of compatible methods can be compared against the needs of the situation to make a selection.

1.4 An Illustrative Example

To demonstrate the concept of attribute-based method selection for the development of aerospace products, consider the following flight vehicle control law design task: "Tune the control law design parameter for a particular flight control system feedback loop." The premise of the task is that some control law architecture has already been defined and now a junior, less-experienced control law engineer has been charged with tuning a specific parameter of that architecture, e.g. a feedback gain, to achieve desired closed-loop vehicle performance. The method selection framework assists the engineer in discovering control law tuning method alternatives and comparing these alternatives prior to a final choice of method to use.

1.4.1 Determining Available Method Options

As described in general in subsection 2.2.3 and section 4.4, the flight control law parameter tuning task can be decomposed into a set of input information (or "design artifacts") transformed into output information:

- Input Artifacts: closed-loop performance criteria, and the control law loop transfer function
- Transformation (or "design activity"): control law parameter tuning, which involves "basic activities" such as simulating, analyzing and adjusting
- Output Artifact: tuned design parameter value

Let the engineer's baseline method be the so-called root-locus method. Using compatibility-relevant method attributes, it is evident that the root-locus method is certainly a fit for the task at hand since it converts the input artifacts of controller performance criteria and loop transfer function information into a tuned design parameter value output artifact by performing the "control law parameter tuning" design activity.

One alternative method that is also compatible with this design task is the frequency-response method. The frequency-response technique also uses design criteria and the control law loop transfer function to come up with a tuned feedback gain via the "basic activities" of simulating, analyzing and adjusting. Of course, the principles, procedures and mathematical tools employed by the root-locus and frequency-response techniques to analyze dynamical systems are different, but all that is required for both of these methods to be applicable to the same design task is for the methods and task to share the same input artifacts, output artifacts, and basic activities.

1.4.2 Comparing Methods by Their Attributes

As discussed in detail in sections 4.5 and 4.6, several comparison-relevant method attributes can be utilized to determine the pertinent differences between the root-locus and frequency-response methods:

- Method Inputs: The root-locus method typically takes closed-loop performance criteria in the form of desired pole-zero locations, whereas frequency-response design criteria usually involve bandwidth and stability margin (gain margin, phase margin) specifications. If one form is easier to obtain than the other depending on, say, the particular vehicle or product context involved, that may influence the final choice of tuning method.
- Method Outputs: no difference (tuned design parameter value)
- Quality of Method Outputs: The frequency-response method may yield more accurate tuning answers than root-locus, but with lower precision and in a less accessible format (time domain vs. frequency domain) due to control law tuning tool availability and the engineer's experience level and skillset, even for the same time and effort spent on each method and after accounting for some initial frequency-response method training for the user (though more method training may change this attribution). However, the precision and accessibility concerns may not be important if there is an overriding need for

accuracy due to dependencies of downstream design tasks on this feedback gain tuning task.

- Method Implementation Cost: Switching to the frequency-response method will entail some upfront tool setup and user training costs (e.g. time, labor, software procurement) – not to mention intangible experience building – that staying with root-locus does not involve. However, it may take longer to execute the root-locus technique each time to come up with a tuned feedback gain relative to the frequency-response method, if both methods were held to a given quality level (e.g. accuracy). And root-locus could involve higher "maintenance" costs over the long run to let it work with multi-parameter control loops and to compensate for its lower average accuracy. Nonetheless, the nonrecurring startup costs associated with choosing the frequency-response method may be an overriding concern if there is a strict schedule constraint on this tuning task.

This example illustrates how the use of key attributes to compare candidate methods in multiple dimensions can be readily correlated against governing constraints on the overall design task to efficiently yield a fairly complete understanding of the important tradeoffs involved in picking one method over the other. For this flight control law gain tuning task, the junior engineer may elect to switch from root-locus to the more accurate and lower recurring-cost frequency-response method because performance criteria input information in bandwidth/stability margin format is readily available and the design team, based on the transparent method comparison provided by this framework, decides to budget in sufficient method familiarization time and infrastructure setup to facilitate the transition. Even this relatively simple case demonstrates the complex and multifaceted issues present in these method selection decisions, which up to now have often occurred tacitly without team involvement or holistic considerations.

1.5 Thesis Organization

The structure and relationships between the remaining sections of this thesis are depicted in Figure 4. Chapter 2 Literature Survey and Theoretical Framework provides the background and context for the thesis motivation (sections 3.1 and 3.2), and both serve as the foundation for the rest of the work. The research question (3.3) immediately follows from the motivation, and research objectives (3.4) are crafted to answer the research question. A research approach (3.5) is developed to gather data for and address the research objectives. From the thesis formulation, section 4.1 Conceptual Framework for Method Selection proposes a construct for directing the case study. Section 4.2 Program Overview sets the backdrop for the rest of the case study. The bulk of the development of the case study (sections 4.3, 4.4, 4.5 and 4.6) is in close alignment with the method selection framework, and the case study results are presented in Chapter 5. The contributions of the work (6.1) in achieving the research objectives are summarized, and based on the case study results the research findings (6.2) are reported to answer the research question. Finally, recommendations going forward (6.3) are offered.

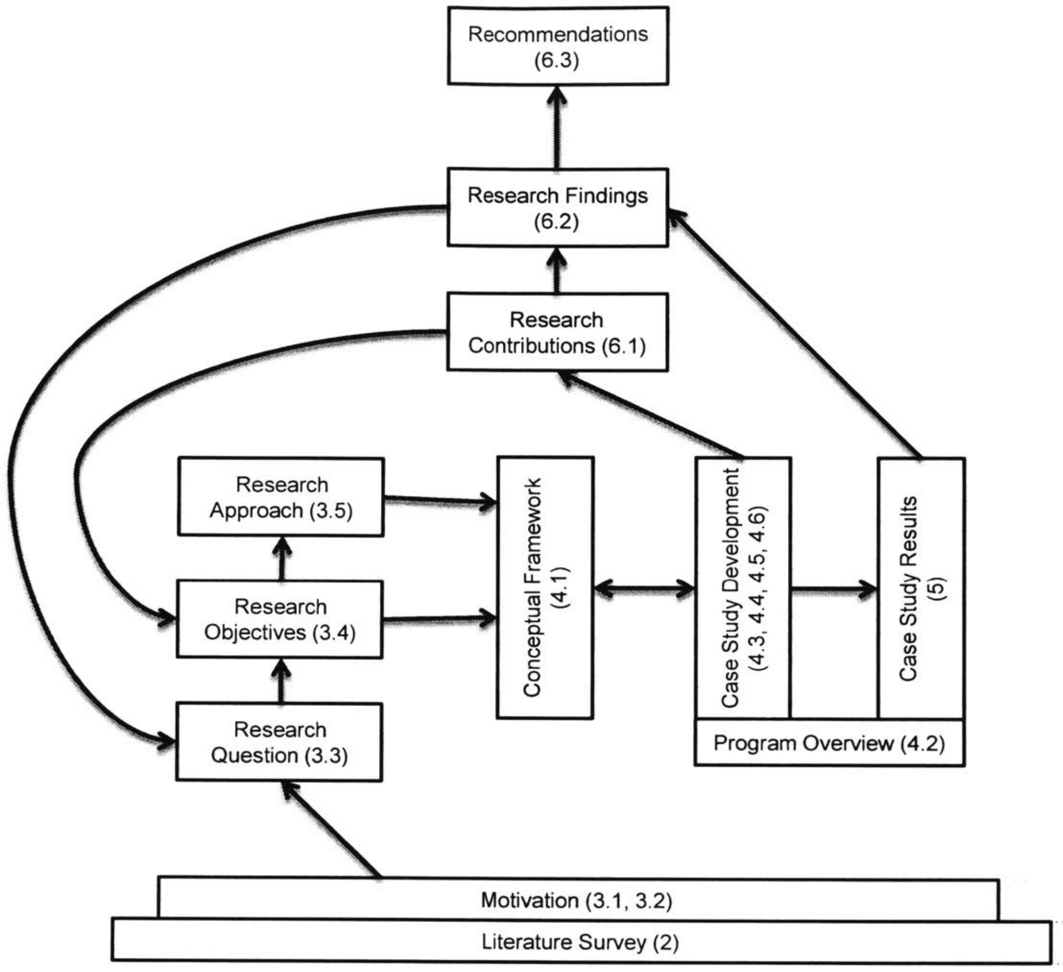


Figure 4: Thesis structure

2 Literature Survey and Theoretical Framework

A more thorough treatment of the background on this work's field of study is contained in this chapter: the issue of aerospace product development cost growth and the area of design science research. Summaries of the literature and definitions are given to provide the context and vocabulary for the thesis motivation, formulation, analysis and discussion that follow.

2.1 Aerospace Affordability

2.1.1 Scope of Research Problem

This thesis is concerned with the broad problem area of rising development costs in the U.S. aerospace industry. The situations in question have several common traits:

- Development projects: Systems or technologies are being designed and produced predominantly for the first time. A detailed treatment of the nature of product development is covered in section 2.2 below.
- Complex systems: As aerospace products are developed to meet increasingly demanding customer requirements, the complexity and scale of the products have grown accordingly.
- Large organizations: Companies and enterprises coordinate multiple groups of people, complicated processes, and vast pools of resources to execute these projects.
- High and mounting costs: The rest of this section expounds on this trait of recent aerospace development project.

It is worth mentioning that even the wide-ranging problem of aerospace development costs has an even broader context: cost is but one of many possible measures of project performance. The classic triple constraint on quality in project management considers scope and schedule in addition to cost. Ulrich and Eppinger (2012, pp. 2-3) lay out five profitability-related measures of a product development project's performance: product quality, product cost, development time, development cost, and development capability. Note the attention paid in this list to production (not just the development period) and to the organization's resulting product development competence (not just that particular project's product). All of these dimensions affect each other; this thesis chooses to highlight cost overruns simply to exemplify the challenges currently facing aerospace system development.

2.1.2 Aerospace Development Cost Trends and Implications

New product development costs – in contrast with unit procurement or flyaway prices – are generally those associated with activities starting from program initiation and ending with delivery of the first operational unit (Eskew, 2000, p. 213). As a general rule of thumb, Ulrich and Eppinger (2012, p. 5) state that product development costs are roughly proportional to the number of people involved and the duration of the project; thus, cost overruns and schedule delays go hand in hand. They use the Boeing 777 as an example of a large-scale development effort on a spectrum of other engineered, discrete products.

The pervading consensus is that U.S. aerospace system development costs are skyrocketing. The overall picture can be painted by a series of sources:

- "The duration and cost of [aerospace and defense] system development efforts...has experienced rapid exponential growth over time" (DARPA, 2009, p. 5).
- "The cost of Defense procurement is on a seemingly ever escalating cycle in the West. In the United States programs are now measured in the hundreds of billions of dollars. Cost increases from initial estimates can rise more than 60%." (Furuhjelm et al., 2017, p. 1)
- "The time required to execute large, government-sponsored systems development programs has more than doubled over the past 30 years, and the cost growth has been at least as great" (Committee on Pre-Milestone A Systems Engineering, 2008, p. 14)
- "The set of 96 major weapon system development programs currently underway have overrun by a total of $296 billion, with an average development cost growth of 42%, and

an average delay of 22 months... This is not a new phenomenon, but it appears to be worsening. Current defense system developments, projected to completion, are overrunning 78% in cost and 63% in schedule." (Collopy and Hollingsworth, 2009, p. 1)

These rising development program costs are troubling in several ways. In the case of defense systems, resulting schedule delays impact the Warfighter and cost overruns ultimately hurt the taxpayer. Annual aircraft unit cost increases have been found to be greater than inflation indices, which – when combined with relatively fixed defense investment budgets – implies fewer aircraft procured over time and forces a drop in government procurement rates and inventories (Arena et al., 2008). Augustine (1986, p. 143) humorously predicted, "In the year 2054, the entire defense budget will purchase just one aircraft. This aircraft will have to be shared by the Air Force and Navy 3-1/2 days each per week except for leap year, when it will be made available to the Marines for the extra day."

For commercial enterprises, cost growth and schedule delays hurt profitability. There is significant incentive to "break the cost curve." As Krishnan (1993, p. 8) notes, "...higher performance companies can reap huge cost benefits by more creative and efficient utilization of resources, smaller work-in-process and reduced development time." A survey of aerospace and defense industry and government sectors reported that more than 40 percent believed the aerospace development project performance problem to be at least as serious as the 2007-2008 housing and banking crises (Deloitte, 2009, p. 1).

2.1.3 Causes of Development Cost Growth

2.1.3.1 General Discussion

When considering what the underlying causes are for the alarming trends in aerospace and defense development costs, DARPA (2009) puts the spotlight on inadequate design practices. While system complexity has increased rapidly by several orders of magnitude, the systems engineering approach and design process applied to develop these systems have remained essentially unchanged. When design organizations fail to cope with growing complexity, project performance inevitably suffers.

A more multifaceted exposition presents five causes of rising development costs: technical complexity, talent shortage, supply chain challenges (e.g. outsourcing, coordination), politics (e.g. perceptions), and program management challenges (e.g. leadership turnover) (Deloitte, 2009). Complexity and, by extension, the systems engineering competence to address it are again called out.

Arena et al. (2008) attribute one-third of military aircraft unit cost increases to "economy-driven" factors (e.g. insufficient competition) and two-thirds to "customer-driven" factors (e.g. aircraft complexity, procurement program instability). They single out complexity as well – mostly driven by the desire for greater system capabilities, new build aircraft, and engineering change orders – labeling it the biggest single contributor to cost escalation.

Ulrich and Eppinger (2012, p. 6) record several challenges that affect the success of product development: the existence of tradeoffs instead of clear optimums, dynamic and uncertain environments, far-reaching sensitivity to technical details, time pressure, and the need for economic viability. Inadequate design capacity to deal with technical complexity could allow many if not all of these challenges to contribute to rising development costs.

It is important to keep in mind that development costs (and durations) are an emergent property or outcome of the underlying project dynamics. In classic system dynamics applications in project management (Lyneis et al., 2001), normal or initial work (non-rework) accomplishment is distinguished from rework (e.g. errors) and scope growth that drive up total duration and costs. Staff changes (e.g. productivity), management actions (e.g. staffing levels, overtime) and domino effects between projects can drive vicious cycles of declining project performance that obscure the root cause – namely, the introduction of rework due to quality issues such as design errors. At the very least, the recognition of these coupled dynamics at work within the project should motivate the pursuit of "design process improvement" as one prong in a multifaceted approach to addressing the growth of development costs.

2.1.3.2 The Importance of a Good Design

Therefore, it is clear that product design is a key influence on the performance of the overall product development effort. Design determines what system is to be implemented and eventually fielded and operated, and as such it has significant leverage on the total cost to develop the system. The Committee on Pre-Milestone A Systems Engineering (2008, p. 21) states, "...about three-quarters of the total system life cycle costs are influenced by decisions made before the end of the concept refinement phase..." And according to Krishnan (1993, p. 8), "A major fraction of a product's manufacturing costs – anywhere between 60-80% – are determined at the product design stage and it has been suggested that the decisions made at this stage be of higher quality."

The literature affirms the importance and potential that lies with improving design capabilities in order to manage product complexity and development costs. One form of the argument asserts that manufacturing performance among firms is converging, thus raising pressures to gain competitive differentiation via product design aptitude (Krishnan, 1993, p. 8). Another highlights how globalization and new information and communication technologies are driving shorter product lifecycles and increased product complexity, which requires the adoption of flexible and optimized product development practices (Birkhofer et al., 2002, p. 1).

First-time quality of the system design contributes to less downstream rework and an overall faster and lower cost development cycle. If inadequate design is performed, the product implementation period will be more costly as test and evaluation reveals design flaws and introduces rework (re-design, re-build, re-test). Even worse, if design flaws are not revealed during testing, any deficiencies relevant to the customer will be exposed after deployment during operational use when it will be even more expensive to correct issues.

2.2 Design Research

2.2.1 Design Science and the Product Development Context

Research on design science studies structured approaches to design. The definition from Cross (1993, p. 21) says, "...design science refers to an explicitly organised, rational and wholly systematic approach to design; not just the utilisation of scientific knowledge of artefacts, but design in some sense as a scientific activity itself." Hubka and Eder (1996, p. 73) present a similar definition: "The term *Design Science* is to be understood as a system of logically related knowledge, which should contain and organize the complete knowledge about and for designing." They propose organizing design science around four types of knowledge based on whether the subject deals with the act of designing or the designed system itself, and whether it takes the form of descriptive statements (theory) or prescriptive statements (practical knowhow). The subject of this thesis is related to design research concerned with understanding and improving the act of designing.

Design is an essential component of the endeavor known as product development. Ulrich and Eppinger (2012, p. 2) define product development as "the set of activities beginning with the perception of a market opportunity and ending in the production, sale, and delivery of a product." Before further exploring the internal structure of product development, it is useful to first recall that product development occurs in the context of the overall lifecycle of the product. Pahl et al. (2007, p. 3) describe the product lifecycle with several stages:

- Needs/goals
- Product planning
- Design/development
- Production/assembly/test
- Marketing/sales
- Use/consumption/maintenance

- Recycling
- Disposal

The generic system lifecycle model from INCOSE (2011) has the following stages:

- Exploratory Research
- Concept
- Development
- Production
- Utilization
- Support
- Retirement

There are many other models of the generic product lifecycle, including the "generic product development process" by Crawley et al. (2016) (described shortly) that appears to be broad enough to encompass the entire product lifecycle. One way to consolidate these models for an aerospace product is the following set of lifecycle stages:

1. Birth: A product begins with an idea or opportunity, from market needs and/or the goals of the enterprise developing the product.
2. Business Development: During this stage, market and competition analysis occur, as well as business modeling and financial analysis activities that refine the product opportunity that should be pursued.
3. Design: This stage is traditionally broken up into conceptual design, preliminary design, and detailed design sub-stages. Note that this refers to a stage in the lifecycle, not the product Design function in the organization.
4. Prototyping: During this period, the development of the product is mainly characterized by fabrication, assembly/integration, and test and evaluation occurring on pre-production versions of the system.

5. Production: Full-scale production of the product can commence once the product design and its implementation are sufficiently mature.
6. Deployment: Sale and delivery of the product to the end user typically initiates after the Production stage (#5) has begun but then continues for as long as there are produced products to deploy.
7. Operation: The product is eventually in the hands of the end user and operating for its intended purpose, while the manufacturer provides support and maintenance services, perhaps long after the Production (#5) and Deployment (#6) stages end.
8. Evolution: Improvements to the original production version of the system can be based on, for example, user feedback after some period of operation and rolled into subsequent production releases.
9. Retirement: The product is eventually removed from general use and disposed of.

Figure 5 portrays this lifecycle model using a representation consistent with that of a complex aerospace system – e.g. most of the product's design is accomplished prior to large-scale distribution and operation of the product.

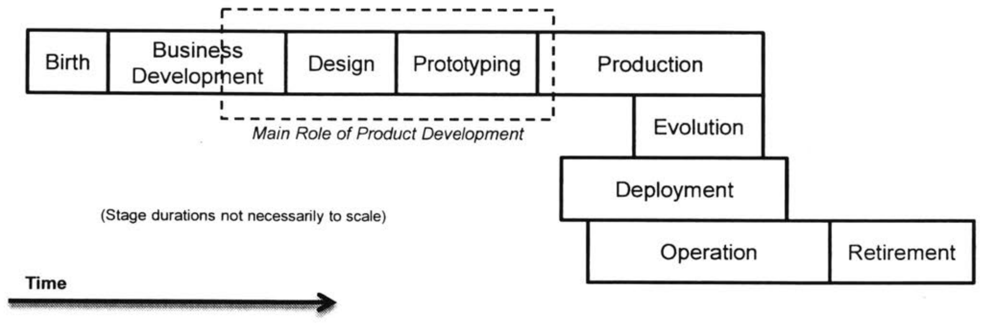


Figure 5: Notional product lifecycle as context for product development

Product development's place within the overall product lifecycle is also depicted in Figure 5. Note that its primary scope includes not only the Design and Prototyping stages of the product, but also relevant aspects of the Business Development that births the product and the transition to full-rate production.

Product development is traditionally represented as a process – the product development process (PDP), which captures "the terminology, phases, milestones, schedules, and lists of tasks and outputs" of product development (Crawley et al., 2016, p. 184). Ulrich and Eppinger (2012, p. 13) suggest three interpretations of the PDP:

- The PDP initially creates a wide set of alternative product concepts and then narrows the set of alternatives down and provides an increasingly detailed specification of the product until it is ready to be produced.
- The PDP is an information processing system that transforms inputs such as strategic opportunities, corporate objectives, technologies and production capabilities into outputs such as production and sales plans.
- The PDP is a risk management system that identifies, prioritizes and reduces uncertainties and their effects.

They also point out several reasons for having a PDP: quality assurance, coordination between functions, planning to allow for scheduling, a basis for management monitoring, and support of improvements in future development efforts (Ulrich and Eppinger, 2012, pp. 12-13).

Examples of PDPs abound. Pahl et al. (2007, p. 130) propose the following PDP steps:

1. Plan and clarify the task
2. Develop solution principle (conceptual design)
3. Develop the construction structure (embodiment design, part 1)
4. Define the construction structure (embodiment design, part 2)
5. Prepare production and operating documents (detail design)

Crawley et al. (2016, p. 188) organize their relatively wide-ranging "generic PDP" around different categories of product development activities rather than a strict timeline:

1. Conceive: determine what will be built

    a. Mission
    b. Conceptual design
2. Design: define what is to be implemented
    a. Preliminary design
    b. Detailed design
3. Implement: convert the design into reality
    a. Element creation
    b. Integration, test, positioning
4. Operate: operate the real system to deliver value
    a. Lifecycle support
    b. Evolution

The "complex system" version of the generic PDP from Ulrich and Eppinger (2012) uses the following steps:

1. Planning
2. Concept development
3. System-level design
4. Design and test (multiple subsystems in parallel)
5. Integrate and test
6. Validation and production ramp-up

Cross (2008, p. 207) reproduced the PDP from British Standard (BS) 7000, 'Guide to Managing Product Design', which is similar to the "generic PDP" from Crawley et al. (2016) in the extent of its scope:

1. Motivation/Need
    a. Trigger
    b. Product Planning
    c. Feasibility Study

2. Creation
    a. Design
    b. Development
    c. Production (manufacture, assembly, test)
3. Operation
    a. Distribution (packaging, transport, installation, commissioning)
    b. Operation (use)
4. Disposal

Before moving on, the distinction between product development and technology development is made. Whereas an enterprises uses product development to create something to sell to external customers, technology development is predominately concerned with creating artifacts that can be utilized by product development within the enterprise or perhaps sold to other enterprises. Technology development leans in a more scientific direction than the aerospace industry's engineering design-oriented notion of product development. While not necessarily of immediate benefit to consumers in and of themselves, technologies are key ingredients in the products that *are* valuable to consumers. Put another way, there is a difference between flight and manufacturing technology selection (which is under the purview of product development) and flight/manufacturing technology development. This is depicted by Figure 6, which illustrates how several technology development efforts could be ongoing to mature those technologies sufficiently to eventually include them in the development of the product.

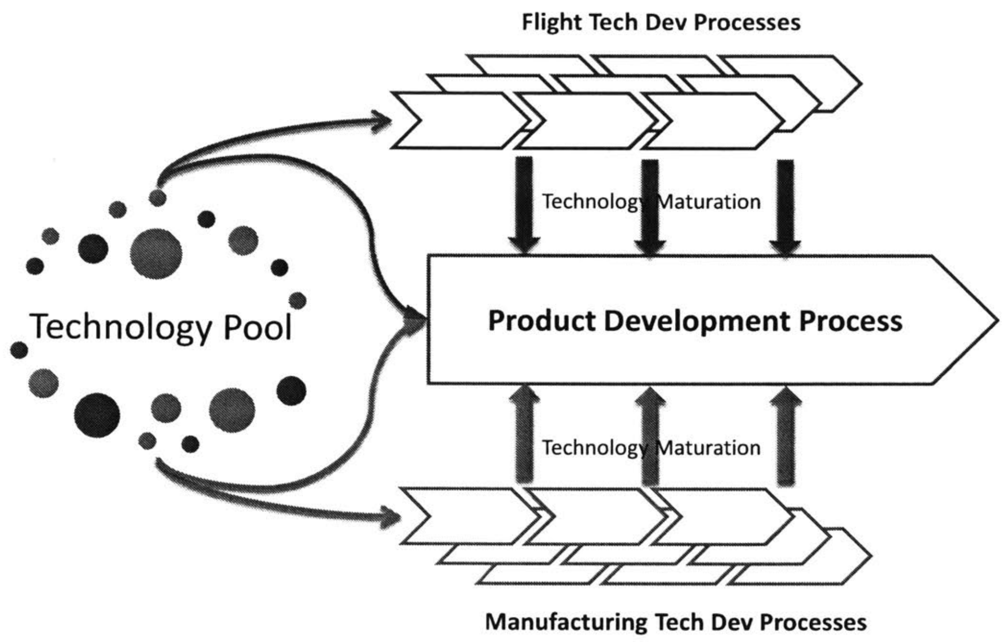

Flight Tech Dev Processes
Technology Maturation
Technology Pool
Product Development Process
Technology Maturation
Manufacturing Tech Dev Processes

Figure 6: Multiple technology development processes feeding into the product development process

2.2.2 Product Development Projects as Systems: The Design System

Just as there exists the familiar concept of a Production System that consists of all the raw materials, storage and distribution centers, tooling, assembly rigs, workers, and other elements responsible for producing the product, it is helpful to regard Product Development itself as a system of elements working together to develop the product. The Product Development System includes the PDP but is more than just the PDP; process is only one aspect of product development.

There is substantial basis for this viewpoint in the literature. Browning et al. (2006, p. 107) reference the ZOPH model that identifies the product, process, agent (organization, technologies, resources, methods, tools, etc.), and goal domains that make up any project or program. Danilovic and Browning (2007) show the agent domain divided into separate Organization and Tool domains. Crawley et al. (2016) describe a "global PDP" that moves beyond the process to consider all of the following attributes of the product, the product design activity, and the product implementation activity:

- Needs
- Goals
- Function/process methods/flow
- Form/tools
- Behavior/schedule
- Operator/team
- Costs

Any project is a system consisting of five domains, and project development projects are no exception. An overview of the project domains and the interactions between them are depicted in Figure 7 in the style of Danilovic and Browning (2007). These domains make up the Product Development System:

- Goals: This domain houses the objectives (e.g. product performance goals, development cost measures, design team capability targets) and constraints (e.g. product price cap, development schedule limits, manufacturing constraints) pertaining to the other domains.
- Product: The primary desired outcome of the Product Development System, the product has at least two versions: the product design (information) and the built product (reality). The Product domain is created by the Process domain.
- Process: The Process domain is interchangeable with the PDP and consists of the series or sequence of transformations and tasks that eventually yield the product. Also, upstream activities in the process define downstream activities. A traditional PDP can be used to structure the Process domain. The Process domain is enabled by the Resources and Organization domains.
- Resources: This domain covers the enablers (tools, equipment, etc.) and controls (policies, standards, etc.) – in IDEF0 terminology (Menzel and Mayer, 2006) – that embody the Process domain and are wielded by the Organization domain, such as

design methods, manufacturing technology and integration/test assets. Resources are mapped or assigned to elements of the Process. Note that the term 'Tools domain' from Danilovic and Browning (2007) is intentionally not used here, and replaced with the more generic term 'Resources', in order to allow for less confusing subsequent usage of the term 'design tools' (to be defined).

- Organization: Those involved in the product development project are organized into teams, groups, etc. by this domain. Members of the Organization are mapped to and perform elements of the Process and they use Resources – indeed, many times groups deem themselves as belonging to a particular segment of the Process and claim certain Resources as their own. This domain considers team skills, functions, work assignments, and staffing levels.

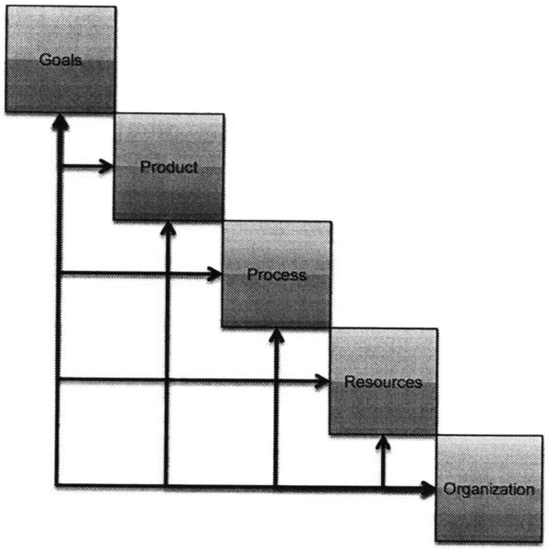


Figure 7: The product development project's interacting domains

Although in principle each domain interacts with every other domain in the project, as shown in Figure 7, in practice some transformations and assignments are more prominent than others, as portrayed in Figure 8 in the style of Danilovic and Browning (2007). In Figure 8, the "generic PDP" from Crawley et al. (2016) is used to decompose the Process domain into Conceive, Design, Implement and Operate sub-domains; the process Conceives and Designs the product design and Implements and Operates the built product. The Organization domain is mapped or "designed" to execute the Process domain. Resources such as design tools and manufacturing equipment are assigned to facilitate the Process. Thus, the PDP is not only decomposable into phases – the traditional understanding of product development – but it also interfaces with the other domains of the Product Development System. Within the Process domain, upstream tasks define the exact form of downstream tasks, as in the case of downstream design work planning, manufacturing process design, supply chain decisions, and integration/test planning.

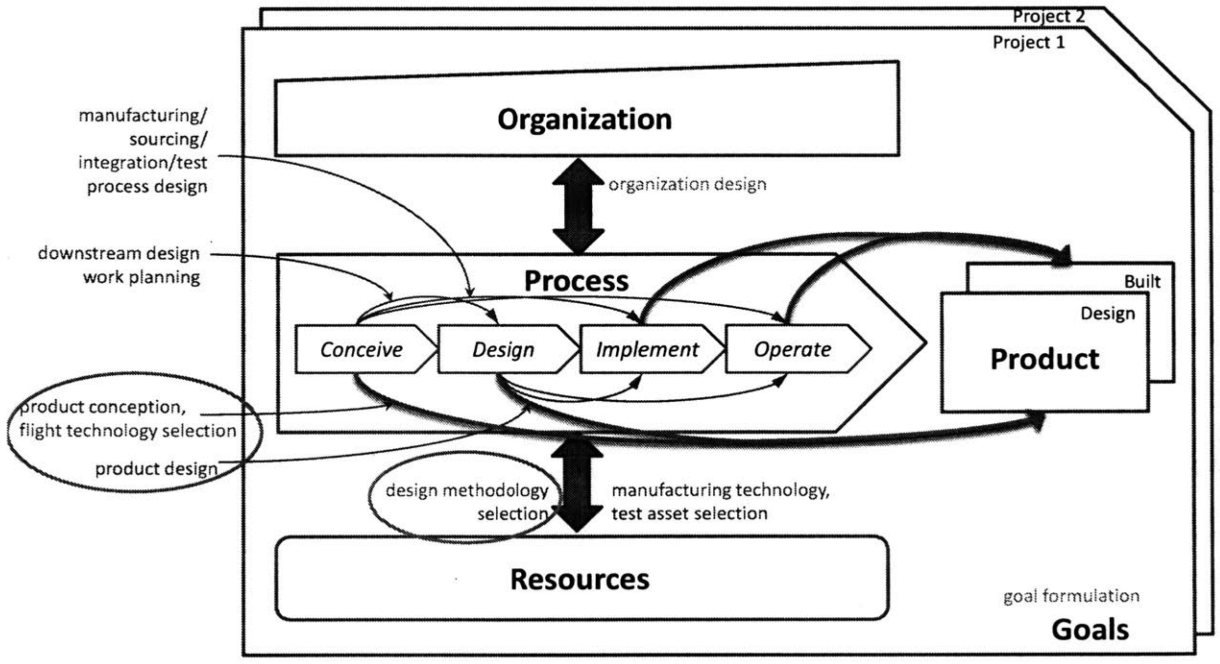

Project 2
Project 1
Organization
manufacturing/ sourcing/ integration/test process design
organization design
downstream design work planning
Process
Conceive
Design
Implement
Operate
Built
Design
Product
product conception, flight technology selection
product design
design methodology selection
manufacturing technology, test asset selection
Resources
goal formulation
Goals

Figure 8: Product Development System, with primary interactions between domains of product development projects shown and research scope circled

Refocusing back on product conception and design (circled in Figure 8 and referred to collectively as "product design"), it is now evident that there is a product design "cross-section" that can be viewed across the Product Development System, as depicted in Figure 3. The product design elements of the Process domain are essentially "cast" to the other domains, generating a viewpoint of all product design facets of the Product Development System. This product design perspective is not just the "design phase" of the PDP – it includes all the aspects of and interactions between the project development project domains that are geared towards product design activities. This cross-domain "product design subsystem", or simply design system, is subordinate to the overall Product Development System and is comprised of the design Goals, the Product design itself, the design Process, the design Resources, and the design Organization. As shown in Figure 3, there are three types of design Resources in particular that are important to differentiate:

- design methods (discrete problem-solving approaches),
- design tools (implementations of methods), and
- design-enabling platforms (generic assets used to build tools).

This design system framework reinforces the point that design is essential to yet distinct within product development. Cross (2008, p. 204) states, "...it is important to realize that 'design' is only a part of a larger process of product planning and development." The PDPs previously described all designate design as a distinct step(s). Design is placed next to Marketing and Manufacturing as three key functions that participate together in product development, where the Design function conducts design of the product itself (as opposed to the Manufacturing function's design of the production system, or other forms of "designing" that take place during product development) (Ulrich and Eppinger, 2012, p. 3). Furthermore, design exists at different levels of product/system abstraction and along various "planes of decomposition" (e.g. Crawley et al. (2016, p. 302)) – high-level total aircraft aerodynamic performance design is different from (though it may be coordinated with) low-level airframe stringer structural design.

The concept of the design system also provides a basis for applying lessons from design science research. If product design in product development is essentially a system, the implication is that the design system itself can be "designed". This notion is usually referred to as "design process improvement", but the more precise terminology of the design system framework prevents a narrow focus on Process (tasks) by explicitly counting Goals, Resources and Organization as well. Scenarios can be posed to demonstrate instead how "design *system* improvement" can be reasoned through; for example, from a design Resources standpoint:

- Fixing everything else constant (e.g. design goals, design staff experience level, productivity), what tradeoffs (in terms of project performance measures like cost or duration) exist for various possible design tool configuration choices?
- After selecting the design tools, do you then adjust the design organization? Or the design process? Or should both be adjusted simultaneously? Or should you pick tools simultaneously with formulating the organization and process?
- How can you improve design method/tool integration within the design Resources domain, to alleviate the manual labor burdens of complex system development and control quality? Are more integrated (interconnected, not necessarily "un-modular") methods/tools better than simpler tool connections? How should one decide whether design tools should be unified, federated, or unconnected? What is the right methods "structure" or set of connections between methods?
- Should design method choices always follow the prior establishment of the design process? Or can the design methodology drive the design process in some cases?

2.2.3 Model of Design

2.2.3.1 Defining 'Design'

To explore avenues for applying guidance from design research on the design systems employed by aerospace development projects, it is illuminating to take a step back and recall some common definitions for the general term "design" and some basic models of design. Ulrich (2011, p. 2) states, "Design is conceiving and giving form to artifacts that solve problems." Crawley et al. (2016) would portray design as the transformation of an ambiguous information vector (influences on system architecture) into the precise information vector ("the design") describing the product to be implemented. According to Hubka and Eder (1996, p. 4), "The task of the designing consists of thinking ahead and describing a structure, which appears as (potential) carrier of the desired characteristics (properties, particularly the functions). One can express this statement also in process terms: designing is defined as the transformation of information from the condition of needs, demands, requirements and constraints (including the demanded functions) into the description of a structure which is capable of fulfilling these demands."

Most general models of design address the process of designing. Wynn and Clarkson (2005) use several dimensions to categorize these design models based on whether the model is:

- Stage-based (project lifecycle phases, chronology, morphology) vs. activity-based (iterative problem-solving, cyclical, rework-intensive, day-to-day)
- Problem-oriented vs. solution-oriented
- Abstract (generic), Procedural (specific), or Analytical
- Descriptive vs. Prescriptive
- Design-Focused (improve product) vs. Project-Focused (improve design management)

Using these dimensions, they recognize patterns among the design models in the literature, such as:

- Stage-based models are usually problem-oriented
- Procedural approaches are usually problem-oriented
- Abstract approaches are usually activity-based

As an example of a design process model, Cross (2008) presents an "integrative" model with the following stages:

1. Overall problem
2. Sub-problems

3. Sub-solutions
4. Overall solution

Wynn and Clarkson (2005) provide several examples of these design process models from the literature that can be used to find patterns in an aerospace system design process:

- Jones's model has three activities that flow in serial order: first analysis, then synthesis, then evaluation. It is an example of an abstract, problem-oriented, activity-based model.
- Cross's 1994 model has four serial activities – exploration, generation, evaluation, communication – with iteration between generation and evaluation. It is also abstract, problem-oriented, and activity-based.
- Darke's model has three serial activities: generator (solution to a subset of design objectives), conjecture (full solution based on generator results), analysis (improvement on full solution). It is an abstract, solution-oriented, activity-based model.
- March's model is an abstract, solution-oriented, activity-based cycle of three activities:
    1. Production: abductive reasoning; propose a candidate solution based on available information
    2. Deduction: deductive reasoning; apply physical principles to determine the performance of the candidate solution
    3. Induction: inductive reasoning; identify possible means of improving the performance of the current candidate solution

Of course, there are also many domain-specific and enterprise-specific design process models. In the end, Wynn and Clarkson (2005, p. 35) admit, "Despite the extensive research undertaken since the 1950s, there is no single model which is agreed to provide a satisfactory description of the design process."

Now, the design system framework (referring back to Figure 3 and subsection 2.2.2) reveals that product design is comprised of the design aspects of the domains that make up the product development project. However, each of the design system domains presents different opportunities for improvement:

- Design Goals: usually predetermined by the development program
- Product design: this is the deliverable of the design system to be ultimately improved
- Design Process: With many variations and debates on how to improve the design Process proper (e.g. see Krishnan (1993), Eppinger et al. (1989), Unger (2003)), the way forward on Process domain improvement is many ways clearer than that for design Resources; nonetheless, it is important to model the design Process when pursuing design system improvement.
- Design Resources:
  - Methodology: this thesis explores significant opportunities here
  - Toolset: the value of a design tool is driven by the value of its underlying design method
  - Design-Enabling Platforms: frequently more adeptly handled by the software/hardware developers
- Design Organization: much existing discussion on how to improve this as well

2.2.3.2 Design Process

Again, it is usually important to model the particulars of the design Process at hand during a design system project improvement effort because the design Process is the main mechanism by which design Resources and the design Organization affect the product design. The design process can be modeled as a network of objects and transformations on those objects, not dissimilarly to Object-Process Methodology:

- Design Artifact: This is information (an object) about or leading to the product design that can be converted from or into other artifacts, usually denoted by a noun. Example artifacts:

    - Customer brief
    - Market data
    - Solution concept description
    - Concept analysis results
- Design Activity: This is a transformation or conversion experienced by an artifact, usually denoted by a verb. Example activities:
    - Interpreting
    - Collecting
    - Synthesizing
    - Analyzing
- Design Task: This is a particular transformation of information – the combination of a design activity with its corresponding input and output design artifacts. One finds that common text labels or descriptions for tasks are frequently unable to capture all of the inputs and outputs involved in a task. Example task labels:
    - Interpret customer brief
    - Collect market data
    - Synthesize solution concept
    - Analyze solution concept

Hence, the design Process consists of design artifacts and design activities combined to form tasks. Figure 9 illustrates a notional design process using the defined syntax, including complete task descriptions (in graphical, not text, form). Note that multiple tasks could interact with the same artifact (e.g. both tasks 2 and 3 contribute to outputting artifact 6 while task 4 uses artifact 6 as an input), but each task has only one activity. Artifacts from external sources that are not created by any activity in the design process are known as exogenous inputs, such as Artifacts 1, 2 and 4 in Figure 9.

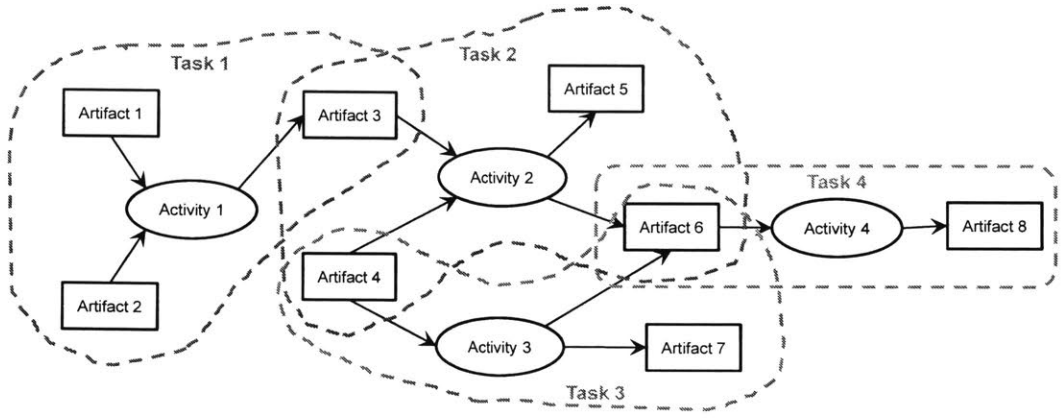


Figure 9: A notional design process composed of design artifacts and design activities

A directed graph of tasks can be induced from the process in Figure 9, as shown in Figure 10. This exhibits how the design process is indeed a network of design tasks, but it also demonstrates how important details of the information exchanges that are occurring throughout the design process can become hidden or implicit knowledge instead of explicit in the model representation. For example, exogenous inputs and outputs can be left out, as well as shared artifacts among activities.

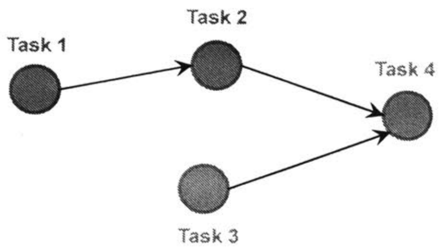


Figure 10: The notional design process reduced to graph form

Recall that Ulrich and Eppinger (2012) characterized the PDP as an information processing system that transforms inputs into outputs. Krishnan (1993, pp. 7-8) agrees: "Design of complex products requires resolution of a multitude of conflicting considerations, thousands of decisions, and reasoning and search in multiple domains; the very scope and complexity necessitates that knowledge and data be distributed, and design be divided into a set of interacting tasks or information exchanging activities." Modeling the design process as a network of information transformations supports this viewpoint: as seen in Figure 9, design information is exchanged across design activities and tasks.

2.2.3.3 Design Methodology in General

This thesis asserts the potential of using design methodology as an avenue for design system improvement. A design methodology is simply the set of individual design methods (and the interactions between them) used to solve the tasks in a design process; in other words, design tasks progressively transform information *by means of* design methods. A design method is a technique that applies first principles to solve problems. From Schmidt (2015, p. 19): "...'a method is a procedure for attaining something' and 'a methodology is a body of methods employed by a discipline'." As Hubka and Eder (1996, p. 132) state, "We define the term design method to mean a system of methodical rules and instructions. These are intended to guide and/or determine the way of proceeding for executing a certain design activity and to regulate the interaction with the available technical means... A coordinated set of methods is termed as a methodology." According to Andreasen et al. (2015), "A method is a goal-oriented rationalization or simplification of engineering work in the form of a standardized work description." Pahl et al. (2007, p. 9) spell out the broader implications of methodology: "Design methodology...is a concrete course of action for the design of technical systems that derives its knowledge from design science [or 'scientific design'] and cognitive psychology, and from practical experience in different domains. It includes plans of action that link working steps and design phases according to content and organisation."

A large number of design techniques fit the definition of a design method here. For examples of specific design methods and method catalogs, see Jones (1992), Cross (2008), Pugh (1991), Hubka and Eder (1996), Schmidt (2015), and Ulrich and Eppinger (2012). For instance, Franke and Deimel (2004) considered the following design methods:

- abstraction and problem formulation
- abstract function structure
- brainstorming
- specific function structure
- design catalogue
- conflict modelling (TRIZ)
- synectics
- morphological box
- cost-benefit analysis
- evaluating by points
- literature research

Design methods can be categorized along several dimensions (see Schmidt, 2015). Example dimensions include:

- Generic vs. discipline-specific: A generic method can be applied to any product or to a system as a whole, and/or it can find widespread application across a range of traditional design-related disciplines (mechanical engineering, software, industrial design, etc.). In contrast, specific design disciplines utilize many methods that are meant for their own fields.
- Structured/formal vs. unstructured/informal: A structured or formal method is characterized by the presence of a defined procedure, whose result is less dependent on the specific people involved; unstructured methods usually have no formal procedure and outcomes more heavily influenced by those who execute the method (Schmidt, 2015, p. 26). Design tools and enabling platforms also tend to be more difficult to define for unstructured methods.

- Type of design activity or task usually associated with the design method: Examples of ways to categorize methods by their function include:
  - Type of reasoning (i.e. March's model): deductive, inductive, abductive
  - Activities from Jones (1992):
    - Strategy control
    - Exploring design situations (divergence)
    - Searching for ideas (divergence and transformation)
    - Exploring problem structure (transformation)
    - Evaluation (convergence)
  - Activities from Andreasen et al. (2015, p. 54):
    - Creating a design goal
    - Creating product ideas and concepts
    - Decisions and selection
    - Evaluation of product features

Additional emphasis is placed on distinguishing design methods from "design tools" and "design-enabling platforms", which are displayed in the context of the design system domains in Figure 3. A design tool is a particular manifestation of a design method that the designer directly interacts with. The method encapsulates the "theory" behind the tool (or tool feature, in the case of a tool that implements multiple methods), while the tool helps the designer actually execute the method and solve a design task. Multiple tools could implement different varieties of the same underlying method, and a single substantial tool could implement multiple methods to, say, allow the designer to use one tool to perform several tasks.

A design-enabling platform is a generic asset used to build design tools. Platforms come in the form of software packages that have aerospace design applications, as well as hardware equipment (e.g. building blocks, machining equipment) that can be used to create design tools.

Multiple tools could be built from one platform, or one tool could require multiple platforms to build. For example:

> The vortex lattice method for determining configuration aerodynamic characteristics could be implemented with a software design tool that uses a particular numerical solver and has a specific user interface. The tool itself is coded in the numerical analysis software application MATLAB, a commonly-used enabling platform.

While an individual design method is nominally used to execute a single design task, a combination of methods (the design methodology) is used to execute a network of tasks (the design process). Like the design process, the design methodology is a network of methods with structure. According to Franke and Deimel (2004), methods can be combined by matching outputs from one method with inputs to another, as illustrated in Figure 11. These inputs and outputs are the same parcels of information exchanged and converted by the design process (i.e. design artifacts). The input/output structure in Figure 11 was contrived to be consistent with the example design task network in Figure 9, employing one method to solve each task.

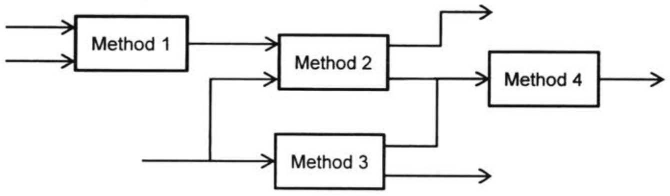


Figure 11: The notional design methodology

2.2.3.4 *The Role of Methods in Design*

A design method uses problem-solving principles to perform design tasks. At the level of abstraction of the overall design system, the design process is "method-neutral" – the design tasks only specify *what* the information flow is to mature the design; the tasks themselves do not stipulate *how* (i.e. what methods to use) to mature design information from task to task. Ernzer and Birkhofer (2002) would say that it is required for methods to support the design process:

"...a methodical support of the designer is indispensable" (p. 1305). Lahonde et al. (2010, p. 1) say, "Design methods are inseparable from product development." Put another way, the "sub-problems" in the Cross (2008) "integrative" design process model are 'solved' by design methods – the use of methods and the generation of sub-solutions eventually integrate together to form the overall solution. Note that the same method could apply at multiple levels, i.e. recursively.

A design method is assigned to a design activity as its instrument to carry out the design task; methods are enablers of activities, and one can begin drawing finer distinctions such as separating the activity domain from the method domain (more on this to follow in section 4.4). See Figure 12 for a more detailed look at the interface between the design system's Process and Resources domains using the same notional design process and methodology (or design strategy) from Figure 9 and Figure 11. Figure 12 uses syntax similar to Object-Process Methodology (OPM) (Dori, 2011) to denote how Method 1 is assigned to Activity 1 (and by extension Task 1) as the enabler of Activity 1, Method 2 is the enabler of Activity 2, and so on. One method is assigned to each design activity, for this particular design strategy; other design strategies would map other methods to the design process differently. Possible design toolset and design-enabling platform configurations are also depicted in Figure 12, where Tools 1 and 2 each implement more than one design method and are both built from one design-enabling platform. Note that in this example, the designer needs to interact with two design tools to execute Design Task 2 because the underlying method that is solving Task 2 is implemented across two tools – a situation not uncommon in aerospace design practice.

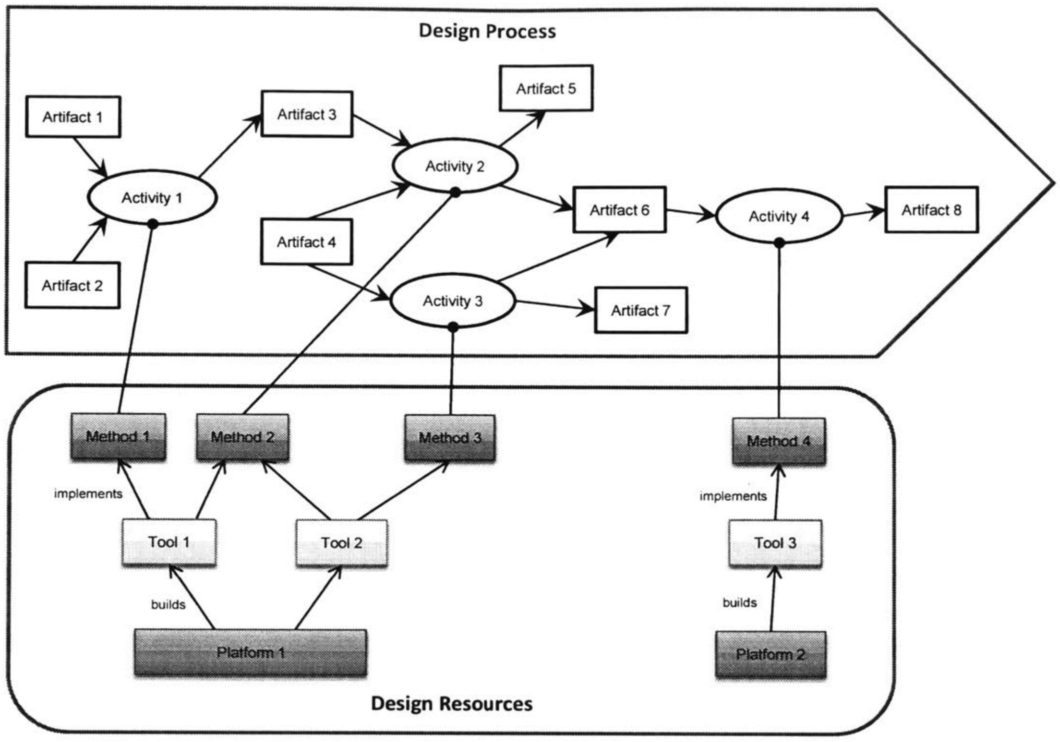


Figure 12: Notional design Process and Resources domain interactions, depicted using OPM syntax

Therefore, the way in which design methods are integrated into the design system has a significant impact on the performance of the product development project. Wynn and Clarkson (2005, p. 46) make it clear: "According to Pugh (1991), successful product design is subject to the integration of such general design methods with traditional engineering expertise." In fact, the union of the design methodology with the design process is sometimes called the design strategy. For Jones (1992, p. 75), design strategies lay out the "design actions", how methods are to be used to execute a design, and are characterized by the degree of pre-planning involved and the pattern of search employed. Cross (2008) has a similar definition for design strategies, remarking that 'prefabricated' and 'random search' (i.e. rigid vs. flexible frameworks) are the two ends of a spectrum that design strategies fall on; deciding which framework is appropriate depends on the design situation and the team's skills. Ulrich and Eppinger (2012) present a design strategy throughout their work that falls somewhere in the middle of that spectrum.

The structure and flow of design tasks usually drive how the design methods and tools are integrated together and their sequence of execution. Conversely, different methodology structures could inspire design process changes. The two domains are coupled, implying that simultaneous planning of both aspects of the design strategy could be iterative.

A distinction should be made between a design methodology populated by structured methods and a structured approach to methodology. The former is a property of the design Resources domain; the latter is a design team and management issue that has the promise of generally enhancing the design methodology and, by extension, the overall design system. Of course, the appropriate degree of formality for the methodology is a subject of debate for the design team to wrestle with, as Hubka and Eder acknowledge:

> In the context of the systematic discursive instructions for procedure, the question must emerge whether the whole design process is algorithmically solvable. Franke denies this on the grounds that 'algorithm' should only be interpreted in its strictest sense of machine instructions. Even so, procedural models of designing have emerged and continue to emerge, as 'flexible algorithms,' not only in design strategy but also in design tactics. (Hubka and Eder, 1996, p. 133)

This leads to a discussion of the latter issue: the search for a deliberate approach to design methodology. Lahonde et al. (2010, p. 1) believe that doing so helps optimize the design process and improve enterprise performance by allowing designers to "structure and rationalize their activities", reducing development time, minimizing errors, improving product quality, and reducing costs.

2.2.4 Design Method Transfer and Selection

A structured, deliberate approach towards design methodology is supported by design method transfer (or method deployment), the overall process by which design methods are put into practice by designers, i.e. how a method goes from an unused state to a utilized state. The designer may or may not be aware of these to-be-transferred design methods, some of which may have been hitherto limited to academia, hence the term 'transfer' to denote the conveyance of methods from academia to industry (Stetter and Lindemann, 2005). Braun and Lindemann (2003) comment that current forms of design method transfer usually involve impersonal transmission of product development method knowhow via databases, libraries, multimedia, technical literature and websites. These impersonal means may organize methods into product lifecycle phase categories or bins, with no further guidance. The designer has to resort to entering deeper into method descriptions to determine the suitability of any particular method, but there are too many details to evaluate multiple methods; there is insufficient capacity for this to be a viable approach. At this point in the discussion, they introduce the "Munich Model of Methods" (MMM), which relates tasks in the design process with method deployment steps and method characteristics. Their method deployment steps are:

1. Selection
2. Adaptation
3. Application

Method characteristics (or "attributes") include:

- Input
- Procedure
- Output
- "Resources" (e.g. users, skills, experience)
- "Support" (e.g. descriptions, examples, tools, forms, hints)

According to the MMM, properties (or "boundary conditions") of the design task and characteristics of the method are compared and aligned to perform each method deployment step. To follow this paradigm, Braun and Lindemann (2003) argue that a method transfer support system that presents method knowledge in a structured, modular fashion is needed, such as the Process-oriented Method Model (PoMM) from Birkhofer et al. (2002), which describes product development methods using a standardized and comprehensive framework.

Stetter and Lindemann (2005) present a broader five-stage model of design methods transfer:

1. Initiation of the method implementation process (identification of strengths and improvement potential, establishment of moderators and team building)
2. Analysis of the product development system (search for inadequacies in the PDP)
3. Choice and adaptation of methods
4. Implementation of methods
5. Evaluation of the impact

Gericke et al. (2015) see two major approaches to the transfer of design methods: textbooks and web-repositories; and community-based web platforms. They report a mismatch between how designers discover methods (by outcome) and the presentation of methods in textbooks and web repositories (by title or process stage). Personal contacts and trusted relations end up dominating the search for new methods, though sharing experiences is easier said than done. This observation nonetheless calls for greater attention to be placed on ensuring that the needs of designers are met during the earlier stages of method deployment.

The choice of method for a given design situation is a critical, distinct step early on during design methods transfer. The issue at hand is how to perform the mapping in Figure 12 of design methods to design tasks. Braun and Lindemann (2003) lay out three possible courses for the step of selecting methods in particular:

1. By assignment to a superior process: classify methods by which stage of the design process each one is usually associated with, then identify which process stage the current design task belongs to in order to match methods with tasks. This is the classic approach to method selection, where methods are usually mapped to the PDP (see Schmidt (2015)). It can be difficult to classify methods this way – a given method can fit in multiple stages, especially when the method is multidisciplinary. Lahonde et al. (2010, p. 1) note that, historically, the first method selection guides were based on decomposition of the design process where a set of methods are recommended for each process phase – the same basis for the majority of today's guides. For example, see the correlation of design process steps with methods from Pahl et al. (2007, pp. 564-565), Hubka and Eder (1996, p. 169), and Schmidt (2015, pp. A39-A44).
2. By assignment to method attributes and states: compare method attributes with design task attributes to select an appropriate method. Attributes could include inputs, outputs, staff skill level, and duration to complete/execute. This approach lends itself to modular implementations, being easy to deposit into method selection decision support system databases, especially if standardized terms are used to characterize methods and tasks. This approach can be considered a superset of the superior process assignment approach, if a method's design process classification is one of its attributes. Lahonde et al. (2010) also remark on the use of method attributes, describing attributes as items of information or properties such as inputs/outputs, type (e.g. creativity, multi-criteria decision-making), and nature and quantity of resources (time, human, material).
3. By assignment to elementary working steps: correspond a task's internal activities with the elementary steps that make up a method to determine which method is appropriate. Instead of an attribute-based view, this approach takes a process-based view by regarding both design tasks and methods as being comprised of elementary, nominally inseparable "sub-tasks" or working steps. Though design tasks – especially for design

systems with higher level scopes – are usually easily decomposed into sub-tasks, this approach raises the question of whether every method is also decomposable into a procedure. Even if a method's elementary working steps can be articulated, will the articulation be unique? What if the design task and a method happen to be decomposed into different sets of elementary steps when in fact they are fundamentally compatible? Braun and Lindemann (2003) comment that, thus far, research along this process-based view of design method transfer has addressed situational adaptation and combination of methods that occur further downstream in the design methods transfer process, not necessarily the upfront selection of methods that has to take place first. Franke et al. (2003) and Franke and Deimel (2004) take up this approach by correlating design tasks with methods by the "basic activities" that made up both groups. They differentiate between main basic activities ("...essential to receive the result...") and supporting/assisting basic activities ("...supports only the procedure to receive the result"). Examples of basic activities include prioritizing, abstracting, evaluating, documenting, analyzing, presenting logical chains, varying, informing, finding analogies, combining, substantiating, and selecting – different tasks and methods consider different basic activities to be "main" as opposed to "supporting".

Stetter and Lindemann (2005, p. 444-446) argued for a multi-criteria approach to design method selection based on the "essential, invariable characteristics" of a method:

- Evaluation of the function
- Evaluation of the information
- Evaluation of the procedure
- Evaluation of the applicability

Gericke et al. (2015) explored the following factors affecting design method selection:

- Availability of required resources

- Impact on the design process
- Required expertise/competence
- Expected financial benefits
- Personal benefits from applying the method
- Management support
- Experience with similar methods and gut feeling
- Recommendations of colleagues and peers

Their empirical study had several findings:

- Specific guidelines for selecting among new methods, if they existed, were only relevant for methods with large impact or high implementation costs. In those cases, an estimate of the expected benefits (e.g. time savings, quality improvements) from using the new method were required, though no review of actuals relative to expectations after method implementation was conducted.
- The most important criterion to consider when selecting methods was method efficacy, but even this was assessed informally.
- Method choices had to fit within constraints. The methodology had to be compatible with the existing design process, as realization of the need for specific methods usually occurred in the middle of projects. Some methods were required by the Board, customer mandate, or legal requirements; no assessment was done prior to the "selection" of these methods.

Jones (1992, pp. 79-83) developed a set of design process stages and a matrix mapping of design methods to those process stages to support method selection. Interacting with essentially an "input/output chart", the designer indicates which design process stages they are starting from and looking to end with to determine the row and column of the matrix cell that contains applicable design methods from which to select from.

Ernzer and Birkhofer (2002) propose progressively refining the pool of candidate design methods for a design situation in three stages:

1. Organize all available methods into a standardized method pool: screen existing methods based on designer suitability and maturity for the PDP, using a master template based on a unified method model (e.g. the Process-oriented Method Model from Birkhofer et al. (2002)) to consistently assess the methods and organize information
2. Filter the standardized method pool down to a company-relevant method set: elicit specific company needs (attributes) to produce method attributes and requirements that are used for deciding which methods from the standardized pool are company-relevant. Mapping company needs to method requirements is not easy because attributes are interdependent and not equally important.
3. Select a method mix from the company-relevant method set based on project-specific criteria

It is expected that the first two stages would be completed once initially for an enterprise while the third stage would be executed for each product development project. They tested their proposal by comparing the design method mix that resulted from applying these three stages in an academic setting with the design methodology used in industry, for the field of Design for Environment. They found that the methodology configured by academia was more defined and explicit in completing every traditional step in the design process than the industry methodology.

Recent studies explore the use of data analytics and machine learning to provide design method recommendations that mimic those from the tacit experience of seasoned designers (Fuge et al., 2014).

A common theme that emerges from the design method selection literature is the prominence of using attributes to characterize methods and as criteria when choosing methods. According to the definition from Braun and Lindemann (2003, p. 4), "As method attributes, we regard a set of characteristic information and properties which specify a method. In literature method attributes are also addressed as method characteristics or features." To further define terms, while the name of the method property itself is called an attribute, the value or condition of that attribute is called the attribute's state. For example, one attribute of a method could be its output quality, which can take on states of 'low', 'medium' or 'high'. The attribute 'output quality' could also be decomposed into sub-attributes such as output completeness quality or output accuracy quality.

3 Motivation and Thesis Formulation

This chapter reviews the present research need and the gap in existing literature. It also formulates the thesis, posing the fundamental research question that fills the research gap, laying out the strategy and actions to take to answer the research question, and describing techniques for carrying out the research strategy.

3.1 Fighting Cost Growth by Improving Design and Design Methodology

As discussed in section 2.1, the product design and the design practices that generate it are substantial influences on aerospace development project costs. Simply put, better design practices yield better development project performance and directly address the problem of development cost growth. Improving the "design system" described in section 2.2 curbs rising development costs by promoting reduced design cycle times and system lifecycle cost benefits associated with first-time quality of the system design.

Aerospace design teams and managers have significant opportunities to pursue design system improvement by means of design methodology, a field of inquiry with relatively unexplored potential in industry practice that has significant leverage for a variety of reasons. Firstly, design methodology holds a strategic place in the design system: the methodology sets the theoretical capability of the design system to "problem-solve" the design process. In other words, better design tools or more efficient design-enabling platforms do not equate to better design methods. Strictly speaking, tool improvements or enabling platform upgrades cannot surpass the fundamental assumptions and limitations behind their underlying methods. Continuing with the vortex lattice method example: regardless of how much more numerically efficient the designer makes the software design tool or whether the computational environment in which it is coded is upgraded or not, the designer will still ultimately be limited by the fundamental physics assumptions behind the vortex lattice method. Sometimes enterprise investments focus too much on tools and enabling platforms at the expense of methods, which can lead to "design infrastructure" lock-in – the potential of tools and enabling platforms to drive down development costs is driven by the methodology they rest on.

Secondly, better design methods can better tackle today's "wicked" aerospace system design problems. Jones (1992) describes four issues that motivate the need for new methods:

1. Modern design problems are more complex than traditional ones, both externally (e.g. technology transfer, standards/compliance, sensitivity to human overlap) and internally (e.g. heightening investment, difficulty in discovering rational decision sequences).
2. Solving modern design problems involves more interpersonal obstacles and conflict between stakeholder needs.
3. There are now new kinds of complexity outside the scope of traditional design, where the space to search for new system solutions is too large and unfamiliar.
4. Traditional design copes with complexity inadequately by using tentative, simplified solutions to rapidly explore the problem and solution spaces

Design methodology investment can also improve design cycle times and team capacity by "operationalizing" product development to some extent. A reliance on organic designer experience alone to navigate the design methodology space no longer supports the pace of today's aerospace market and customer. A structured approach to design methods is called for, including an emphasis on what Hubka and Eder (1996) label "discursive" or rational methods that are logic-driven, as opposed to intuition-driven, and therefore transferable, learnable, transparent and repeatable. The findings of Gero et al. (2013) state, "...the more structured a concept generation creativity technique is, the more likely that designers using this technique tend to focus more on problem-related aspects of designing, i.e., design goals and requirements." Franke and Deimel (2004) continue: "The fast development of innovative products requires an efficient application of methods during the entire development process."

Put another way, not only is there benefit from improving the design team's methodology, but there is also cost associated with utilizing "the wrong methods", i.e. those that are inappropriate or suboptimal for the situation, which can contribute to design errors, rework, delays, and additional costs. Braun and Lindemann (2003, p. 2) concur: “For a problem-oriented and successful deployment of methods, the selection, adaptation, and application should always focus on the underlying situation and consider the boundary conditions. The use of unsuitable methods may lead to a loss of acceptance of methodical problem-solving in common.”

3.2 The Need for Method Selection Decision Support

3.2.1 The Objective: Recognizing and Expanding the Method Space

The design team/management has several avenues for tackling design methodology improvement. One way to categorize these various forms of potential advancement is revealed by inspection of the notional methodology and its interfaces in Figure 11 and Figure 12, as follows in "inside-out" sequence:

1. Upgrades of the individual design methods currently utilized by the design process
2. Improvements to methodology internal integration and use sequence among methods (which may drive design process adjustments)
3. Enhancement of the designer's means and ways for deploying (choosing, adapting, applying) methods to solve the design process's tasks
4. Advancements in the distribution of method implementations across the design toolset

Of the courses of action above, the third course pertaining to design method transfer or deployment was chosen for the focus of this research due to its significant leverage and potential. The third course addresses the fact that many methods are available for use by industry today – without the need for development of new design methods – but are untapped from lack of awareness. Meanwhile, outcomes from pursuing the other courses of action depend on the third course's upstream method deployment choices.

The design team conducts design method deployment under a variety of scenarios with real-world conditions. Facets of these scenarios are illustrated below:

- Given a design task, the designer selects a method to accomplish the task.
- Given a design task and a baseline method, the designer evaluates an alternative method for use instead.
- Given the set of all system design tasks (the design process), the system design team formulates the overall design methodology.

- The designer deploys methods while in the middle of a development project, under tight schedule constraints.
- The design team is in between development projects with fewer schedule constraints, preparing for future design efforts and building up design capacity by training in methods and planning design strategies.

These are all examples of how the method deployment depicted statically in Figure 12 could occur dynamically over time.

From a design organization standpoint, the *program* is responsible for maturing the product design and ensuring development program performance, while the *functions* are supporting the program with product development enablers and controls. The program deploys and executes the design methods to generate artifacts; the functions make design methods available for the program to use. The program will be most aware of any deficiencies with the design methodology and justification for improvements, but they do not necessarily have the expertise or even interest to make the improvements – their imperative is to produce deliverables and move on. The reverse is true for the functions: they are in the business of developing methods and tools but do not have that firsthand understanding of program method needs.

Further complicating matters is the uncertain nature of funding for methodology improvements, which depends on the balance between conflicting considerations of budget availability (externally-funded programs) and intellectual property protection (internally-funded functions). Sometimes the design organization's funding is phase-gated as well, which further discourages expending resources on methodology investments and anything else not seen as directly contributing to program deliverables. The net result is that there is a non-negligible cost for the design organization to do anything but minimize the design method deployment effort and continue using the current methodology already in place.

Therefore, although development programs see the mounting cost of unplanned iterations and other consequences of deficient design method deployment, the inertia of legacy design practices is still a dominant factor. This is the status quo of design method transfer: designers choose the methods they have always used, often implicitly. Method selection situations – like the scenarios described above – always arise during product development, and designers make choices whether consciously or not. The design leadership does not always make the design methodology itself something to be formulated and an open conversation topic amongst the designers. As Ernzer and Birkhofer (2002) point out, "The problem is that life cycle design becomes a very complex task since the whole product life-cycle has to be taken into account. Most designers in the industry do not have the knowledge and/or time to integrate (good) life cycle design principles in their day-to-day work."

Even if the design team is actively shaping their methodology or design process, these decisions are usually defaulted due to lack of perceived need to alter course on design methods, scarce resources or other project constraints, lack of confidence in alternative methods, or insufficient awareness of alternatives. When design tools and enabling platforms have already been configured for the limited range of design methods familiar to and trusted by the designers, many are "stuck" with the same methods because there is insufficient time to deploy new ones. Ernzer and Birkhofer (2002) assert, "The problem nowadays is not the lack of methodical support in product design, rather that of choosing the most suitable method from the many methods now available to the designer. Method selection today is often based on the popularity of a method rather than a real analysis of companies' needs."

The mission of an effort to improve design method deployment, then, should be a structured, deliberate approach that explicitly addresses the design organization's use of the method space while seeking to expand that space illustrated in Figure 13. Only methods that are considered and assessed can be selected to solve design tasks, and only methods known to the designers can be considered for deployment at all. Thus, expanding the design method space involves making more methods known to the organization, helping the organization assess as many methods as possible from those that are known, and supporting the decision-making that assigns considered methods to design tasks.

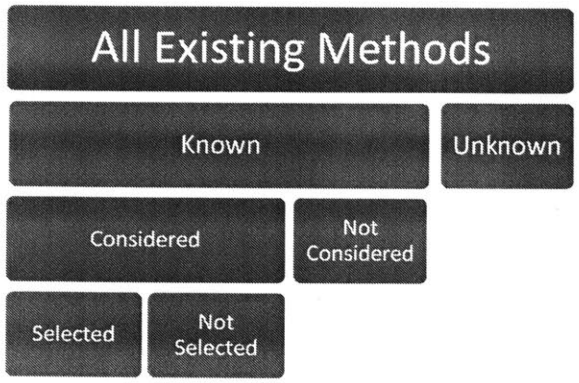


Figure 13: A decomposition of the design method space revolving around method selection

Design teams need to make overt, transparent decisions about design methodology – to make the method adoption process explicit rather than controlled by experienced but tacit knowledge. Methods should be picked based on all relevant considerations – the design task, the resulting methodology network, project constraints, etc. – not just individual designer comfort level. The team's design methodology should be a choice. The designer should pick the right "tools" for the job at the right time, and be as flexible as one's aptitude to choose whatever fits the current need, so that methodology decisions are deliberate, visible and team-involved, just like product design decisions. In the words of Ernzer and Birkhofer (2002), "For successful use of methods in life cycle design, it is essential to select and customize the methods carefully according to the needs of the company."

Design teams need to build up design capacity by having a wider range of methods available for use and making the best use of the full scope of design methods available in the field. Analogously to an aerodynamic designer wanting to generate multiple wing design solutions for a new airplane under development, there should be a way for designers to consider multiple methods as options for their design system that not only avoids disrupting their workflow but actually improves the overall outcome. Expanding the aerospace industry's design method "toolkit" also helps it capitalize on the latest advances in system design methodology and explore underutilized methods to tackle today's complex development needs. By bridging the gap between design research and design practice, design methods can be transferred from academia to industry and find meaningful use. As Lahonde et al. (2010) affirm, although engineering design seeks to promote methodology advancements in the enterprise, "many studies report their lack of [method] use in practice. This is not a new phenomenon and yet, no appropriate answer has been proposed to overcome this problem."

These objectives call for more of a design "culture shift" rather than a "silver bullet" (e.g. a magical method database that solves everyone's problems). In some situations, legacy methods are actually appropriate to use; only beneficial replacement of existing methods should take place. Furthermore, selecting a new method is not meant to be a permanent choice, otherwise one would be reverting to the status quo of inappropriately sticking to any particular method. The issue is not which particular methods are used, but rather the way in which methods are selected.

3.2.2 The Strategy: Attribute-Based Selection

To promote this design culture shift and achieve the dual objectives of, first, method use transparency, followed by expansion of the space of available methods, two broad strategies come to mind. The first is a grassroots approach that promotes specific design methods for wider use; the organization encourages individuals or groups to become familiar with more methods in the hopes that they advantageously deploy these methods. The concept here would be to build up organic capability in particular methods so that they receive more widespread use. The analogy would be to a craftsman or mechanic who is familiar with more tools or equipment and therefore is in a better position to select the best one for the task at hand. This could be accomplished with training and directives, mandates, or even method use quotas. Braun and Lindemann (2003) refer to these as "personal means of method transfer", such as individual coaches, trainers and consultants, but these are often limited to specific method sets. Many methods have overlapping purposes and accomplish the same functions – which methods should the organization choose to promote? What if the promoted methods do not end up fitting the design problems that appear later? One answer is to have the team learn more methods, but such targeted efforts to train the team on methods require nontrivial amounts of effort for each method advocated by the organization. There is a plethora of methods in existence – it could easily become impractical to train the design team on every method they might want to use upfront before they need to use it.

An alternative strategy involves making the general deployment and transfer of design methods more beneficial and value-added, as well as easier and less costly – without respect for specific methods upfront. The intent would be to provide more guidance than a listing or catalog of methods – plenty of these already exist (see subsection 2.2.3) – and advance to the level of a decision support system that helps transfer methods to design practitioners. As Braun and Lindemann (2003) state, "Besides the individual implementation of methods by trainers, consultants, etc. (which cannot be substituted but only supported) the transfer of methodological know-how by the means of supporting systems has to be enhanced." Braun and Lindemann (2003) would agree with the design community's need for a method selection support system that supplements, not substitutes, personalized or individual method transfer that becomes more value-added during method adaptation and application. They would proceed to argue for a support system that is formulated according to the design problem, the user, and local conditions, ideally with navigation tools and multiple views of the methodology network (not just a table of contents). Method selection decision support is most effectively employed when it is part of a larger design strategy evaluation and improvement initiative undertaken by the enterprise.

This approach first bolsters method selection, the initial step of method deployment from Braun and Lindemann (2003). This allows the more involved method adaptation and application steps to be carried out only as needed on a case-by-case basis, in contrast to the grassroots strategy that invests in full method training before the need for the method for the design situation at hand is established. The scientific community agrees that a guide to know which design method to use in a specific case should address the issue of design methods underuse and poor method transfer, which have origins in selection difficulty and lack of support mechanisms, according to Lahonde et al. (2010) and Geis et al. (2008).

Despite a variety of existing method selection guides of various forms that cover different aspects of the design process, user needs are still not met, especially for novice designers (Lahonde et al., 2010). One challenge is the sheer number of design methods that exist, as Franke and Deimel (2004, p. 1) point out: "Many methods exist, but often the user does not know which methods are appropriate to support the specific task. Furthermore, he is not able to spend time to deal with the efficiency of different methods for selecting the suitable ones." The method selection decision support must help manage the plethora of methods; Braun and Lindemann (2003) suggest leveraging the commonality among them: "It cannot be the objective to invent new methods or give new names to existing ones, but it is rather the necessity to 'illuminate the method haze' by focusing on the elementary tasks and principles that 'hide inside' methods." It must be a goal for any design method deployment guide to prevent method transfer from becoming burdensome and saturating the designer with too much information.

It can be difficult to choose the right method even if the number of alternatives can be managed. Ernzer and Birkhofer (2002) claim that with there being so many methods developed from all over world, "choosing the right methods from this mass of methods is very difficult," given the methods' differing pros and cons; dependencies on the design phase, product, and the design task at hand; and complications from the need for compatibility between disparately-developed methods.

Section 2.2 described three courses of action for assigning methods to design tasks from Braun and Lindemann (2003): assignment by superior process, attributes, and elementary working steps. This research hones in on the promise of attribute-based method selection and trades. Assignment by superior process can still result in multiple methods assigned to a task or multiple tasks to a method, which is not as helpful. In any case, a method's corresponding stage in a superior process can be considered a particular attribute of that method. The problem with assignment by elemental steps is that a design task's constituent subtasks are frequently driven by its assigned method's procedure; i.e. the method is usually not chosen based on the subtasks. Similar to software object properties, attributes appear to be a mechanism for systematic classification, search and selection of design methods. They also allow for a smooth transition to more detailed, downstream method deployment activities like adaptation and application.

3.2.3 Gap in Literature and Practice

Innovation in the area of design method selection – and design culture shifts, in general – in large-scale aerospace development is particularly difficult and relatively less prevalent in the literature and research. As alluded to in section 2.1, it is a highly-regulated industry involving complex and costly systems produced under heavily-gated development processes. The present section has already spelled out the challenging context of method deployment in organizational and budgetary terms. Aerospace product development also occurs infrequently compared to other industries, which erodes the momentum and perceived value proposition for design method space expansion. A method selection scheme must accommodate diverse business needs and program states to be effective.

The existing literature on design method selection puts forth similar approaches to tackle the problem, but there is little open consensus or conclusive results. Although there has been much research into method selection approaches that go beyond method catalogs, much of this consists of redundant variations on few themes, or even the same theme, without significant explicit convergence. In addition, method selection frameworks are proposed but often the motivation and/or validation is lacking, as little empirical data is provided to support these proposals. For example, the intent of Ernzer and Birkhofer (2002) was that their research would initiate discussions, and they insisted that their framework of company attributes and requirements for selecting methods needed validation via surveys and industry application. Such applications are hard to come by in the design of complex aerospace vehicles and systems.

3.3 Research Question and Formal Problem Statement

Sections 3.1 and 3.2 provide the rationale for posing the specific research question within the research problem area defined in subsection 2.1.1. As the circled portions of Figure 8 suggest, the scope of this research question is limited to the following:

- Conceptual design stage of aerospace product development: the earliest design phases are especially interesting to study because they have the most leverage on the development project's outcome
- System-level design: staying at a high level of abstraction has vast design leverage and encourages consideration of more generic and accessible design tasks and methods rather than less-prevalent domain-specific design issues from a focus on the design for a specific discipline/function or product subsystem
- Design methodology: see section 3.1 about the potential that lies in investing in the problem-solving techniques behind design tools used to execute design processes
- Method deployment and selection: subsections 3.2.1 and 3.2.2 describe the benefits of improving how methods are deployed and, in particular, chosen by designers

The design method selection problem is now defined using the terminology developed from section 2.2:

Given:

- A set of design artifacts and activities and the interrelationships between them (i.e. a design process)
- Criteria for trading off different design methods that can execute the same design task

Decide: which set of design methods should be assigned to perform the design activities (e.g. as illustrated in Figure 12)

Accordingly, the design method selection problem is concerned with how to assess a method (methodology) for use by a design activity (process) and make subsequent selection choices. Structured approaches to decision-making usually take the form of some kind of trade study, but what should be the specific structure of a design method selection trade study? All aspects of the problem statement are the subject of study: How should the design team reason through generating alternatives (individual methods, sets of methods, etc.)? What are the selection criteria and weightings among them? How does one score a method and establish true preference among the alternatives?

Many possible mappings could serve as the answer to any given instance of the design method selection problem: one method per activity, one method to many activities, many methods to one activity, etc. Furthermore, the problem statement could be revisited iteratively as part of a larger design strategy formulation effort – the design process is nominally fixed at the start of the problem, but if some flexibility is permitted in the process flow, there could be method assignments that change the network of design tasks and require iteration for convergence. In this sense, the design process could be formulated top-down (from "laying down" activities and passing artifacts between them) or bottom-up (based on the methodology network and the required input/output interfaces between methods).

Within the scope defined thus far, the present work responds to the problem statement by posing the following research question: "Is an attribute-based approach to selecting conceptual design methods for complex aerospace systems effective?"

3.4 Research Objectives

The overall research goal set to address the research question is to determine the effectiveness of an attribute-based approach to selecting conceptual design methods for complex aerospace products. The strategy to achieve this goal is outlined by two research objectives:

A) Realize a conceptual framework for design method selection: Drawing upon the literature, postulate a structured set of principles that employs the concept of attributes to solve the design method selection problem in the context of conceptual aerospace system development. Four sub-objectives tackle this task:

   A1) Develop a model of an aerospace system design strategy to facilitate subsequent analysis

   A2) Develop a means for discovering alternative design methods

   A3) Develop an attribute-based approach for selecting among alternative methods

   A4) Combine the alternatives discovery and method selection schemes

B) Validate the conceptual framework by means of industry stakeholder feedback: Have aerospace conceptual design practitioners provide first-order validation of the utility of results provided by the attribute-based method selection approach. The intent is to substantiate – with real industry designers in the field – the general principles and guidelines of method selection behind the framework, not to try to come up with the definitive method selection formula. The focus is not on the efficacy of particular methods or even which method attributes are most important for the selection phase of method deployment, but rather the feasibility of the overall approach.

3.5 Research Approach

Several techniques are utilized to carry out this research strategy. From a high-level perspective, the nature of the research is applied (driven by a real-world problem), exploratory (seeks insights and basic relationships to help direct subsequent research) and qualitative (instead of quantitative), and it is conducted by means of an empirical case study on a recent aerospace vehicle development project in industry. A case study allows hypotheses to be formed and tested in a real-world application, in support of Research Objective B. The subject of the case study should provide archetypal patterns and illustrations that are widely applicable.

To address Research Objective A1, an understanding of the baseline design strategy in the case study is pursued via primary research and interviews with actual members of the design organization from the development project. Lahonde et al. (2010), Gericke et al. (2015) and others provide precedent for the use of interviews to gain insights into the needs and issues facing practicing design engineers. Primary sources and documents are also a rich source of background information and preliminary models of processes and methods that can be later refined in the interviews. Public domain literature serves as a reference for broad knowledge on industry practices such as common aerospace design processes that can be used to "cross-check" case study data.

In addition, formal representations of the case study design process and methodology that capture interactions between domains are necessary to establish a common understanding of and language for the design system. They can be attained from a number of modeling techniques.

- Object-Process Methodology (OPM) fittingly distinguishes between design artifacts and design activities in the process flow and graphically portrays method-to-activity mappings.

- Matrix tools such as the Design Structure Matrix (DSM), Domain Mapping Matrix (DMM), and Multi-Domain Matrix (MDM) are useful for mapping elements to each other and sequencing tasks (Danilovic and Browning, 2007).
- Graph/network theory is useful for analyzing clustered relationships between elements.

Research Objective A2, alternative methods discovery, calls for study of the process-methodology interactions within the design strategy. The MDM matrix tool is useful for analyzing assignments between the domain of design tasks and the domain of design methods as the space of alternative methods is explored and additional methods are considered. Open literature serves as a source for alternative methods, and interviews provide key stakeholder feedback that refines the candidate method search scheme.

Selection among alternatives for Research Objective A3 revolves around the use of method attributes in the manner of Braun and Lindemann (2003), Birkhofer et al. (2002), Lahonde et al. (2010), and Ernzer and Birkhofer (2002). The literature provides many examples and listings of attributes for characterizing design methods. And again, primary research with the actual design team in the case study provides valuable real-time validation (or invalidation) of proposals to downselect to the most relevant attributes.

Finally, the prior elements integrate together into a conceptual framework for Research Objective A4 that exemplifies the principles of attribute-based design method selection in the form of a procedure.

4 Case Study: Development

This chapter presents a concept for product design method selection to guide the case study. After providing background on the subject of the case study, the work describes the application of the concept in the case study, including what and how data was collected and analyzed.

4.1 Conceptual Framework for Method Selection

Before applying an operational concept that could disrupt the status quo, it is important to keep in mind certain guidelines for evaluating and improving upon an existing design strategy, or even design system. It is critical to stay connected throughout the process with the context and needs of all stakeholders involved – not just the immediate design team, but also the relevant members of the rest of the product project or program team, functional groups or competencies, and enterprise leadership. The importance of buy-in from experienced, trusted members of the design organization is massive. It is helpful to have some process in place for the effort that spans, for example, from defining design system improvement goals to developing possible design process enhancements, from pilot programs testing concepts to operationalized deployment.

With that said, a conceptual framework for supporting the designer's method choices is proposed. A method selection decision support system would need to conform to this framework, which is presented below from the user's point of view, but note that some of the work involved is completed by the decision support system:

1. Model the baseline design strategy (design process and methodology) as a descriptive model of the flow of design tasks, the currently-used baseline methods, and the as-is assignments of methods to tasks
2. Based on the baseline design strategy model, observe how the decision support system lists the individual design tasks, each consisting of input, output, and activity

3. For each task:
    a. Provide context on how methods are to be used
    b. Discover alternative methods based on input/output compatibility when the decision support system adds unused, compatible methods as candidates to the design strategy model
    c. Observe how the decision support system characterizes the baseline and alternative methods with attributes and portrays individual methods based on the context provided
    d. Compare baseline and alternative methods by their attributes
4. Conduct method trades by either:
    a. Evaluating method attribute comparisons against requirements on task (or even method) attributes, or
    b. Aggregating individual method attributes into higher-level methodology properties and comparing with requirements on process (or even methodology) properties

Details on the development of each element of the framework follow.

4.2 Program Overview

The subject of the case study is a flight vehicle R&D program that was undertaken at The Boeing Company in response to a solicitation for vehicle concept proposals issued by an external customer. The solicitation outlined the program objectives, a high-level program schedule, and the guidelines for submitting proposals. This case study focuses on the so-called Pre-Proposal stage of the program that started from the original customer solicitation, produced the initial vehicle concept, and ended with submittal of the proposal to the customer. While subsequent program stages (e.g. late conceptual design, preliminary design) followed the initial proposal, the scope of the present research question (section 3.3) centers attention on the earliest design phases of complex aerospace development.

This R&D program is representative of a fairly archetypal aerospace development process in the early conceptual design phases. It emphasized a rapid prototyping development approach and utilized a relatively small, cross-functional team. Due to the smaller team size, individual members took on larger roles and more responsibilities. Exogenous inputs to the program's design process (details in section 4.3) provide clues on the elements of the program organization and enterprise functions that participated in the conceptual design process:

- Enterprise leadership
- Program management: program manager, chief engineer
- Business development/strategy
- Technical competencies: systems engineering, technology groups (product functional decomposition), integrated product teams (product physical decomposition)
- Finance
- Legal/Contracts

Note that technology groups and IPTs, and not just systems engineering and enterprise technical leadership, contributed to system-level design.

4.3 Model of Baseline Conceptual Design Strategy

This section begins by detailing the approach taken to model the baseline product design strategy employed in the case study. It then provides descriptions of the individual model elements before integrating these elements together to build up representations of first the process and then the methodology of the design strategy. Model improvement ideas close out the section.

4.3.1 General Modeling Approach

4.3.1.1 Modeling Purpose and Scope

The purpose of formally modeling the design strategy is to have a common understanding of the design work that transpired in the case study. Modeling also establishes a concrete language for discussing the relatively abstract and nuanced construct known as the design system (and the design strategy in particular in this case).

The point here is to come up with *a* model of the design strategy, not *the* definitive description. There are certainly many different opinions on ways to model the program's design strategy – only one instance is needed to illustrate the method selection conceptual framework, which is general enough to be applied to any version of events. Moreover, the exercise seeks to describe what did happen, not what should have happened or some kind of prescriptive or recommended design process and methodology, which are easy modes of thinking to fall into while modeling design strategies.

The scope of the effort focused on the design of the vehicle/system itself, not parallel designs that were occurring at the same time like, for example, the design of the manufacturing process to build the system. Technology infusion during the design process pertains to flight technology, not manufacturing technology. Similarly, vehicle concepts and manufacturing process concepts were not considered together under each “design alternative”. Of course, each vehicle concept has manufacturability characteristics that need to be considered.

Modeling and analysis of the design strategy includes consideration of the design process, the design methodology, and the relationship between them. The design process depends on the methodology at a certain level. A large-scale method (artifact transformation) change can change the process (artifact-activity flow) itself, as evident in Figure 11 and Figure 12. In other words, the design process is method-independent only at a high enough level of abstraction. The present modeling exercise keeps the process model high-level enough to avoid a dependence on the methodology. This way, to make the present exercise tractable, method selection is constrained by the process architecture.

When modeling the design process, some design task feedbacks (earlier tasks depending on inputs generated by later tasks) are weaker than others. To keep the model complexity manageable and the research effort tenable, only stronger dependencies and feedbacks were modeled.

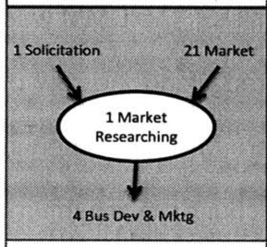
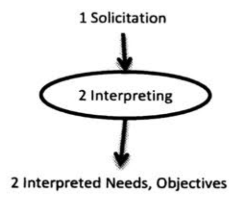
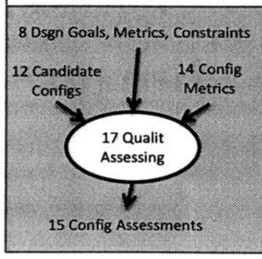
4.3.1.2 Means and Techniques for Modeling

The research techniques introduced in section 3.5 were applied in various ways during the case study to model the design strategy. Primary source material references and personal interviews with design team members provided the bulk of the raw data used to build up the design strategy model. See the Appendix for details on the questions used during the interview process. Design tasks (the usual representation of the design process in written and verbal descriptions that are found) were decomposed into activities and input and output artifacts. When this was less straightforward to do, the lead group that performed a task was determined, which helped reveal what the design artifacts were. The focus of the process modeling was not about the precise timing of activities, but rather the basic dependencies between activities. The resulting model – which converged to its final form only after several interview iterations – is of realistic complexity with a sizeable number of elements.

Many of the methods employed by the design team were relatively unstructured and difficult to define or label by the user – the designer's emphasis was on completing a task, not executing a method per se. Characterization of these unstructured methods was achieved by eliciting the elemental steps performed by the user during the task and then correlating this procedure with those of unstructured methods documented in the literature to see which one it most closely aligned with. Researching informal methods, in the terms of Schmidt (2015), helped case study interviewees define their unstructured techniques.

Table 1 shows an example of how the definitions of the modeled design tasks and their methods were refined during the modeling effort. A task's associated lead group shed light on the input and output artifacts involved in the task, and the interviewees' descriptions of how the task was solved helped identify the design method used.

Table 1: Excerpt of design tasks with associated lead groups and baseline methods

Activity & Artifacts	Task	Lead Group	Method(s)	Method Description(s)
 1 Solicitation 21 Market 1 Market Researching 4 Bus Dev & Mktg	1 Research underlying market, competition, and business case	Business team	19 (unstructured) Literature Searching	• Use enterprise processes, but with local interpretations • Variable sources
 1 Solicitation 2 Interpreting 2 Interpreted Needs, Objectives	2 Interpret customer solicitation	Enterprise competencies-technical	14 (unstructured) Subject Matter Expertise	• "Heavily empirical" • Reliance on their domain experience
 8 Dsgn Goals, Metrics, Constraints 12 Candidate Configs 14 Config Metrics 17 Qualit Assessing 15 Config Assessments	17 Qualitatively assess candidate concepts	Systems Engineering	7 Decision matrix	1. List criteria with weights 2. Score each concept against each criterion 3. Calculate weighted average scores for each concept
			14 (unstructured) Subject Matter Expertise	• Subjective evaluations • Intuition about plausibility

When using OPM diagrams (OPDs, or Object-Process Diagrams), it is helpful to identify the exogenous input and output design artifacts that were either only consumed or only produced by the design process, thus establishing the boundary of the process. When using matrix tools to represent sequential tasks, one must differentiate between a "consumption" interaction (i.e. A is consumed by B) and a "production" interaction (e.g. A produces B), either through different interaction markings, or separate consumption and production matrices altogether. In terms of graph theory, directed graphs were used where nodes represented both design activities and design artifacts, and edges represented the consumption and production of artifacts by activities.

As depicted in Figure 14, an iterative sequence that integrated the various model representations together was used to refine the model morphology. First, OPDs were used to engage with interviewees and come up with the basic design process dependencies; the morphology of the process was in a preliminary, rather jumbled state. Then these dependencies were entered into a DMM with both consumption (or "destroying", d) and production (or "creating", c') interactions between design artifacts and activities. The DMM was recast into a directional MDM of artifacts and activities where inputs entered along the row of each element and outputs left along the column of each element. This square MDM was then in a format that could be entered into a graph analysis tool like Pajek that could run minimum-energy clustering algorithms (e.g. Kamada-Kawai) to inform how the OPD morphology can be "cleaned up" for the next iteration.

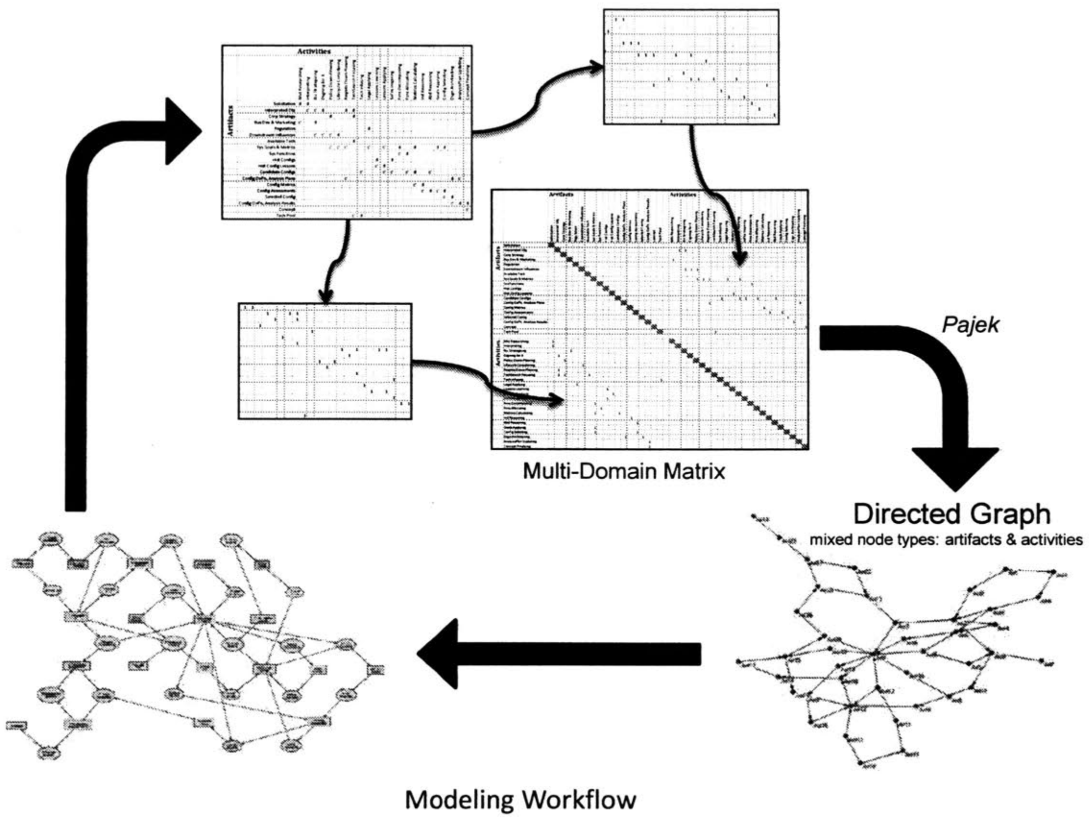


Figure 14: Design process modeling workflow

4.3.2 Elements of the Design Strategy Model

4.3.2.1 Design Activities

The label for a design activity centers on a verb; nouns may be used to clarify or specialize the verb, or to suggest the primary object of the verb (i.e. a design artifact). Labels and descriptions for the design activities present in the baseline design process are provided below:

1. (Act1) Market Researching: gathering, aggregating and interpreting market and competitive research data
2. (Act2) Solicitation Interpreting: gathering solicitation data, identifying key information, translating terminology, "decoding" and clarifying, elaborating on details (e.g.

priorities/weightings among objectives), determining unsaid assumptions and inferring the unwritten from the customer solicitation

3. (Act3) Business Strategizing: developing market-driven influences on design, including payload specifics
4. (Act4) "Design for X" Requirements Developing: considering design for various lifecycle properties (aka "-ilities"), e.g. Design for Manufacturability, Supportability, Reliability, Quality, Flexibility, Usability, Maintainability
5. (Act5) Policy Down-Flowing: applying enterprise strategy and "gatekeeping" to the design
6. (Act6) Lifecycle Considering: applying downstream design, implementation and operational influences on upfront design goals and/or constraints on concept generation
7. (Act7) Requirements Down-Flowing: applying customer objectives and requirements
8. (Act8) Flight Technology Search Focusing: determining technology availability, establishing maturity thresholds, screening technology possibilities
9. (Act9) Flight Technology Infusing: assessing technology insertion options, inserting technologies into design concepts
10. (Act10) Legal/Compliance Applying: administering regulations and standards on design
11. (Act16) Metrics Calculating: evaluating alternative concepts using various criteria (soft qualitative or hard quantitative), employing deductive reasoning (rules combined with specific cases producing facts)
12. (Act17) Qualitative Assessing: evaluating alternative concepts for feasibility, high level pros/cons, ways to improve them, employing inductive reasoning (specific cases combined with facts producing rules)
13. (Act18) Configurations Improving: updating the pool of alternative concepts, employing abductive reasoning (facts combined with rules producing specific cases)
14. (Act20) Configuration Selecting: screening alternative concepts

15. (Act21) Design Elaborating/Architecting: providing initial requirements, design definition and analysis for the down-selected solution concept
16. (Act22) Extended Configuration Definition and Analysis Plans Updating: revising prior plans based on latest information as analysis results come in
17. (Act23) Proposal Concept Finalizing: verifying that proposal submission requirements are met by design information available so far, including risk analysis
18. (Act24) Historical Configuration Lessons Learning & Applying: reviewing the history of configurations in this product domain; identifying prior shortcomings and historically-informed design considerations and performance metrics; utilizing lessons learned from history to inspire
19. (Act25) Functions Deploying (Decomposing & Allocating): generating alternative concepts, employing classic systems engineering functional decomposition, allocation and deployment whereby, for each functional attribute, an element of physical form is assigned, then elements are integrated into complete configuration concepts

4.3.2.2 Design Artifacts

A design artifact is described with a noun, the object of a design activity. Labels and descriptions for the design artifacts present in the baseline design process are provided below:

1. (Art1) Solicitation: program and design objectives, key metrics, high-level schedule (and indicators of level of urgency), cost constraints, and guidelines for submitting and evaluating proposals, from the customer; could be revised, or stated needs otherwise changed
2. (Art2) Interpreted Customer Needs & Objectives: rephrased program objectives with finer distinctions between minimum thresholds and stretch goals, specific conditions for assessing metrics, detailed design considerations; also interpretations of intentions for narrow focus on minimum requirements vs. longer-term applications

3. (Art3) Corporate Strategy: company policies and procedures; technology strategy; financial and resource allocations; longstanding company best practices, traditions and design standards; considerations about broader company interests like safety, reputation and risk tolerance; downstream strategies for core capabilities/competencies, sourcing, manufacturing, sales
4. (Art4) Business Development & Marketing: market needs, segmentation and sizing; competitive analysis; product marketing, pricing, placement/channels, promotion/advertising
5. (Art5) Regulation: airworthiness/certification; contractual obligations; external design standards (e.g. MIL-STD)
6. (Art6) Downstream Influences: issues pertaining to post-design work, e.g. manufacturability, supply chains, ground/flight test, demonstration/validation, maintainability, reliability, adaptability, sales and deployment, interoperability
7. (Art7) Available Flight Technologies: various flight technologies deemed mature enough for product insertion
8. (Art8) System Design Goals, Metrics & Constraints: formal statement of design objectives; quantitative design metrics and qualitative criteria; relative weightings of metrics; concept of operations; technology selection/insertion criteria; risk criteria; constraints on product complexity, manufacturing, sourcing/teaming, time and resources, interoperability/interfacing with legacy elements
9. (Art10) Historical Configurations: past/previously conceived exploratory concepts; past experience with similar projects, configurations
10. (Art12) Candidate Configurations: sketches of design solution alternatives with technology insertion options

11. (Art13) Extended Configuration Definition and Performance Analysis Plans: solicitation-specified definition and analysis plans; initial layouts, topologies, design parameters, weight estimates; constraints on initial requirements and specifications
12. (Art14) Configuration Metrics: calculated metric values for the configurations
13. (Art15) Qualitative Configuration Design Assessments, Tradeoffs: pros/cons and feasibility of the configurations; inductively-reasoned design recommendations
14. (Art16) Selected Configuration: depictions of design solution concept with selected technologies
15. (Art17) Extended Configuration Definition and Performance Analysis of Selected Configuration: initial requirements and specifications for the solution concept; performance characteristics
16. (Art18) Solicitation Concept: elaborated design solution along with supporting analysis, including risk assessments
17. (Art19) Flight Technology Pool: the subset of technologies deemed relevant for product insertion in this situation
18. (Art20) Enterprise Competencies & Capabilities: design team and skills, manufacturing systems and technologies, infrastructure, assembly, test & verification team and capacity, supplier management, marketing, partners, reliability & maintainability, operations, training, logistics
19. (Art21) Market Data: primary and secondary market research, in the general domain but beyond the immediate scope of the customer solicitation

4.3.2.3 Design Methods

Labels and descriptions for the design methods present in the baseline design methodology are provided below, where unstructured (u) methods are further labeled to denote that they were "induced" based on descriptions elicited from the interviews, as explained in subsection 4.3.1:

1. (Mtd3) (u) Committee deliberation: the "right" group of people (e.g. technologists, designers, analysts) performing broad, unstructured trades using inquiry- or Socratic-like learning to elicit information and make decisions; action items frequently taken afterwards; may be similar to what is known as the "gallery method" (Schmidt, 2015)
2. (Mtd5) (u) Brainstorming: sketching, drafting, "PowerPoint-ing"; according to Pahl et al. (2007, p. 190): "rough sketches or rough scale-drawings of possible layouts, forms, space requirements, compatibility, etc."; may be similar to what is known as the "Searching for Visual Inconsistencies" method (Jones, 1992)
3. (Mtd6) Morphological matrix: functional decomposition with various options for each function; would use all input data such as historical lessons when combining solution threads into concepts; was probably not used with "systematic variation" method (Schmidt, 2015; Pahl et al., 2007)
4. (Mtd7) Modified decision matrix: aka "weighted trade matrix of alternatives"; focus was on each alternative's score, not the final ranked list of alternatives; part of a family of methods that performs trade studies
5. (Mtd13) Point-Based Design (PBD): in contrast to set-based design (SBD)
6. (Mtd14) (u) Subject Matter Expert (SME) Consultation: intuitive use of heavily empirical domain experience; many mediums possible (e.g. phone, in-person); internal and external personnel, including self (designer is the SME); individual or group; enablers include operational realism, doses of plausibility and utility, experience to make right assumptions; given the broad definition, this method is used for virtually every task, but it is called out explicitly if it was especially leaned upon; may be similar to what is known as the "Delphi method" (Pahl et al., 2007)

7. (Mtd15) (u) Rapid Custom Domain-Specific Analysis (RCDSA): calculations specific to a domain or discipline (e.g. aerodynamics, propulsion, stability & control) formulated by SME opinion; first principles science and textbook equations using approximations, estimates and assumptions based on experience; first-order (lower-order) accuracy; often implemented with simple spreadsheets; according to Pahl et al. (2007, p. 190): "rough calculations based on simplified assumptions"
8. (Mtd16) (u) Systematic Combination: select a solution for each subfunction that meets requirements and constraints, then combine compatible subfunctions; reflections on brainstorming sketches; according to Pahl et al. (2007): "concentrate on promising combinations and establish why these should be preferred above the rest"
9. (Mtd17) Risk analysis: risk matrix of likelihoods and consequences; includes priorities and mitigation plans
10. (Mtd18) Readiness levels: readiness of technologies, interfaces and systems; used as measurements of risk and criteria during decision-making under risk
11. (Mtd19) (u) Literature Searching: local interpretations of literature; external and internal sources like corporate policies, PDP guidance, market research processes, company imperatives, design standards/checklists, make/buy policies; aka "Information Gathering" (Pahl et al., 2007); Jones (1992) offers formal guidelines for the method
12. (Mtd20) (u) Needs-Usage-Standards Method: to develop aircraft system requirements, iteratively balance customer objectives, operational usage, airworthiness design standards, and vehicle capabilities; tailoring for prototype vs. production
13. (Mtd21) Sizing and Performance Computer Program: solves for coupled aerodynamics-performance-propulsion-weights problems
14. (Mtd22) Computer Aided Design (CAD): automated drafting
15. (Mtd23) (u) Technology osmosis: technologists sharing ideas when the program becomes visible, with both push and pull

16. (Mtd24) Technology roadmaps: artifacts from the technology development process feeding the PDP
17. (Mtd25) (u) System Integration Priorities method: integrate subsystems and functions into the system in a particular prioritized sequence derived from SME experience
18. (Mtd26) (u) Design catalog: empirical database, e.g. past manufacturing processes, technologies, approaches; reuse existing solutions; assemble jigsaw puzzle of known, proven approaches; determine challenges, unknowns; assume there will be innovation; assess integration risks; phenomenology tests
19. (Mtd27) Document-Based Design: in contrast to model-based design (MBD); an information exchange method; documents used to link standalone models for common system representation
20. (Mtd28) Independent Review: reviewers are external to the immediate design team; interestingly, domain experts could be the most cautious about their particular domains because they know the challenges and horror stories involved

4.3.3 Integrated Model Progression

4.3.3.1 Design Process

The design activities and artifacts described in subsection 4.3.2 make up the model of the baseline conceptual design process in the case study. A variety of representations of the model provide multiple viewpoints of the system design process employed in the case study. Figure 15 depicts the directional DMM between design activities and artifacts where, for example, design activity Act2 "Solicitation Interpreting" takes design artifact Art1 "Solicitation" as an input (or "destroys" Art1) and outputs (or "creates") design artifact Art2 "Interpreted Customer Needs & Objectives". Design artifacts that are only "destroyed" are exogenous inputs, e.g. Art20 "Enterprise Competencies & Capabilities".

c' = creating
d = destroying

Artifacts	Act1 Mkt Researching	Act2 Interpreting	Act3 Biz Strategizing	Act4 Dsgning for X	Act5 Policy Down-Flowing	Act6 Lifecycle Considering	Act7 Reqmts Down-Flowing	Act8 TechSearch Focusing	Act9 Tech Infusing	Act10 Legal Applying	Act16 Metrics Calculating	Act17 Qualit Assessing	Act18 Config Improving	Act20 Config Selecting	Act21 Dsgn Architecting	Act22 AnalysisPlan Updating	Act23 Concept Finalizing	Act24 Lessons Learning, Applying	Act25 Fcns Deploying
Art1 Solicitation	d	d																	
Art2 Interpreted Needs, Obj		c'	c'	d			d	d											
Art3 Corp Strategy					d			d											
Art4 Bus Dev & Marketing	c'		d																
Art5 Regulation										d									
Art6 Downstream Influences			c'	c'	c'	d													
Art7 Available Tech								d											
Art8 Dsgn Goals, Metrics, Constraints					c'	c'	c'			c'	d	d	d	d				c'	d
Art10 Hist Configs																		d	
Art12 Candidate Configs									c'		d	d	c'	d				c'	c'
Art13 Config Def'n, Analysis Plans							c'								d	c'			
Art14 Config Metrics											c'	d							
Art15 Config Assessments												c'	d	d					
Art16 Selected Config														c'	d				
Art17 Config Def'n, Analysis Results															c'	d	d		
Art18 Concept																	c'		
Art19 Tech Pool								c'	d										
Art20 Competencies				d															

Figure 15: Domain Mapping Matrix of baseline design process model

The Figure 15 DMM can be turned into a square MDM matrix, which enables entry into a graph analysis program and generation of the clustered directed graph of mixed node types shown in Figure 16. This analyzed graph view of the design process helped inform the final morphology of the OPD form in Figure 17.

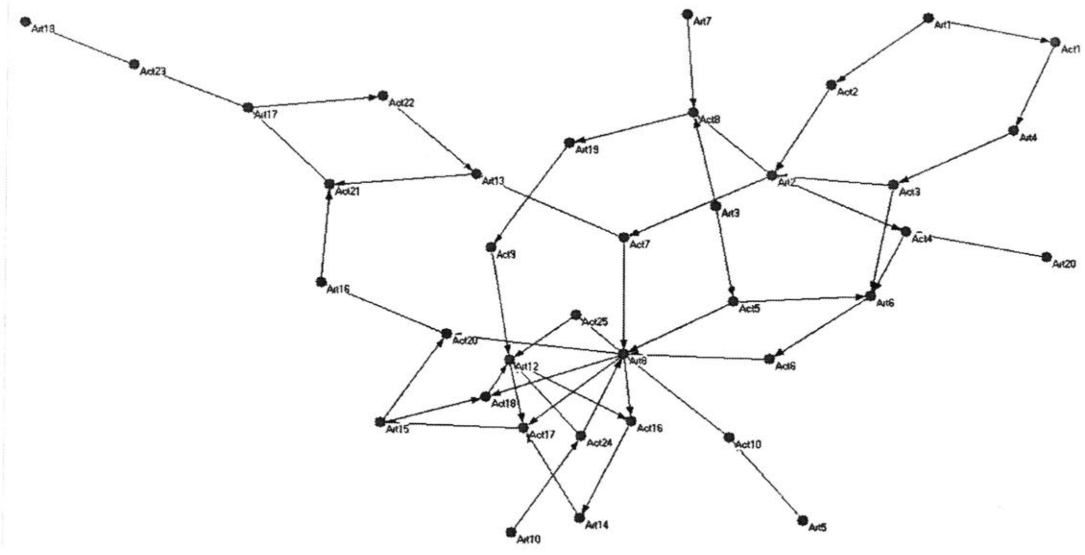


Figure 16: Directed graph of baseline design process model

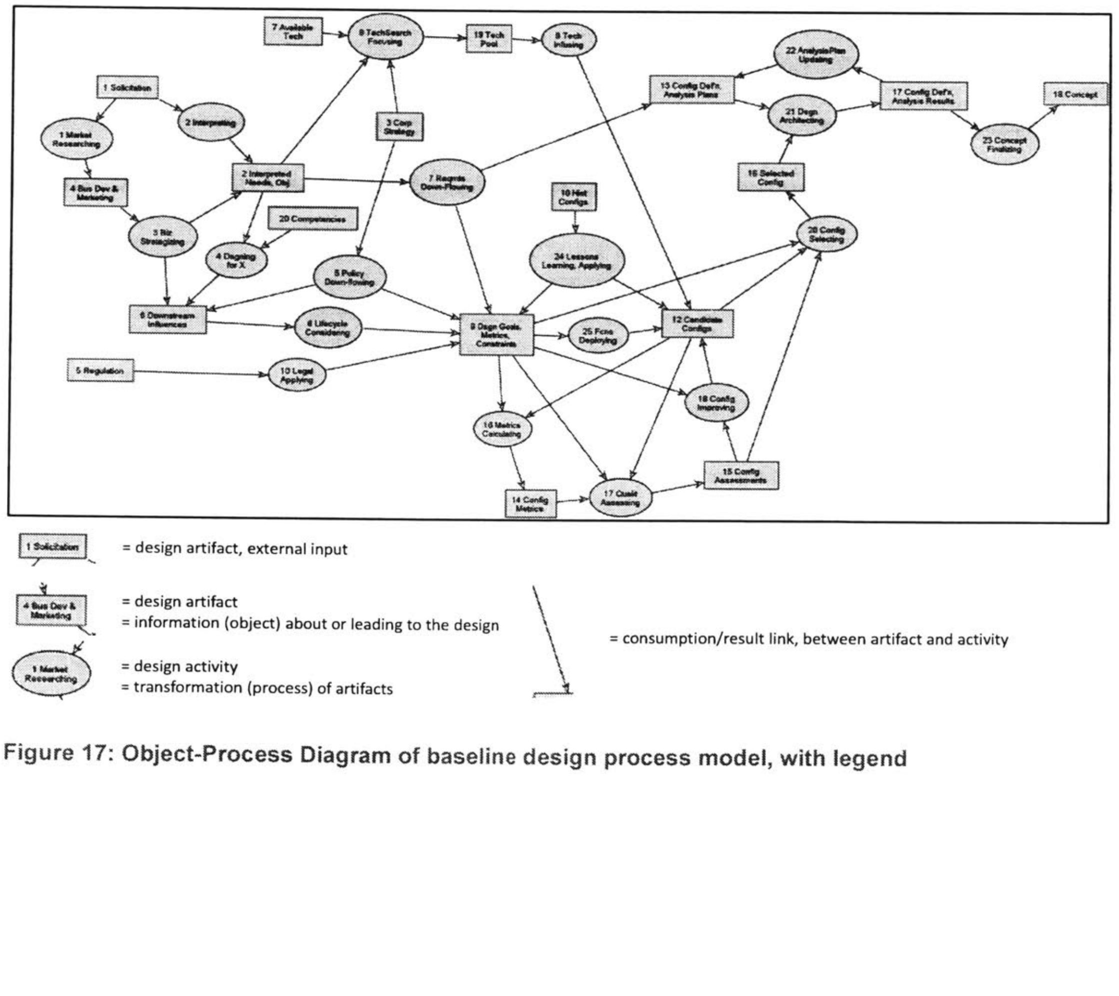


Figure 17: Object-Process Diagram of baseline design process model, with legend

As described in subsection 2.2.3, a set of design tasks can be induced from the design process model, based on the basic form:

Input Artifact(s) → Activity → Output Artifact(s)

Although a list or even a network of tasks is a more familiar representation for design processes, the labels or short descriptors assigned to design tasks often focus on either the input or output and miss the connective details of the representations in Figure 15, Figure 16 and Figure 17 that follow the activity-artifact pattern.

Below is a list of the design tasks modeled in this case study (the numbers below are identifiers that match the corresponding design activities; they do not denote sequence):

1. Task 1 Research underlying market, competition, and business case
2. Task 2 Interpret customer solicitation
3. Task 3 Develop business strategy for product lifecycle
4. Task 4 Determine program lifecycle needs and enterprise competencies
5. Task 5 Apply corporate strategy and policy to product lifecycle
6. Task 6 Develop product lifecycle design goals and constraints
7. Task 7 Translate customer needs and program objectives
8. Task 10 Apply regulations, standards and contractual statutes
9. Task 25 Generate candidate concepts
10. Task 24 Apply historical lessons learned
11. Task 8 Select flight technologies for infusion into product
12. Task 9 Infuse technologies into candidate concepts
13. Task 16 Calculate performance metrics for candidate concepts
14. Task 17 Qualitatively assess candidate concepts
15. Task 18 Improve candidate concepts
16. Task 20 Select concept configuration

17. Task 21 Perform design elaboration/architecting and analysis on selected concept configuration
18. Task 22 Update configuration definition and analysis plans
19. Task 23 Finalize proposal configuration definition and analysis

To further validate the case study design process model embodied by these matrix and graphical representations, this design task list was reviewed with the designers and compared with common aerospace conceptual design process models from the literature (e.g. Raymer (1992)).

4.3.3.2 *Design Methodology and Complete Strategy*

Subsection 4.3.2 also lays out the baseline design methods assigned to solve the modeled design tasks. The DMM between design methods and activities is given by Figure 18, where the "instruments" of each activity are denoted with the "I" marks. Note how several activities required more than one method, and how a method was often deployed for multiple activities.

Methods \ Activities	Act1 Mkt Researching	Act2 Interpreting	Act3 Biz Strategizing	Act4 Dsgning for X	Act5 Policy Down-Flowing	Act6 Lifecycle Considering	Act7 Reqmts Down-Flowing	Act8 TechSearch Focusing	Act9 Tech Infusing	Act10 Legal Applying	Act16 Metrics Calculating	Act17 Qualit Assessing	Act18 Config Improving	Act20 Config Selecting	Act21 Dsgn Architecting	Act22 AnalysisPlan Updating	Act23 Concept Finalizing	Act24 Lessons Learning, Applying	Act25 Fcns Deploying
Mtd3 (u) Committee						I							I	I					
Mtd5 (u) Brainstorming									I										I
Mtd6 Morphological matrix																		I	I
Mtd7 Modified decision matrix												I		I					
Mtd13 Point-Based Design														I					
Mtd14 (u) SME Consultation		I			I	I	I			I		I			I	I		I	
Mtd15 (u) RCDSA											I				I				
Mtd16 (u) Systematic Combination													I						
Mtd17 Risk analysis																	I		
Mtd18 Readiness levels																	I		
Mtd19 (u) Lit search	I		I		I													I	
Mtd20 (u) Needs-Usage-Standards															I				
Mtd21 Sizing/Perf pgm															I				
Mtd22 CAD															I				
Mtd23 (u) Tech osmosis								I											
Mtd24 Tech roadmaps								I											
Mtd25 (u) Integration priorities															I				
Mtd26 (u) Design catalog				I															
Mtd27 Document-based design							I				I				I	I			
Mtd28 Independent Review															I				

Figure 18: Domain Mapping Matrix of model of baseline design methods mapping to design activities

The DMM in Figure 18 portrays the interface between the process and methodology of the design strategy. A graphical representation of the design methodology proper could be achieved by replacing each activity in Figure 17 with the methods that enable it using the mapping information in Figure 18. However, due to certain activities requiring multiple methods, complete connective insight into the methodology to the same extent as the notional illustration in Figure 11 would require additional information about which design artifacts are feeding each particular method. Instead, the DMMs in Figure 15 and Figure 18 are used to formulate the Figure 19 three-domain MDM of the modeled interactions between and within the design artifacts, activities and methods, i.e. the complete design strategy. Note that the artifact-activity DMM and the activity-method DMM are the user-specified interaction matrices from which the other interactions were induced; the calculations behind the induced matrices in Figure 19 and their interpretation will be discussed in subsection 4.4.2 in the context of discovering alternative methods, for which this matrix analysis is well-suited.

Similarly, the OPD of the design process model in Figure 17 can be augmented to represent the entire design strategy, as shown in Figure 20. Four clusters of design tasks are evident:

- Problem Definition
- Technology Infusion
- Solution Exploration
- Solution Convergence

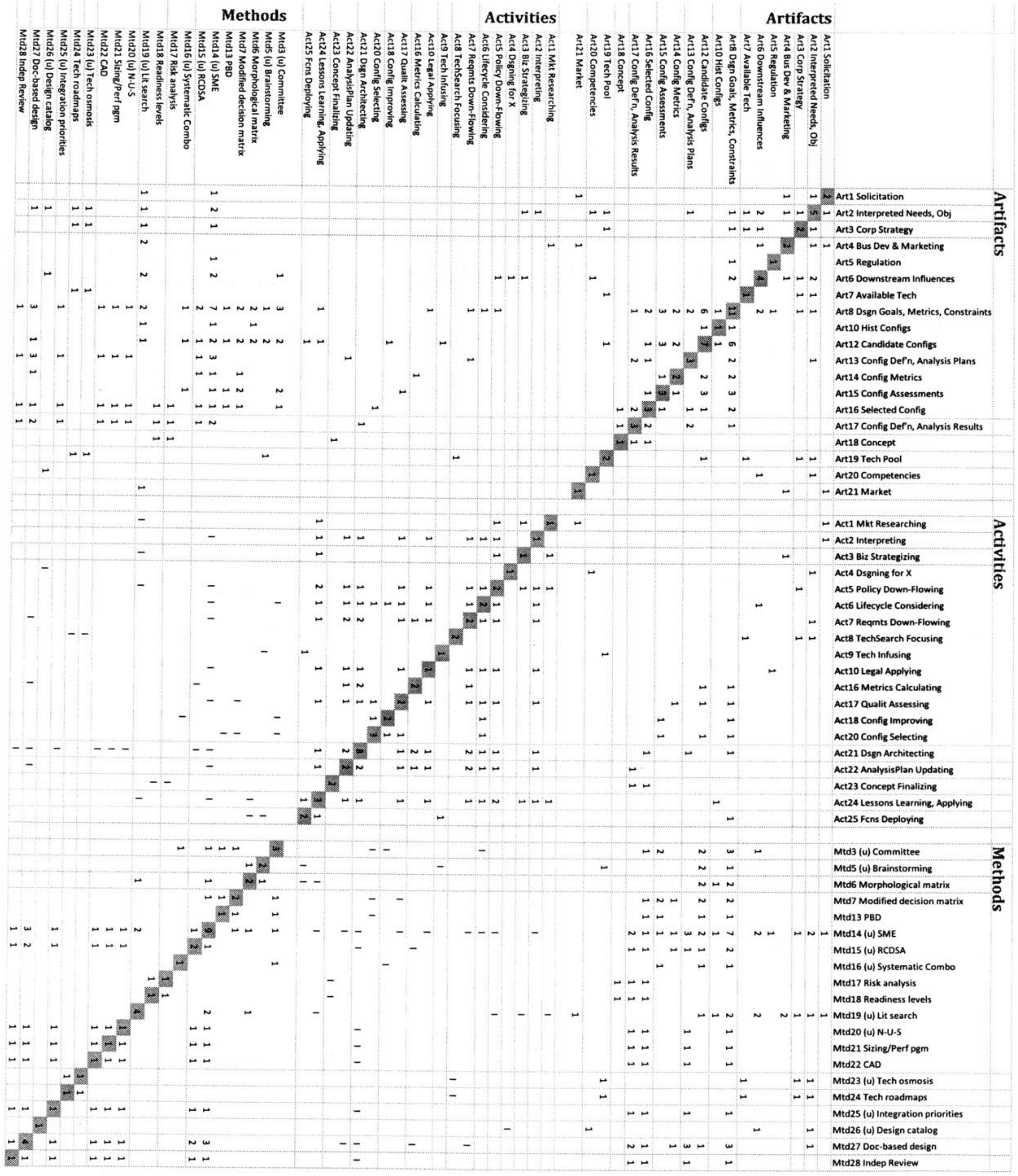
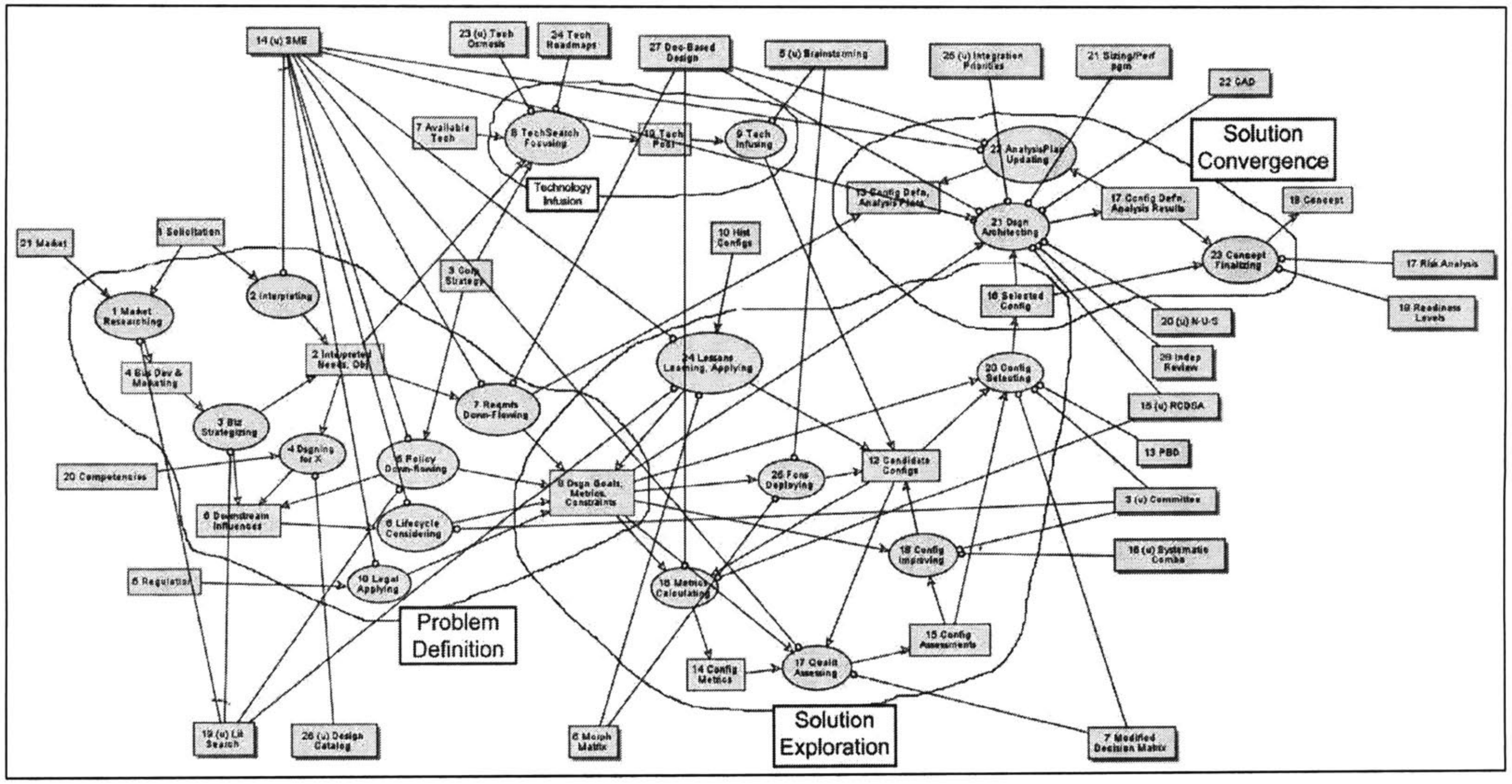


Figure 19: Multi-Domain Matrix of baseline design strategy model



= design artifact, external input

= design artifact
= information (object) about or leading to the design

= design activity
= transformation (process) of artifacts

= design method/tool
= enabler of or approach to executing activity

= consumption/result link, between artifact and activity

= instrument link, between activity and method

Figure 20: Object-Process Diagram of baseline design strategy model, with legend and annotations

4.3.4 Model Development Suggestions

A couple of areas for adjusting the design strategy model have already been voiced. First, there may be a missing feedback loop from the Solution Exploration cluster of design tasks back to the Problem Definition cluster of tasks – the results of analyzing an early set of alternative solution concepts are frequently used to revise the assumptions used to formulate design objectives in order to improve feasibility. This could be modeled by feeding Art15 "Qualitative Configuration Design Assessments" back to existing activities like Act7 "Requirements Down-Flowing" and/or Act6 "Lifecycle Considering", or to a new Revising activity that updates Art8 "System Design Goals, Metrics & Constraints".

Another remark from the interviewees essentially deemed Mtd27 "Document-Based Design" (DBD) not as a proper design method in this context but more of an aspect of the design tool that implements a method. If DBD was considered an actual method, it would be solving a task concerning information exchange – but information exchange is occurring with every task, almost as "overhead". Therefore, DBD and its functions should be removed from the immediate scope of design method selection and saved for discussion on design toolset formulation. These suggestions were saved for future reference.

4.4 Discovering Method Alternatives

The identification of alternative product design methods that are compatible with a given design task is explained in this section. After reviewing the issue of an initial pool of methods from which to choose from, the development of the complete multi-domain matrix form of the design strategy model is shown. A subset of the baseline design strategy is then used to illustrate a general procedure for discovering alternative methods and the application of this procedure on said subset, followed by descriptions of the particular alternative methods identified.

4.4.1 The Space of Available Design Methods

Subsection 3.2.1 introduced the notion of the design method space as illustrated in Figure 13. Out of all existing methods, only a relative few are chosen for deployment in any given development project design system. The rest that are not deployed comprise the set of "non-deployed" or unused methods for any given design situation, including:

- Considered but rejected methods
- Not considered but known methods
- Unknown methods

As subsection 3.2.1 argued, a key part of the mission to improve design method deployment is to move as many unused methods as possible from the second and third categories to the first category – in other words, to help designers be aware of and assess more design task-solving options (i.e. design methods) for use than they currently do. The first category – considered but not selected methods – combined with the set of selected/deployed methods together constitute the 'Considered' group of methods in Figure 13, the space of alternative methods that contains all candidates that are considered for deployment to solve design tasks.

There are many sources of alternative design methods, including the design catalogs referred to in subsection 2.2.3. Lahonde et al. (2010) provide a list of sources of method information utilized by their interviewed designers. Of course, the sheer volume of methods available in academia and industry can easily saturate an enterprise capacity's to reasonably organize and consider them. To narrow down the field, Jones (1992, p. 89) offers five criteria for including a method for consideration in his method catalog:

- Effectiveness
- Relevance
- Convenience
- Familiarity
- Criticism

As alluded to in subsection 2.2.4, Ernzer and Birkhofer (2002) suggest a higher-level version of attribute-based selection of individual methods (section 4.5) to determine how many and which methods to make known to the overall company, since company needs go beyond generic criteria from the literature. They propose a mix of product-related (e.g. complexity, scale, dominant lifecycle phase), company-related (e.g. design competency, degree of innovation, strategic aims, resources), and environment-related (e.g. market, laws/standards) attributes for characterizing the influences on the enterprise's needs. These enterprise attributes are then correlated with the attributes of all known methods to filter them down to a set of company-relevant methods.

Ultimately, it is up to the enterprise to decide what is best for them. For the purposes of this case study, it is assumed that there exists some pool of available methods that can be analyzed for compatibility with particular design tasks.

4.4.2 Design Strategy Matrix Analysis

The MDM of the design process and methodology relating the artifact, activity and method domains was presented in Figure 19. The origins of the component DSMs and DMMs that make up the design strategy model MDM are depicted in Figure 21. The design process (activity-artifact) bidirectional DMM from Figure 15 based on designer interviews was split into two unidirectional DMMs (as illustrated in Figure 14) and entered into the MDM as matrices B and D. The static DMM assigning baseline methods to activities was also entered into the MDM; it was entered twice (F and H=F^T), with one entry simply being the transpose of the other.

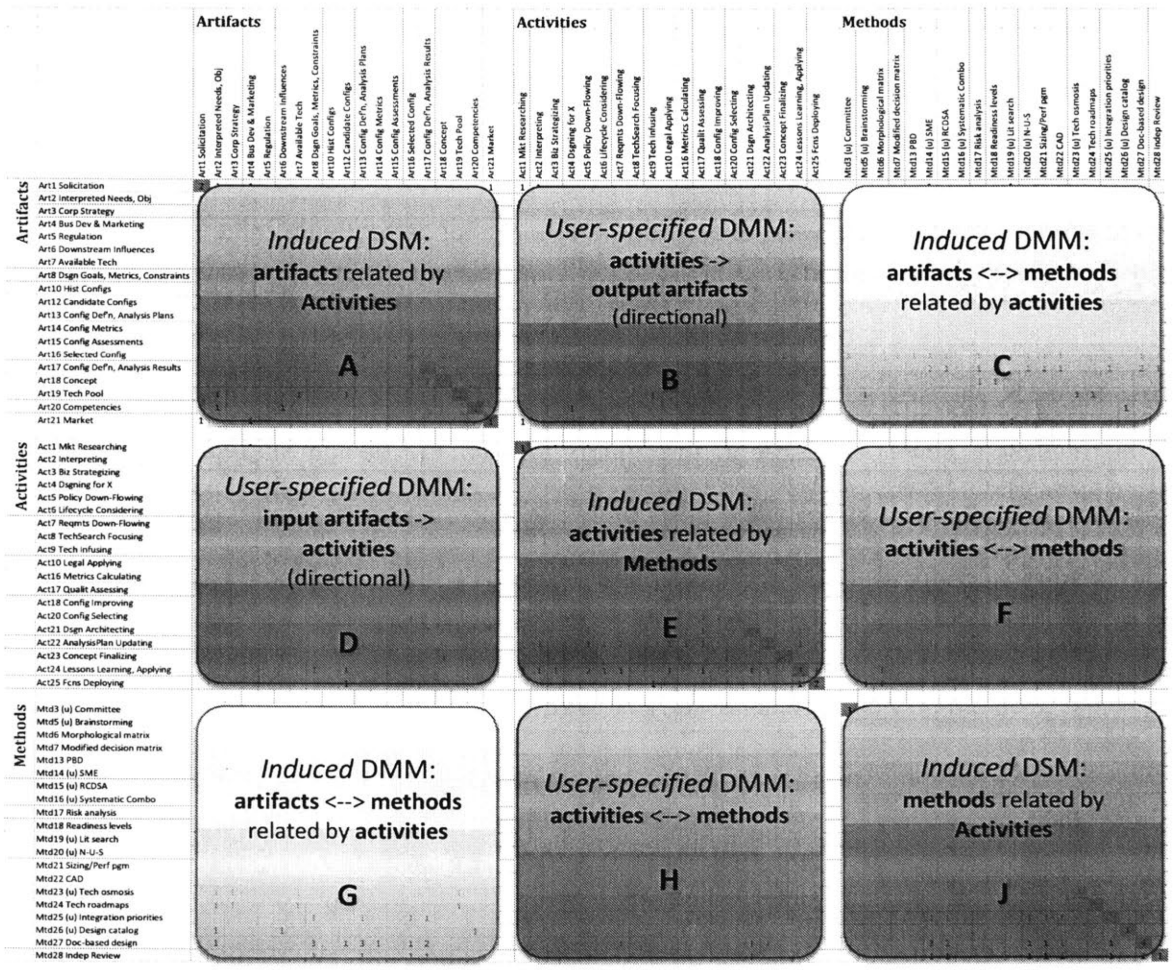

Artifacts
Activities
Methods
Induced DSM: artifacts related by Activities
A
User-specified DMM: activities -> output artifacts (directional)
B
Induced DMM: artifacts <--> methods related by activities
C
User-specified DMM: input artifacts -> activities (directional)
D
Induced DSM: activities related by Methods
E
User-specified DMM: activities <--> methods
F
Induced DMM: artifacts <--> methods related by activities
G
User-specified DMM: activities <--> methods
H
Induced DSM: methods related by Activities
J

Figure 21: Abstract summary of multi-domain matrix of baseline design strategy model

Once these user-specified DMMs were available, the following induced matrix calculations were possible:

- DSM of artifacts related to each other by the activities they share:

  $A = (B+D^T)^* (B+D^T)^T$,

  where $(B+D^T)$ essentially reconstructs the bidirectional process DMM in Figure 15

- DSM of activities related to each other by the methods they share:

  $E = F * H$

- DSM of methods related to each other by the activities they share:

  $J = H * F$

- DMM of artifacts related to methods by the activities they share:

  $C = (B+D^T) * F$

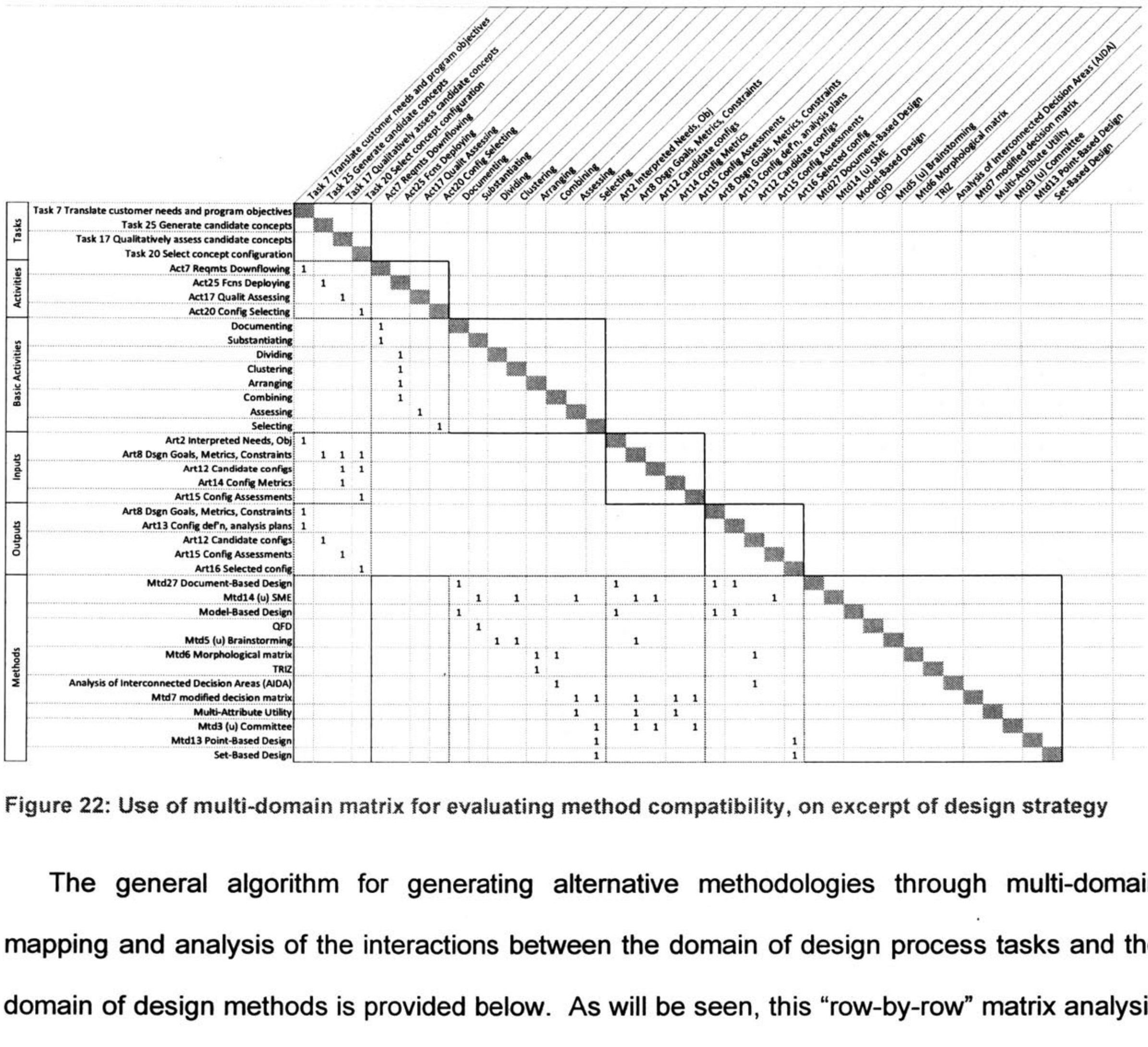
  $G = H * (B+D^T)^T = C^T$

It is important to keep in mind what exactly an "interaction" or dependency means for each DMM and DSM of the MDM.

4.4.3 Generating Alternative Methodologies via Multi-Domain Matrix

The analytical power available in this matrix form of the design strategy model will be demonstrated in detail in the following parts of this section. The basic idea is that the available methods from subsection 4.4.1 (both aerospace-specific design methods and more generic methods that are applicable to aerospace design) will be characterized in terms of the underlying problem-solving functions they serve. These functions (or design tasks) are composed of design artifacts and activities – thus, each method (baseline and alternative) will be mapped to the artifacts and activities they serve. These methods will be folded into the MDM representation, first with currently-used baseline methods, followed by expansion of the method domain with additional and potentially net-beneficial methods, in order to generate alternative assignments of methods to design process activities.

Figure 22 shows an excerpt of how the MDM can be expanded to support alternative method choices. The four design tasks are comprised of four design activities and a number of input and output artifacts from the MDM. The design activities are decomposed into basic activities after the style of Braun and Lindemann (2003) and Franke and Deimel (2004), as discussed in subsection 2.2.4. Then methods are mapped to these basic activities and input/output artifacts to determine which methods are compatible with each design task in the end.



		Task 7 Translate customer needs and program objectives	Task 25 Generate candidate concepts	Task 17 Qualitatively assess candidate concepts	Task 20 Select concept configuration	Act7 Reqmts Downflowing	Act25 Fcns Deploying	Act17 Qualit Assessing	Act20 Config Selecting	Documenting	Substantiating	Dividing	Clustering	Arranging	Combining	Assessing	Selecting	Art2 Interpreted Needs, Obj	Art8 Dsgn Goals, Metrics, Constraints	Art12 Candidate configs	Art14 Config Metrics	Art15 Config Assessments	Art8 Dsgn Goals, Metrics, Constraints	Art13 Config def'n, analysis plans	Art12 Candidate configs	Art15 Config Assessments	Art16 Selected config	Mtd27 Document-Based Design	Mtd14 (u) SME	Model-Based Design	QFD	Mtd5 (u) Brainstorming	Mtd6 Morphological matrix	TRIZ	Analysis of Interconnected Decision Areas (AIDA)	Mtd7 modified decision matrix	Multi-Attribute Utility	Mtd3 (u) Committee	Mtd13 Point-Based Design	Set-Based Design
Tasks	Task 7 Translate customer needs and program objectives |▨|																																						
	Task 25 Generate candidate concepts		▨																																						
	Task 17 Qualitatively assess candidate concepts			▨																																					
	Task 20 Select concept configuration				▨																																				
Activities	Act7 Reqmts Downflowing	1				▨																																			
	Act25 Fcns Deploying		1			▨																																				
	Act17 Qualit Assessing			1			▨																																			
	Act20 Config Selecting				1			▨																																		
Basic Activities	Documenting					1			▨																																	
	Substantiating					1				▨																																
	Dividing						1				▨																															
	Clustering						1					▨																														
	Arranging						1						▨																													
	Combining						1							▨																												
	Assessing							1								▨																											
	Selecting								1								▨																										
Inputs	Art2 Interpreted Needs, Obj	1																▨																									
	Art8 Dsgn Goals, Metrics, Constraints		1	1	1																▨																								
	Art12 Candidate configs			1	1																▨																							
	Art14 Config Metrics			1																▨																						
	Art15 Config Assessments				1																▨																					
Outputs	Art8 Dsgn Goals, Metrics, Constraints	1																				▨																				
	Art13 Config def'n, analysis plans	1																					▨																			
	Art12 Candidate configs		1																					▨																		
	Art15 Config Assessments			1																					▨																	
	Art16 Selected config				1																					▨																
Methods	Mtd27 Document-Based Design									1								1				1	1				▨													
	Mtd14 (u) SME										1		1			1			1	1						1			▨												
	Model-Based Design									1								1				1	1						▨											
	QFD										1																				▨										
	Mtd5 (u) Brainstorming											1	1						1												▨									
	Mtd6 Morphological matrix													1	1										1									▨								
	TRIZ													1																				▨							
	Analysis of Interconnected Decision Areas (AIDA)														1										1										▨						
	Mtd7 modified decision matrix															1	1		1		1	1														▨					
	Multi-Attribute Utility															1			1		1																▨				
	Mtd3 (u) Committee																1		1	1		1																▨			
	Mtd13 Point-Based Design																1										1												▨		
	Set-Based Design																1										1													▨	

Figure 22: Use of multi-domain matrix for evaluating method compatibility, on excerpt of design strategy

The general algorithm for generating alternative methodologies through multi-domain mapping and analysis of the interactions between the domain of design process tasks and the domain of design methods is provided below. As will be seen, this "row-by-row" matrix analysis is fairly algorithmic and could be automated.

4.4.3.1 *Compatibility of Method Combinations with Tasks*

For each design task, available methods are deemed to be candidates if they are compatible with the task. Compatibility is defined either via input/output artifact or (basic) activity. In artifact-based (or input/output-based) compatibility, a method (or method combination, in general) is compatible if its inputs and outputs match those of the task. Note that method combinations, where multiple methods are "connected together" to collectively solve a task, need to be internally compatible themselves (or "method-method compatible") – we assume this information is known or given, to keep the focus of this illustration on method-task compatibility. If certain methods require to be used in groups/combinations, then perhaps they should be combined into one method, at least during method alternatives discovery. For example, in Figure 22, Task 25 (Act25) takes artifact Art8 as an input, as does baseline method Mtd5; it also outputs artifacts Art12, as do both baseline method Mtd6 and alternative method AIDA. Thus, candidate method combinations that are compatible with Task 25 based on artifacts include:

- Mtd5 – (internal connectivity) – Mtd6
- Mtd5 – (internal connectivity) – AIDA

In activity-based compatibility, a method combination is compatible if its underlying transformation matches that of the task. However, methods rarely come cleanly categorized by singular transformations (a single generic verb is too vague), unlike the modeled design task which was explicitly specified using a context-specific design activity (e.g. "market researching" in contrast to "gathering"). Instead, we decompose both the design activity and the methods into constituent "basic activities", which is a more standard approach to characterizing methods by transformation (subsection 2.2.4). For example, Act25 can be decomposed into four basic activities:

- Dividing (same as baseline method Mtd5)
- Clustering (same as Mtd5 and baseline method Mtd14)
- Arranging (same as baseline method Mtd6 and alternative method TRIZ)
- Combining (same as Mtd6 and alternative method AIDA)

Thus, candidate method combinations that are compatible with Task 25 based on activities include:

- Mtd5 – Mtd6
- Mtd5 – Mtd14 – TRIZ – AIDA

As demonstrated above, input/output compatibility is a fairly complete definition for method-task compatibility, but an additional method compatibility check by activity can help verify that the internal method processing matches that requested by the design task.

4.4.3.2 Initially-Specified Matrices

As previously stated, the input data for this procedure are:

- The initially-specified DMMs (B, D, F and H in Figure 21)
- Knowledge of which methods are compatible to use in groups/combinations

These initial matrices in Figure 22 can then be formed:

- (tasks) x (input artifacts)
- (input artifacts) x (methods)
- (tasks) x (output artifacts)
- (output artifacts) x (methods)
- (tasks) x (basic activities)
- (basic activities) x (methods)

4.4.3.3 Induced Matrices

The following matrices can then be induced from the initially-specified matrices in Figure 22:

- M1 = DSM of tasks related by all artifacts
- M2 = DSM of tasks related by input artifacts
- M3 = DMM of tasks related to methods by input artifact
    - If a task-method pair shares the same number of inputs as the number of task inputs (see M2), then that method is a candidate choice for that task based on inputs

    - That method may be required to be combined with other methods
- M4 = DSM of tasks related by output artifacts
- M5 = DMM of tasks related to methods by output artifact
    - Similar to inputs case
- M6 = DMM of tasks related to methods by all artifacts = M3 + M5
    - For each design task and for each compatible or required method combination, evaluate whether the task's inputs/outputs match the combination's aggregate inputs/outputs
    - If a task-method pair shares the same number of total artifacts (inputs + outputs) as the number of task total artifacts (see M1), then that method is a candidate choice for that task
    - It may be that not a single method but a method combination shares the same number of ins+outs, but it is hard to tell from this DMM because the counts are not unique, so artifacts could be double counted
    - Thus, it is best to evaluate inputs separately from outputs
- M7 = DMM of tasks related to methods by basic activity
    - If a task-method pair shares the same activity (or basic activities) as that of the task, then that method is a candidate choice for that task based on activities

The algorithm would then be, for each design task:

1. Enumerate combinations of methods, with some size cap (e.g. combine from one to no more than five methods)
2. For each combination, evaluate whether the design task's inputs/outputs and basic activities match the combination's aggregate inputs/outputs and basic activities. It may be desirable to relate tasks to methods by output artifact first before inputs, since outputs are the final goal of using each method.

4.4.4 Candidate Alternate Design Methods in Case Study

Certain tasks were chosen from the design process as a subset of the MDM form to illustrate the principles of the framework. The entire design process is available to see how an entire methodology could be selected. This subsection presents the four comparisons of method combinations for the four design tasks studied in Figure 22.

4.4.4.1 First Design Task

The baseline and alternative method combinations found to be compatible with Task 7 “Translate customer needs and program objectives” based on the matrix-based generation of alternative method choices in subsection 4.4.3 are shown in Figure 23, including connectivity details internal to each combination.

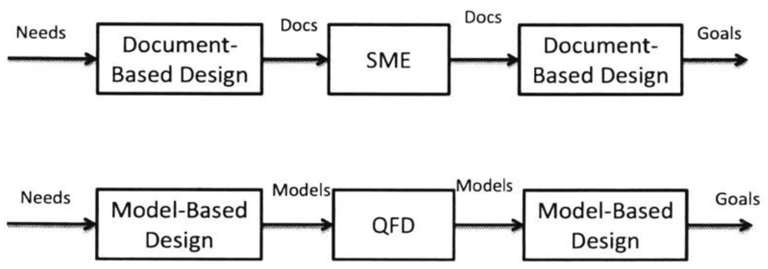


Figure 23: Baseline (above) and alternative (below) method combinations compatible with design task 7.

Note that the baseline method Mtd27 “Document-Based Design” (DBD) and the alternative Model-Based Design (MBD) carry out the implicit information exchange task that pervades the entire design process (subsection 4.3.4) but is only shown explicitly for this case. DBD and MBD essentially translate the artifacts into/from tangible formats that the core methods (baseline method Mtd14 SME and alternative Quality Function Deployment (QFD)) operate on.

From an artifact standpoint, the alternative method combination preserves the internal flow of information and simply substitutes the use of SME experience to flow customer objectives down to system design goals with the use of the QFD method (and substitutes the use of documents with models for actually handling the data).

From a basic activity standpoint, Task 7 (including the implicit information exchange task) consists of "substantiating" and "documenting" actions. Both DBD and MBD are documenting, and both SME experience and QFD are used for substantiating.

4.4.4.2 *Second Design Task*

The baseline and alternative method combinations found to be compatible with Task 25 "Generate candidate concepts" are shown in Figure 24.

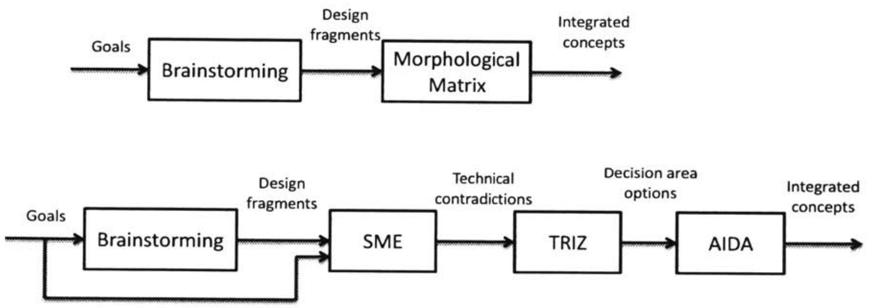


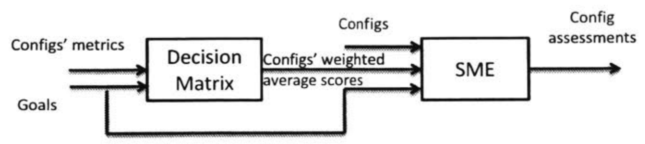
Figure 24: Baseline (above) and alternative (below) method combinations compatible with design task 25.

From an artifact standpoint, the alternative method combination uses AIDA to provide the final integrated concepts instead of the morphological matrix method. The brainstorming method directly fed the morphological matrix in the baseline case, but in the alternative, the SME is brought in to help use the TRIZ method first to provide the information required by AIDA. In other words, the SME, TRIZ and AIDA methods together form the alternative to the morphological matrix method.

From a basic activity standpoint, Task 25 consists of "dividing," "clustering," "arranging," and "combining" actions (Braun and Lindemann, 2003). The baseline combination uses brainstorming (dividing and clustering) and the morphological matrix (arranging and combining); the alternative combination of brainstorming, the SME, TRIZ and AIDA each perform dividing, clustering, arranging and combining, respectively.

4.4.4.3 *Third Design Task*

The baseline and alternative method combinations found to be compatible with Task 17 "Qualitatively assess candidate concepts" are shown in Figure 25.



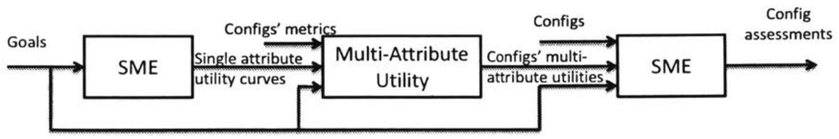


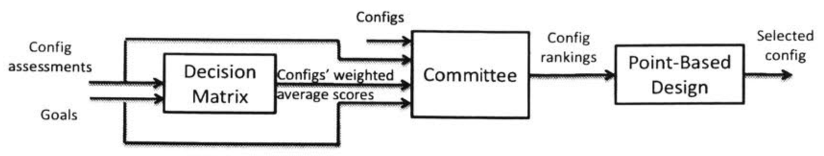
Figure 25: Baseline (above) and alternative (below) method combinations compatible with design task 17.

From an artifact standpoint, the main change is that the alternative method combination uses the Multi-Attribute Utility (MAU) method in the place of the baseline decision matrix method. MAU requires an additional input (single-attribute utility functions) that the decision matrix did not need, precipitating an added use of the SME's experience at the start of the method combination's information flow.

From a basic activity standpoint, Task 17 simply performs "assessing," the same as the baseline and alternative method combinations.

4.4.4.4 *Fourth Design Task*

The baseline and alternative method combinations found to be compatible with Task 20 "Select concept configuration" are shown in Figure 26.



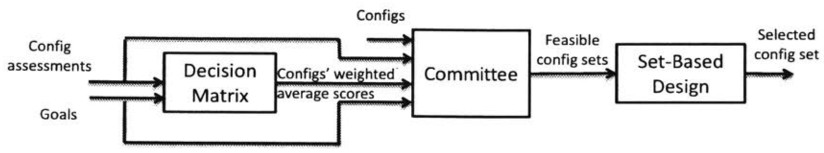


Figure 26: Baseline (above) and alternative (below) method combinations compatible with design task 20.

From an artifact standpoint, after the decision matrix and committee methods establish preference among the configurations, the baseline method combination uses a point-based approach to select the solution concept, while the alternative allows a set of solutions to proceed. Thus, internal to the alternative combination, the committee needs to frame the configurations in terms of sets instead of point solutions. Moreover, since the final output artifact of the alternative method combination is a set of configurations, the designer will need to determine if downstream methods or the design process itself need to be adjusted. Similarly, the fact that the third method combination comparison considered replacing the decision matrix method with a MAU approach needs to be monitored by this fourth comparison – if the MAU method is used for Task 17, qualitatively assessing candidates, then the designer executing Task 20 may want to use MAU to process the configuration assessments in a fashion consistent with how it was done in Task 17.

From a basic activity standpoint, Task 20 simply performs "selecting," the same as the baseline and alternative method combinations.

4.4.5 Descriptions of Alternate Methods for Case Study

Descriptions for the design methods suggested by subsections 4.4.3 and 4.4.4 for the alternate design methodology are now provided.

- Model-Based Design: use of a shared system-level model with multiple views, connected to disciplinary models; requirements, design information and test procedures are integrated in one model
- Quality Function Deployment: maps customer needs to product functions, or one domain to another domain in general, while considering correlations within the domains and competitive benchmarking
- TRIZ (Theory of Inventive Problem-Solving): structured innovation using analogy; law of Ideality: more function with less form; Technical Contradictions among 39 Technical Characteristics can be resolved using 40 TRIZ Principles
- Analysis of Interconnected Decision Areas: identify several feasible options in each decision area; indicate which options are incompatible with others; list all of the sets of options that can be combined together without incompatibility; when there is a single quantifiable criterion of choice (e.g. cost) find the compatible set of options that best satisfies that criterion
- Multi-Attribute Utility: elicit single attribute utility curves and multi-attribute utility attribute weights; score alternatives with single attribute utility curves; alternative with highest multi-attribute utility is selected
- Set-Based Design: design progression revolves around sets of solutions instead of point solutions; emphasis on justifying eliminating alternative concepts, instead of justifying preserving concepts; at each stage, focus on the overlap between each group's possible design space; in contrast to point-based design

4.5 Characterizing Methods with Attributes

4.5.1 Sources of Method Attributes

To obtain a general picture of the appropriate path to take for method attribution to support selection decisions, the interviewees were first queried for their preconceived notions of which method attributes were important to their choice of methods and to the design effort in general. The cost of using a method was a recurring theme in the respondents' answers. They also highlighted the speed of method use and design cycle time as a competitive differentiator in the face of customer turnover. The risk and track-record associated with a method was also considered noteworthy, as was the designer's familiarity with a method alternative.

A survey of method attribute frameworks from the literature provides more holistic sourcing of attributes that could be relevant to method selection. As mentioned in subsection 2.2.4, the Process-oriented Method Model (PoMM) from Birkhofer et al. (2002) is a systematic method database that uses a mix of "access modules" (overview information providing "gateways" into each method) and "process modules" (method implementation information for application in a design process). A method's process modules include:

- Input: material, information, documents
- Sequence (or procedure): Structure (procedure steps related to each other) with Descriptions for each step
- User: skills, qualifications, motivation, experience
- General conditions: procedure, infrastructure, company, colleagues
- Working aids: tools, troubleshooting
- Hints
- Output: information, documents, knowledge, direct and indirect consequences

The User, General Conditions, Working Aids and Hints process modules could be considered features of the various types of method enablers, in IDEF0 terms (Menzel and Mayer, 2006), that are provided by the design Organization and design Resources domains, in Product Development System terms (subsection 2.2.2). A method's access modules include:

- Selection: classification (e.g. by PDP phase, by kind), relationship to other methods
- Specification: working principle, history, definition, aims, benefits, keywords
- Links: videos/multimedia, literature, examples, stories

The Munich Model of Methods (MMM) from Braun and Lindemann (2003) also referred to in subsection 2.2.4 employs method attributes already included by the PoMM, such as requested inputs, procedure and active principle, and required degree of intuition.

Schmidt (2015) uses two groups of attributes to characterize a compilation of design methods. The first group pertains to method categorization along the following lines:

- PDP phase: planning, concept design, testing, etc.
- Domain of product description: physical, functional
- Breadth of solution space: individual, set, expanding/diminishing space
- Formality of method

The second group assesses method uncertainty:

- Type of uncertainty: physical, functional
- Source of uncertainty: market, corporate, product, use, political, cultural
- Reason for uncertainty: lack of definition, knowledge, trust of knowledge

Schmidt (2015) also describes several method metrics organized by method purpose. For example, ideation-oriented methods can be measured by their novelty, variety, quantity and quality; risk assessment methods can be gauged by their failure mode control effectiveness and failure mode identification miss rate; and so on.

Hubka and Eder (1996, p. 133) would classify a method attribute into one of two classes. In one sense, methods are tools to be applied that have:

- Goals (purpose)

- Breadth of application: one field or several
- Necessary conditions, e.g. resources
- Intended users: type of operator/designer, preconditions, individual or group
- Sources: originating fields
- Mechanisms: how it works, behaves
- Time demands

Methods also act as forms of information with the following properties:

- Content correctness, accuracy
- Verifiability of effect
- Completeness
- Format understandability
- Information timeliness, currency
- Universality of application

The tool- or application-oriented attributes from Hubka and Eder (1996) have similarities with those considered by Ernzer and Birkhofer (2002), which include implementation effort, application effort, design phase focus, and lifecycle phase focus. And Jones (1992) uses a comparable template to standardize descriptions of the methods in his collection with attributes such as aim/purpose, procedure, effectiveness, usability, common applications, learning time/training needed, and total elapsed time or cost needed.

4.5.2 A Generic Method Attribute Framework

The method attributes from the literature were synthesized into a comprehensive attribute set. This set consists of the following top-level attributes:

1. Method Classification: The category the method falls under. The categorization scheme can be left to the enterprise to help the attribute framework align with existing work or organization breakdown structures and minimize disruptions to current workflows.

Possible schemes (see subsection 2.2.3) include classifying by discipline or field, typical PDP phase, and design activity or function/purpose.

2. Method Inputs: The information and materials acted upon by the method. High-level descriptions of inputs are appropriate for assessing method compatibility with tasks, while detailed descriptions may be needed for comparisons between methods and certainly required for implementing the method.
3. Method Outputs: The information and materials produced by the method. Analogous to the Method Inputs attribute.
4. Method-Supplied Quality of Outputs: This attribute qualifies the benefit – or the grade, caliber or condition of the benefit – derived from the Method Outputs.
5. Method-Demanded Quality of Inputs: This attribute specifies the condition of the Method Inputs that is requested by the method.
6. Method Processing: The method's procedure, or its elementary working steps and basic activities (see subsection 2.2.4) and how the steps relate to each other (structure), along with the principle at work behind the procedure and the timing/dynamics involved.
7. Method Enablers: This includes the prerequisite working environment traits, resources (infrastructure, staffing level), user characteristics (skills, experience, motivation), and support implements (tools, working aids, examples, hints) from the design Organization and design Resources domains.
8. Method Controls: The oversight of the method "owner", governing enterprise policies, and constraints on method definition and behavior.
9. Method Implementation Costs: The total lifecycle costs involved with using the method. Types of costs include monetary funds (e.g. labor, material, license fees), time demands (e.g. labor effort, computer runtimes), other "consumables", and efficiency measures (units of output per unit time). Time demands are usually the limiting factor in aerospace conceptual design, though funding and budget constraints exist as well. These costs

could be recurring or nonrecurring, and distributed among different phases of method deployment.

This attribute set breakdown is by no means unique, of course. Additional attributes can also be derived from this set, e.g. method "completeness" can be judged based on the Method Outputs along with context that defines what is considered complete.

The following treatment of sub-attributes for the Method-Supplied Quality of Outputs and Method Implementation Costs attributes was first developed by consolidating classifications from the literature and then refined with the interviewees over the course of the case study. The attribute of Method-Supplied Quality of Outputs can be decomposed into several sub-attributes for a more precise definition of output quality:

- Accuracy: average proximity to the "truth"; a measure of the method's systematic error and average fidelity
- Precision: the amount of "scatter" in the data; a measure of the method's random error and output reproducibility given the same inputs and method procedure (but different/varying enablers like users and/or tools)
- Accessibility: the usability of the format the output comes in

It is also possible to derive additional quality sub-attributes from this collection, e.g. output uncertainty is a function of accuracy and precision. Note that method "risk" is not only determined by output uncertainty but also contextual factors such as the importance of method accuracy in that situation. Other proposed sub-attributes were rejected, such as "robustness" of outputs to changing inputs, which was deemed to have too much overlap with the other sub-attributes because they already consider average accuracy and precision for a full range of inputs.

Likewise, Method Implementation Costs can be allocated by the method deployment phases in which they are incurred, as follows:

- Cost to decommission the baseline method (if applicable): to “un-learn” or “uninstall” the method currently commissioned for use, if necessary to “make room for” the method currently being considered as an alternative; usually only noteworthy if there is significant inertia associated with the baseline method that would need to be overcome
- Cost to commission the (alternative) method: to transition from the current state to a state where the method can be executed; the nonrecurring cost of method implementation that may include:
    - Method adaptation costs to tailor a generic method for the situation at hand
    - Method installation costs that put enabling platforms and tools in place for access by designers
    - User learning curve, gaining required skills, experience, and working environment
    - Analytical validation based on objective evidence
    - Intangible validation based on firsthand internal experience, confidence, familiarity, trust-building, “accreditation” and concurrence/buy-in from leaders and non-advocates
- Cost to execute the method once: the recurring cost of method use after its commissioning, each time it is used in and of itself; e.g. computer runtimes, labor hours to complete an iteration
- Cost to maintain the method: to preserve a general state where this method can be executed; the recurring costs of method use over the lifecycle of the development project, or over multiple projects, that are not directly associated with any particular instance of method use; e.g. the low efficiency and costs of manually synchronizing design documents in document-based design that are not present in model-based design

Besides attribute decomposition, another issue to address is how to determine states for each attribute (see subsection 2.2.4). Consistent scales should be used for the same attribute across all methods and tasks. The granularity of the scale could influence the difficulty of attribution: a less granular scale is simpler but more ambiguous, while a more granular scale is more involved to utilize but less imprecise. A simple ordinal scale (e.g. 1-2-3-4-5, low-medium-high) was used during this case study to quantify the state of each attribute, the rationale being that the relative comparisons between the attribute states of methods would be the relevant concern, not the state values in absolute terms. Indeed, designers interviewed for this study commented that absolute references and state definitions could change over the design lifecycle, but the key is a common reference for all methods at any given point in time.

4.5.3 Attributes Relevant to Method Selection

During development of the case study, the generic method attribute set was down-selected to those attributes most relevant to the method selection problem. The designer should not be saturated with too much information during the initial step of choosing a method. This especially holds if they are comparing multiple methods and only want to deal with those attributes key to the selection decision. According to Braun and Lindemann (2003, p. 5): "...we distinguish between attributes which are particularly relevant for the selection of a method and attributes which mainly concern the adaptation or application of a method." For example, the primary selection-focused method attributes from Braun and Lindemann (2003) include the method-requested input, the requested resources, and the achievable output; secondary attributes relevant to method selection include the active principle behind the method, the method's classification (e.g. by PDP stage), and the required intuition of the user.

To aid the search for selection-focused method attributes, method selection itself is divided into three stages or tiers. In the first stage or lowest tier, the basic compatibility of the method with the design task is evaluated. As covered in section 4.4, if the inputs, outputs, and internal processing of the task and method align, then the method's purpose or function meets the task entailed, the fundamental feasibility of using the method is established, and the lowest tier of selection has been reached. The next tier seeks meaningful comparisons between multiple compatible methods. This second tier goes beyond discrete/ binary compatibility of method-task function to evaluation of the continuum of performance in methods delivering that function, thus enabling the designer to discriminate between "equally" compatible methods. Through the use of criteria for conducting method tradeoffs, the designer passes the second tier once a preferred method has been determined after comparing several feasible alternatives. The third tier of method selection involves verifying the viability of this method choice, perhaps by simulating the application of this particular method with sufficient detail to convince stakeholders that the correct choice was made. More detailed method attributes, such as procedural details, would probably be called upon at this point to reach this third tier.

From this three-tier model of method selection, the first two tiers (compatibility and comparison) are the tiers required to arrive at a method choice. With compatibility established by the approach laid out in section 4.4, the second tier of method comparison is used to focus the search for method attributes critical to the selection decision. After accounting for stakeholder feedback from the case study interviews, it is proposed that method comparisons only require the following attributes: Method Inputs and Outputs (to sufficient detail for comparison), Method-Supplied Quality of Outputs, and Method Implementation Costs. The intent was for most designers to find these attributes, along with their sub-attributes, to be important to consider in most design method selection situations when conducting method tradeoffs. More detailed comparisons between the inputs required and the outputs achieved by the methods would verify the methods' compatibility with the current design task. And output quality and implementation cost comparisons enable basic cost/benefit trades.

4.5.4 Method Use Context Drivers of Attribution

To this point, only indirect references have been made to the design situation and contextual factors that influence the characterization of methods by attributes and the subsequent selection of a method to meet that design context. The context of method use and its implications on method attribution and method selection are now explored and defined.

As an illustration of how attribution depends on method use context, the accessibility sub-attribute of a design method's output quality can be seen as dependent on how much method implementation effort is applied (how much the method is tailored for the application at hand, how much training is given to the user, etc.), the status of method-enabling elements (the suitability of available infrastructure to display the method output, the designer's inclination towards the output format, etc.), the type of product and/or task the method is being applied on, and perhaps other factors.

Note that these contextual factors are not necessarily the same as those that influence the relative importance of the accessibility sub-attribute during the method selection decision. For instance, lack of suitable method enablers may drive down the state of the method's output accessibility quality, but if there is no project goal to make the format of this design task's deliverable accessible to a wide audience (e.g. presentations to external customers), then the method's low accessibility is relatively less important as a part of its overall comparison with other methods. Therefore, a distinction is made between attribution context and selection context – the latter is more driven by factors related to the Goals domain of the Product Development System (subsection 2.2.2) than the former is. Selection context is further discussed in section 4.6.

Returning to attribution context, it is evident that there is a need to clarify the task and the method use situation in order to characterize the attributes of a method and successfully apply methods. Method output quality is often a function of details behind the method's enablers, e.g. quality could rise with method user experience or expertise. Method implementation cost depends on current design mentalities, e.g. if designers are already comfortable operating with a shared system model representation environment, the Model-Based Design method is less costly to implement from user learning curve and trust-building standpoints. According to Bevan (2009), a method's value depends in part on characteristics of the specific task, product, and available skills and access to stakeholders at hand.

Table 2 portrays how various contextual factors affect the selection-focused attributes from subsection 4.5.3, thus allowing method portrayals to be tailored to use context. The way the Method Inputs and Outputs are described can be molded to fit the particular product being designed and the specifics of the current design task (e.g. translating customer needs and translating engineering competency considerations are worded differently but could use similar methods). The Method-Supplied Quality of Output attribute is provided at some level of implementation cost (e.g. more implementation effort could improve output quality), for a given task and product, given some enabler qualities. Likewise, the Method Implementation Cost attribute is dependent on the level of output quality to be achieved, for a given task, product and set of enabler qualities, starting from an existing method situation.

Table 2: Method use context influences on method attribute states

Context Elements \ Attributes	Inputs	Outputs	Quality of Outputs	Method Implementation Cost
Task*	e.g. "Task 7: Translate customer needs and program objectives"			
Product*	(The nature of the product can affect all method attributes)			
Nature of Enablers	-	-	e.g. designer skills, experience	
Nominal Output Quality	-	-	-	(assumption for cost attribution) e.g. within 10% of truth
Current Stance Towards Method	-	-	-	e.g. affinity for baseline vs. alternative methods
Nominal Method Implementation Cost	-	-	(assumption for quality attribution) e.g. <100 hours	-

*potentially independent of specific methods

According to the understanding of attribution context displayed in Table 2, the particulars of the design task and product could affect the attributes of all methods being considered uniformly.

For example, if an input artifact description is updated to reflect the current design task description, and the same input artifact is used by several design activities, each of which needs a method or method combination to be executed, then all of the compatible methods or method combinations being considered to solve the various design activities could use the same input description. Furthermore, it is proposed that the task and product context affect all of the selection-relevant attributes of a method. As Andreasen et al. (2015, p. 56) assert, "A method should always be understood in combination with the design activity it will support. The result of the method is rarely the result of the wider activity."

The "nature of enablers" context in Table 2 encompasses both the kinds (e.g. staff with different backgrounds) and quality (e.g. staff experience/skill levels) of method enablers that are available, which can affect both method output quality and method implementation costs. Similarly, the "current stance towards method" context details the mirror issues of the baseline method situation and prospects for alternative methods (e.g. baseline method entrenchment, alternative method validation opportunities), which can influence method implementation costs.

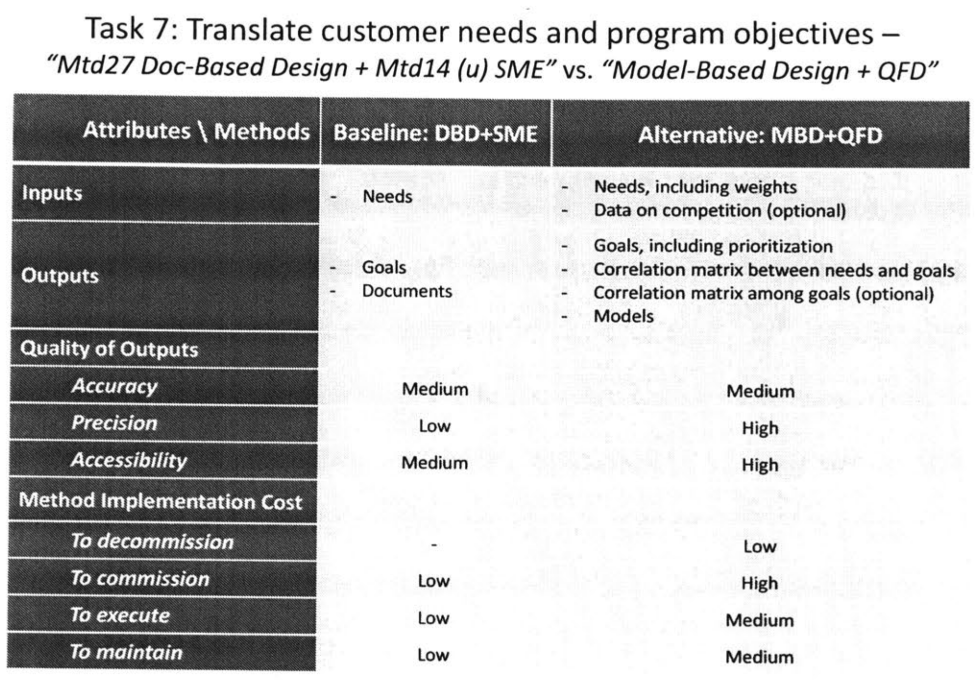
Finally, as touched on already, a nominal output quality level to be achieved is needed to "baseline" implementation cost estimates. The reverse is also true: a nominal method implementation cost bogey provides a reference for comparing the output quality from multiple methods.

4.5.5 Attribution of Methods in Case Study

Based on the generic method attribute framework laid out thus far, the baseline and alternative method combinations considered by the case study for the same sample of design tasks from subsection 4.4.4 were characterized using attributes relevant to method selection, based on stakeholder interviews and primary sources, as shown in Table 3, Table 4, Table 5 and Table 6.

For instance, the Task 7 method combinations in Table 3 are characterized using the language of needs and goals in Task 7 as context, with the same nominal output quality and implementation cost levels in mind for both baseline and alternative for a consistent comparison. The inputs for task 7 include solicitation-stipulated requirements and desirements, the interpretation of the solicitation, and the wider business strategy, according to Figure 20. Outputs include selected and interpreted solicitation requirements, metrics with conditions and targets, and a basis for each requirement. The baseline method (mainly the SME's expertise) produced fairly accurate output in that no design goal important to the customer was left out, but it was thought that if the QFD method was employed, the resultant design goals would not vary as much with the individual designer performing this task. However, the QFD method would require training and is more involved to populate and engage with stakeholders, while employing the SME to translate needs into requirements was very fast and low cost since that was the established method with little overhead.

Table 3: Baseline and alternative method attributes comparison for Task 7.

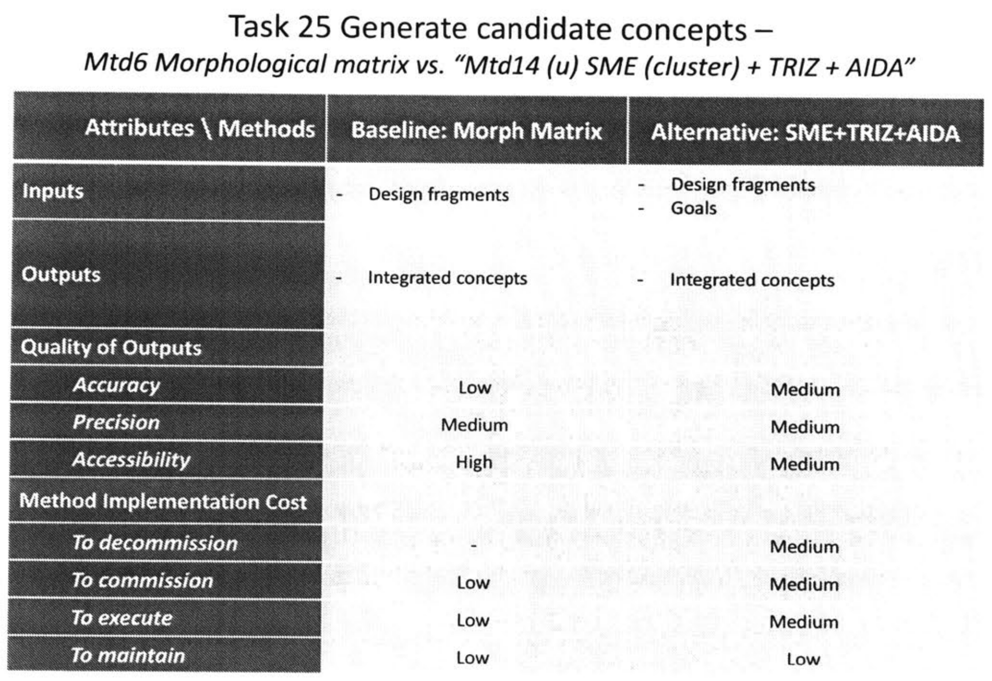


Task 7: Translate customer needs and program objectives –

"Mtd27 Doc-Based Design + Mtd14 (u) SME" vs. *"Model-Based Design + QFD"*

Attributes \ Methods	Baseline: DBD+SME	Alternative: MBD+QFD
Inputs	- Needs	- Needs, including weights - Data on competition (optional)
Outputs	- Goals - Documents	- Goals, including prioritization - Correlation matrix between needs and goals - Correlation matrix among goals (optional) - Models
Quality of Outputs		
Accuracy	Medium	Medium
Precision	Low	High
Accessibility	Medium	High
Method Implementation Cost		
To decommission	-	Low
To commission	Low	High
To execute	Low	Medium
To maintain	Low	Medium

Table 4: Baseline and alternative method attributes comparison for Task 25.



Task 25 Generate candidate concepts –

Mtd6 Morphological matrix vs. "Mtd14 (u) SME (cluster) + TRIZ + AIDA"

Attributes \ Methods	Baseline: Morph Matrix	Alternative: SME+TRIZ+AIDA
Inputs	- Design fragments	- Design fragments - Goals
Outputs	- Integrated concepts	- Integrated concepts
Quality of Outputs		
Accuracy	Low	Medium
Precision	Medium	Medium
Accessibility	High	Medium
Method Implementation Cost		
To decommission	-	Medium
To commission	Low	Medium
To execute	Low	Medium
To maintain	Low	Low

Table 5: Baseline and alternative method attributes comparison for Task 17.

Task 17 Qualitatively assess candidate concepts –
Mtd7 modified decision matrix vs. Multi-Attribute Utility

Attributes \ Methods	Baseline: Modified Decision Matrix	Alternative: MAU
Inputs	- Goals - Metric results of candidate concepts	- Goals - Metric results of candidate concepts - Single attribute utility curves
Outputs	- Weighted average scores of candidate concepts	- Multi-attribute utilities of candidate concepts
Quality of Outputs		
Accuracy	Medium	High
Precision	High	High
Accessibility	High	High
Method Implementation Cost		
To decommission	-	Low
To commission	Low	Medium
To execute	Medium	High
To maintain	Low	Low

Table 6: Baseline and alternative method attributes comparison for Task 20.

Task 20 Select concept configuration –
Mtd13 Point-Based Design vs. Set-Based Design

Attributes \ Methods	Baseline: PBD	Alternative: SBD
Inputs	- Rankings of candidate concepts	- Feasible sets of concepts
Outputs	- Selected concept	- Selected concept set
Quality of Outputs		
Accuracy	Low	High
Precision	Low	Medium
Accessibility	High	Medium
Method Implementation Cost		
To decommission	-	High
To commission	Low	High
To execute	Medium	High
To maintain	Low	Medium

4.6 Method Selection Tradeoffs

4.6.1 Case Study Method Comparisons through Attribute Differencing

As a first step on the path of conducting method trades to arrive at a preferred method choice, the method attribute information illustrated in subsection 4.5.5 can be used to compare the baseline and alternative methods. The designer ultimately wants the method comparisons to be commensurate, where attributes differences are compared while holding other variables constant or ceteris paribus. The framework seeks to structure and pose the question: "How would a designer of experience level X using method-enabling resources of nature Y, from existing method situation Z, fare (in terms of attributes) when executing task A for product B using method 1, as compared with method 2?" Differencing the method attributes produced by the present framework provides a mechanism for doing so.

The Task 7 method attribute differences in Table 3 again exemplify the point. The QFD method input explicitly needs weightings among the customer needs as the bases of goals, whereas the SME may determine such weightings implicitly. On the other hand, QFD explicitly provides degrees of need-goal (or basis-goal, i.e. "what-how") correlation and prioritizes the resultant goals; it can also provide insights into relationships and correlations among the goals and openly support competitive analysis if such data is provided, all of which enriches the method output. From an output quality standpoint, QFD is seen as providing more repeatable (and therefore less uncertain) results in a more consistent format. However, QFD involves higher startup costs and longer execution times.

4.6.2 Method Selection as Trades

These method attribute comparisons via attribute differencing can be used to support an overall design methodology trade study that seeks the appropriate set of methods to use for a given design situation. When formulating a design methodology to carry out a design process, the design team frequently needs to characterize and difference method combinations, not just individual methods. Nonetheless, a practical implementation of some decision support system needs to first characterize individual methods before generating method combinations compatible with particular design tasks (section 4.4).

The design method selection problem was expected to reduce to a trade study eventually (see section 3.3). In fact, Baker and Whalen (2012) would select a method for conducting trade studies by performing another trade study! In their case, each trade study method was evaluated by the following criteria: amount of time needed to conduct the study, amount of data needed, and accuracy. Analogously, design method attributes – inputs/outputs, output quality, and implementation costs – constitute part of the criteria by which we assess the methods' relative suitability and value for solving design tasks. This method attribution is accomplished in a specific manner, as described in section 4.5, to account for the design context and ensure commensurate comparisons.

The other part of the method selection criteria that is needed is some notion of targets for and/or relative importance among the attributes. In other words, it is difficult to tell whether, say, low method output quality is concerning or not without first establishing elements of situational context that are relevant to method selection, as alluded to in subsection 4.5.4. Selection context distinguishes one method from another, as opposed to attribution context which drives a given method's attribute state regardless of the state or "performance" of other methods. Thus, attributes combined with selection context (particulars of the design situation, relative weightings among attributes, single attribute utility curves, etc.) comprise the criteria by which preference can be established among multiple alternative methods.

4.6.3 Approaches for Conducting Trades to Choose Methods

In the case study, "trade studies" to select design methods were performed over the course of the design team dealing with limited resources while deciding what kind of design analysis to perform. The expertise of SMEs provided bounds on the problem as the team considered the development program situation and generated value judgments, i.e. the value of higher order methods in terms of additional performance or insights at some additional cost or duration. Leadership would pose a series of questions to the designers to explore the decision space until the team converged on a methodology path forward. From the literature, it is understood that consideration of the nature of the project (e.g. product novelty, lifecycle phase) and the design tasks and problems to be solved should inform the method selection criteria (Ernzer and Birkhofer, 2002).

Inspiration for possible approaches to conducting method trades using the method attribution illustrated by the case study can be found by returning to the design system (section 2.2) and considering how project design goals flow down to the design methodology through the five design system domains, as depicted in Figure 27. Design objectives on product performance and development cost and schedule flow down and affect product design and design process properties. Product design properties could also drive aspects of the design process, e.g. a more complex or novel vehicle configuration could compel additional design process iterations. Design process objectives decompose into objectives on individual design task attributes that flow down themselves to design method attributes. At the same time, design process objectives also map to objectives on overall design methodology properties that decompose themselves into individual design method attributes.

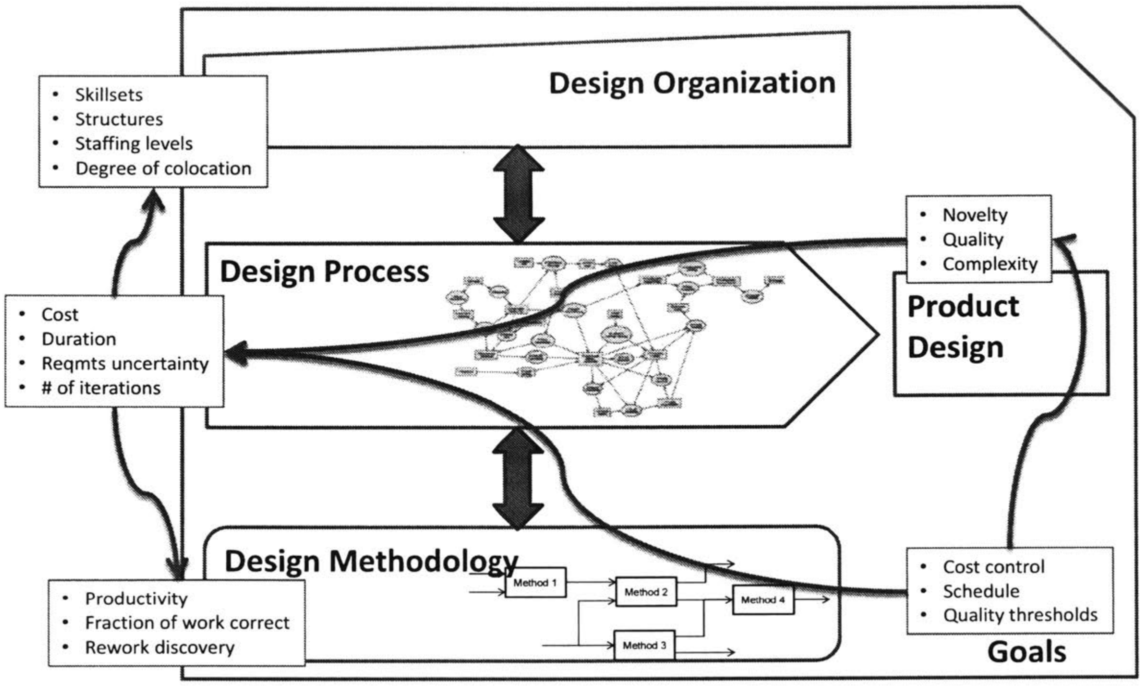
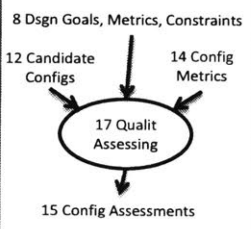


Figure 27: Aggregate design methodology requirements traceability

Therefore, there are at least two broad approaches to determining method use context drivers for selection that can be used to make choices based on attribute differencing – both of which are premised on the notion of inducing systemic impacts on design project performance from the lower-level method attributes. The first approach compares the method attributes with attributes of the design task to find which method has a set of attributes with the best fit to the task. The idea is that method attributes should trace back to task attributes that aggregate to yield global design process properties. This concept is portrayed by Table 7. The impacts of method attributes on task requirements are evaluated, e.g. budget/schedule constraints could limit the number of iterations undertaken with a method (and therefore also limit the output quality of the method).

Table 7: Design task-method attributes comparison

Activity & Artifacts	Task	Task Attributes	Method Attributes	Methods	
				Modified Decision Matrix	Multi-Attribute Utility
 8 Dsgn Goals, Metrics, Constraints 12 Candidate Configs 14 Config Metrics 17 Qualit Assessing 15 Config Assessments	Task 17: Qualitatively assess candidate concepts	Inputs Outputs Basic Activities Complexity Upstream information evolution rate Downstream sensitivity Duration Effort	Inputs Outputs Quality of Outputs - Accuracy - Precision - Accessibility Method Implementation Cost - To decommission - To commission - To execute - To maintain	- Goals - Metrics Weighted average scores - Medium - High - High - n/a - Low - Medium - Low	- Goals - Metrics - Single attribute utility curves Weighted average scores - High - High - High - Low - Medium - High - Low

Different task attributes are found from the literature, some of which are included in Table 7. The familiar ones from Braun and Lindemann (2003) are that a task has inputs (requirements, preconditions), application conditions and boundary conditions (or enablers and controls), and outputs (target conditions, objectives). Task attributes described by Krishnan (1993) include:

- Information dependencies (e.g. inputs/outputs)
- Utility of inputs/outputs
- Availability of inputs/outputs
- Influence/impact of inputs/outputs
- Timing of information exchange
- Content of information exchange
- Output evolution rate, determined by:
    - Upstream evolution: how fast upstream information changes

  - Downstream sensitivity: how much the task's output changes if its input changes

Additional task characteristics from Bevan (2009) include task complexity, required output quality, and available time. Later on, task attributes can be aggregated into higher-level design process attributes or global impacts on the design process/project.

The second approach stays in the design methodology domain and aggregates the individual method attributes into global methodology properties; comparison of methodology properties with objectives on design process properties then follows. In other words, aggregating individual method attributes yields higher-level methodology properties that trace back to design process properties. The following methodology-level characteristics based on project system dynamics can be inferred:

- Raw Productivity ~ output per unit time
- Real Productivity ~ output * quality per unit time
- Fraction Correct (Rework Avoidance) $\propto$ quality
- Rework Discovery ~ quality * time

This approach may not be feasible at the earliest design stages in that the full methodology is not known yet.

5 Case Study: Results

In this chapter, the reflections of the interviewees on the framework for method selection are summarized and the framework survey results presented.

5.1 User feedback on design strategy modeling

Case study interviewees not only commented on opportunities to refine the content of the design strategy model (subsection 4.3.4) but also on their views of the modeling exercise itself. A documented representation of the design process and methodology promoted a transition from tacit to explicit understanding of the underlying design workflow and assumptions. The graphical format of Object-Process Diagrams helped facilitate communication during the interviews and modeling iterations (subsection 4.3.1).

This overt approach to documenting the design strategy resulted in a cogent and sound model as perceived by stakeholders. Designers agreed that several different models of what transpired during the flight vehicle R&D program were possible and still valid, and that the model presented in this thesis was a reasonable version of the events and reasoning behind the rapid prototyping effort. This starting point is seen as a foundation for the ability to infer a generic form of the conceptual design process from a specific case study. Doing so would illuminate the correspondence between local practice and enterprise standards.

5.2 Towards agreement on a set of method attributes

Interviewees were asked the following question: How important is each method attribute and sub-attribute in this case study to design method selection? Put another way: Is there any attribute here that would always be weighted lightly when selecting methods? If so, then it is of low importance generally and should be dropped from the list of those attributes especially relevant to method selection. If an attribute is at least sometimes important, then it should be kept on the list.

5.2.1 Method Inputs

Method Inputs were deemed to be of moderate to high importance to the method selection decision. A recurring observation was the dependency of other attributes on how the Method Inputs were defined, e.g. method output quality depends on the inputs available. Adding or removing inputs and how the design task is defined can affect method attribution. For example, if an input is added as a given or provided, the precision of a method can be increased because that variable has now been constrained.

In addition, one could parameterize or re-formulate the task so that a different method could be employed, especially if the desired method output is insensitive to input particulars. If X number of inputs are available, any method that requires up to X number of inputs can be employed, assuming the method is compatible with the task in other respects (i.e. outputs and basic activities). There was interest in additional guidance and future discussion on how to describe method inputs, for evaluating both method compatibility with tasks (generic) and comparison between methods (context-specific).

5.2.2 Method Outputs

Method Outputs were generally regarded as highly to extremely important for method selection. Outputs are the highest priority because the method has to provide what the designer requests. Outputs are at least more important than inputs, although both inputs and outputs need to be considered when formulating the entire design methodology and dealing with method networks and connectivity. The Method Outputs also provide rationale for other attribution. For instance, low accessibility quality for a method's output could be driven by the type or format of the output itself.

5.2.3 Method-Supplied Quality of Output

Moderate to high importance towards method selection was rendered to the Accuracy sub-attribute for output quality. It was not given paramount importance because, during the earliest stages of design, the key issue is the relative performance and value of various vehicle configuration concepts being traded off, which slightly attenuates the need for absolute accuracy from design methods at that point.

There was a wider spread of opinions on the importance of the Precision sub-attribute, though it eventually converged to a range from low to moderate importance. While the camp of system-level designers and technical leadership did not see great value in knowing a method's Precision, a more disciplinary or functional design camp asserted that a method with low Precision would be worrisome.

Less consensus existed on the need for the Accessibility sub-attribute. The functional or technology design camp of interviewees deemed Accessibility to be less important than Accuracy and Precision because of a fairly high tolerance for low Accessibility methods. However, the enterprise leadership and systems engineering camp put high to extreme importance on Accessibility.

5.2.4 Method Implementation Costs

The composite approach of combining time and money expenditures into one composite "cost" was acceptable to the interviewees, though they recognized that their main focus is on time and that the perspective of the Finance organization would be needed to evaluate monetary impacts. They understood that the method selection decision support framework was geared towards the product designers, but commented that other functions may need to supply buy-in or inputs to the decision support system.

The cost to decommission the baseline method was believed to be a more minor consideration, even of no importance in some quarters, until method-enabling infrastructure or ecosystem constraints begin to apply. The tolerance for decommissioning costs also depends on the current method stance context. Thus it appears that decommissioning costs should be kept on the list of attributes to consider during method selection. Alternative method commissioning costs were deemed highly to extremely important: early conceptual design time constraints cannot tolerate significant method startup costs.

It was suggested that the distinction between baseline method decommissioning costs and alternative method commissioning costs was unnecessary. "Unlearning" and "uninstalling" the old method could be bookkept in the same category as "learning" and "installing" the new method. There may be a case where the two costs are of different magnitudes, e.g. the old method is very ingrained in the organization's process and mindset and the tool infrastructure drives a high decommissioning cost, while the new method has very low cost to set up and get started – but since both are still one-time, nonrecurring startup costs, it may not be necessary to distinguish between them. The important distinction to make is with recurring execution and maintenance costs.

Execution costs are important, perhaps moderately so, for methods that are run frequently or over long lifecycle products. Its importance depends on the frequency of executing the design task (an element of selection context), implying that execution costs should also be kept on the list.

One camp of interviewees considered method maintenance or sustainment costs to be of high importance because of limited resources in development projects, but another camp thought they were not important – since the development program cannot sustain maintenance costs to begin with, these costs are often pushed off to downstream groups or phases and not under the responsibility of the conceptual designers. This harkens back to the paradox of sequential or serial design mentioned in subsection 2.1.3, where committed costs grow much faster than knowledge about the design and incurred costs, thus hiding “technical debt” until it is revealed later.

5.3 Importance of attribution context

The question posed to interviewees was: How important is each element of method use context to method attribution? A general sentiment expressed was that attribution context seemed important, but it was difficult to expound on. In general, additional method user feedback is needed in this area.

5.3.1 Design Task

The nature of the design task was rated as highly to extremely important to how a method is characterized by attribute.

5.3.2 Product

There was a disagreement on whether the nature of the product was of low or high importance to method attribution. Even the camp that deemed it to be highly important believed that methods should actually be generic with respect to the specific product at hand.

5.3.3 Nature of Method Enablers

The nature of the method enablers was rated as highly to extremely important to method attribution. Enabler availability can drive or constrain methods, as in the case of vendor lock-in. Design methods and tools are also sometimes tied to particular staff members – bringing designer A to the team implies adopting the methods and tools of designer A as well. One question is whether the design method should be selected before or after the design team is formulated. To justify pulling an experience designer onto your development program when multiple programs are competing for that designer's time, the expected performance and cost of various staff-method combinations can be evaluated. For example, the experienced designer using baseline methods and a less experienced designer using alternative methods could both achieve the same level of task deliverable quality but at higher cost for the less experienced designer using alternative methods.

5.3.4 Nominal Method Output Quality

A nominal method output quality target was considered to be of moderate to high importance to method attribution.

5.3.5 Current Stance towards Methods

The baseline method situation and prospects for alternative methods target were deemed to be of low importance to characterizing methods by attribute.

5.3.6 Nominal Method Implementation Cost

A nominal method implementation cost bogey was considered to be of moderate to high importance to method attribution.

5.4 Perspectives on an Attribution Scale

Designers were asked for their views on the use of an ordinal scale (low-medium-high) based on method use context to describe the states of method attributes such as output quality and method implementation cost. In general, they found the use of an ordinal scale that only provides sequence in a list to be sufficient; an interval scale that tries to quantify the difference between, say, 'medium' and 'high' would be more difficult to achieve. There may be benefit to increase the granularity of the scale from three to five segments (e.g. low, low-med, medium, med-high, high), but probably no more than that.

The interviewees found the ordinal scale chosen in the case study to be highly relevant to method attribution, highly to extremely easy to use, but only moderately effective owing to its coarse resolution and the difficult nature of the attribution problem itself.

Overall, design team members from the case study found the characterization and comparison of design methods by attributes to be effective for the method selection problem, but they warned about the difficulty of ensuring rigorous, repeatable method characterizations.

6 Conclusions and Recommendations

The outcome of pursuing the research objectives, and how the research objectives were accomplished, is now described. The research results are then used to answer the research questions. Finally, the chapter recommends actions to take based on the research findings, to improve product designs via design method selection support and to further research in this field.

6.1 Research Contributions

The empirical case study conducted on a recent aerospace industry vehicle development program addresses the two major research objectives in this thesis. First, this research developed a concept for operationalizing alternative design methods discovery, evaluation and selection for designers, and for completing the "value chain" by combining a design strategy model with that method selection framework. The present work:

- Created a model of the case study design process and methodology to serve as a common representation of the utilized design tasks and methods for stakeholder engagement and analysis, using primary sources, interviews, and graphical model representations for stakeholder engagement
- Demonstrated the discovery of alternative methods by analyzing a matrix representation of the design strategy model
- Synthesized an attribute-based method selection framework from the literature
- Demonstrated the characterization and comparison of methods using attributes

Secondly, this thesis collected designer and method user feedback as preliminary validation of the conceptual framework by engaging with design team members and leadership from the subject of the case study. These are the intellectual deliverables of the research.

6.2 Research Findings

In response to the primary research question posed, this thesis asserts that an attribute-based approach is effective in supporting aerospace conceptual design method selection. Differencing the attribute states of multiple methods provides a basis for designers to compare method alternatives against the context and needs of the design situation. A subset of design method attributes from a more generic listing has been found to be relevant to the method selection problem: Method Inputs, Method Outputs, Method-Supplied Quality of Outputs, and Method Implementation Costs. Specific elements of method use context have been shown to drive method attribution and should be accounted for. Compatible method candidates are discovered by relating design process activities to design methods according to the inputs, outputs and basic activities they share, thus generating various alternative assignments of methods to design tasks. And finally, the act of modeling and its deliverables provide a common language and visible understanding of a design team's strategy – its design process and methodology and the relationships between them.

6.3 Recommendations for enacting design method selection support

Several recommendations are presented to assist with efforts to implement design method selection decision support for current and future aerospace development projects. These points are not meant to constitute a formal plan, but rather to suggest principles to promote the success of any such endeavor:

- Decide on the scope of the method selection support initiative, and then cross-check this conceptual framework with the operational workflow of the design groups within that scope to further validate and gain confidence in this approach across the relevant spectrum of stakeholders. Try especially to get the opinion and approval of designers that are nominally reluctant to opening up the design method space.
- Search for and learn from existing method support systems, e.g. Methodos (Franke et al., 2003) and CiDaD (Braun and Lindemann, 2003; Ponn and Lindemann, 2006).
- Consolidate existing design method databases within the enterprise before creating a new one.
- Use a standardized template for profiling each design method – more information available for query is better than less, but allow for hierarchies and collapsing/expanding sections to manage the information "barrage".
- When collecting and organizing design methods for the enterprise, search internally first by polling design groups to understand their existing methods and, more broadly, their design systems.
- Design organizations should internally peer review their method attribution "sheets", just as many peer review journal articles before publication.
- Successfully implementation of a framework to support design method trades will require involvement from several stakeholders: project managers, system architects/designers, discipline-specific designers, and method experts.

- Pursue the future work recommendations in section 6.4.

6.4 Future Work

Avenues of academic inquiry to advance the current study and further research in this field are now suggested. The first recommendation is to further develop the conceptual framework for design method selection presented in this thesis. One aspect of the framework that can be expounded on is the development of techniques for design method attribution. A rigorous method attribution process that generates repeatable results (given context) is needed. It is expected that clear definitions of attributes, attribute states, and elements of attribution context would be required.

A second future research topic is to explore approaches to actual implementation of a design method selection decision support system at the grassroots or program level. The necessary spectrum of system designers, method experts, and design strategy leadership to be engaged by the effort would need to be determined, as would the appropriate amount of engagement. Sections 7.2.2.3.5 and 7.4.2 in Hubka and Eder (1996) discuss ways to present knowledge about design practices that could be evaluated for application. Stetter and Lindemann (2005) begin the conversation on approaches for coordinating and synchronizing any implementation endeavor with existing terminology and any similar initiatives in the organization. Prior method support system development efforts can also serve as inspiration (Franke et al., 2003; Braun and Lindemann, 2003; Ponn and Lindemann, 2006).

Another research avenue involves applying this research to additional case studies, to see whether alternative methods and choices were available at the time and how method selection was conducted. Such studies can gather additional data on the gap between design research and design practice, the impact of inadequate design method selection, and details on what designers need from a method selection support system. Applying this framework to representative aerospace design projects can further demonstrate its viability and validate its utility. Possible metrics include the framework's ability to structure methodology and promote the use of structured methods. Types of aerospace case study subjects include:

- Large development programs in their early design stages
- New variants of production programs
- Development programs that require "course corrections" during later design stages and more overt methodology changes
- One design process applied to multiple development projects
- Lower level design aspects, either physical (e.g. airframe) or functional () segments, rather than system level design

An extension of the research in this thesis would examine forecasting impacts on project performance (e.g. duration, cost, quality) based on design methodology choices. One approach is to integrate project performance models with the method selection framework's multi-domain matrix analysis (section 4.4) and design method attributes (section 4.5). If project performance outcomes can be predicted from a given project system "design" or configuration, then project performance models could be used to forecast results from different choices of methods. The performance forecasts would likely (at least initially) be described relative to a baseline or as a "delta", where the baseline would be the methodology status quo and the "delta" would be the impact from switching to an alternative methodology. The sensitivity of model-predicted project performance to alternative methodologies could then be explored. The intent would be to establish analysis capabilities before programming some kind of methodology selection or "optimization" scheme, all in a manner that is transparent to the user so that the performance model predictions support the user's reasoning and supplement the qualitative method attribute comparisons. An existing System Dynamics model of airplane development project performance from Asfour (2015) could be considered for this avenue of inquiry. If the Asfour (2015) model of a small Boeing engineering team's performance could be tailored to the case study in this thesis, then input data for that model could potentially be derived from a methodology choice consisting of a set of design method assignments. Possible project dynamics model inputs in this context include task durations, probability of creating rework and probability of discovering rework, all of which could change based on method selection.

7 Appendix

7.1 Design Strategy Modeling Interview Process and Questions

1. Verify design process (activities + artifacts)
    a. (Offer model options)
    b. Which option is most accurate?
    c. Any missing inputs/outputs, steps, iterations?
2. Understand design method assignments to design activities
    a. Which design activities/tasks should we discuss?
    b. (Clarify methods vs. tools)
    c. What method was used for this activity?
    d. Define the particular method used.
    e. How was the method operated? What is needed to operate it?
    f. What's involved to make that method work?
3. (Develop unstructured method descriptions)
    a. Elicit the designer or lead IPT for each task employing an unstructured method
    b. Elicit steps taken for each task to induce a method definition
        i. Ask what steps were taken for an activity, then induce an unstructured method from the procedure used
    c. Suggest informal, unstructured methods from literature
    d. Sometimes eliciting the design tool can induce what the design method was
4. Understand reasons for method assignments
    a. Elicit method strengths/weaknesses, performance, cost-effectiveness
    b. Characterize methods with more attributes
    c. Why was that method used?

    d. Was method selected first (or was method a constraint), then the design process fell out?
    e. How did method choices evolve over time?
    f. How did the method work out?
    g. What are key method attributes?
5. Explore other possible methods
    a. Elicit effective means for comparing methods
    b. Could this method be used for other design activities?

8 Bibliography

Andreasen, M., Hansen, C., and Cash, P. (2015), *Conceptual Design: Interpretations, Mindset and Models*, Springer, Switzerland.

Arena, M. V., Younossi, O., Brancato, K., Blickstein, I., and Grammich, C. A. (2008), "Why Has the Cost of Fixed-Wing Aircraft Risen? A Macroscopic Examination of the Trends in U.S. Military Aircraft Costs over the Past Several Decades," Report No. MG696, RAND Corporation.

Asfour, A. (2015), "Measuring Engineering Quality in Airplane Development," S.M. Thesis, Department of Aeronautics and Astronautics, Massachusetts Institute of Technology, Cambridge, MA (USA).

Augustine, N. R. (1986), *Augustine's Laws*, Viking Penguin, New York, NY.

Baker, P. J., and Whalen, J. T. (2012), "Survey of Trade Study Methods for Practical Decision-Making," NASA Independent Verification & Validation (IV&V) Workshop, https://www.nasa.gov/sites/default/files/survey_of_trade_study_methods_-_baker.pdf

Bevan, N. (2009), "Criteria for Selecting Methods in User Centred Design," from the International Workshop on the Interplay between Usability Evaluation and Software Development (I-USED).

Birkhofer, H., Kloberdanz, H., Berger, B., and Sauer, T. (2002), "Cleaning Up Design Methods – Describing Methods Completely and Standardised," *DESIGN 2002*, International Design Conference.

Braun, T., and Lindemann, U. (2003), "Supporting the Selection, Adaptation and Application of Methods in Product Development," International Conference on Engineering Design.

Browning, T., Fricke, E., and Negele, H. (2006), "Key Concepts in Modeling Product Development Processes," *Systems Engineering*, Wiley Periodicals, Inc., Vol. 9, No. 2, 2006.

Collopy, P., and Hollingsworth, P. (2009), "Value-Driven Design," *9th AIAA Aviation Technology, Integration, and Operations Conference*, American Institute of Aeronautics and Astronautics, AIAA 2009-7099.

Committee on Pre-Milestone A Systems Engineering (2008), "Pre-Milestone A and Early-Phase Systems Engineering: A Retrospective Review and Benefits for Future Air Force Systems Acquisition," National Research Council, National Academy of Sciences, Washington, D.C.

Crawley, E., Cameron, B., and Selva, D. (2016), *System Architecture: Strategy and Product Development for Complex Systems*, Pearson, Hoboken, NJ.

Cross, N. (2008), *Engineering Design Methods: Strategies for Product Design, Fourth Edition*, John Wiley & Sons, Ltd, Chichester, England.

Cross, N. (1993), "A History of Design Methodology," *Design Methodology and Relationships with Science*, Kluwer Academic Publishers, pp. 15-27.

Danilovic, M., and Browning, T. (2007), "Managing complex product development projects with design structure matrices and domain mapping matrices," *International Journal of Project Management 25*, Elsevier Ltd and IPMA, pp. 300-314.

De Weck, O. L., Roos, D., and Magee, C. (2011), *Engineering Systems: Meeting Human Needs in a Complex Technological World*, The MIT Press, Cambridge, MA (USA).

Defense Advanced Research Projects Agency (DARPA) (2009), "Broad Agency Announcement: META," DARPA-BAA-10-21.

Dori, D. (2011), *Object-Process Methodology: A Holistic Systems Paradigm*, Springer, Berlin.

Deloitte (2009), "Can we afford our own future? Why A&D programs are late and over-budget – and what can be done to fix the problem," Aerospace & Defense, Deloitte Development LLC.

Eppinger, S., Whitney, D., Smith, R. P., and Gebala, D. (1989), "Organizing the Tasks in Complex Design Projects," MIT Leaders for Manufacturing Program, WP# 3083-89-MS

Ernzer, M., and Birkhofer, H. (2002), "Selecting Methods for Life Cycle Design Based on the Needs of a Company," *DESIGN 2002*, International Design Conference, Ecodesign Workshop.

Eskew, H. (2000), "Aircraft Cost Growth and Development Program Length: Some Augustinian Propositions Revisited," *Acquisition Review Quarterly*, U.S. Defense Acquisition University, Summer 2000.

Franke, H.-J., and Deimel, M. (2004), "Selecting and Combining Methods for Complex Problem Solving Within the Design Process," *DESIGN 2004*, International Design Conference.

Franke, H.-J., Löffler, S., and Deimel, M. (2003), "The Database 'Methodos' Assists an Effective Application of Design Methods," International Conference on Engineering Design.

Fuge, M., Peters, B., and Agogino, A. (2014), "Machine Learning Algorithms for Recommending Design Methods," *Journal of Mechanical Design*, American Society of Mechanical Engineers, Vol. 136, October 2014.

Furuhjelm, J., Segertoft, J., Justice, J., and Sutherland, J.J. (2017), "Owning the Sky with Agile: Building a Jet Fighter Faster, Cheaper, Better with Scrum," https://www.scruminc.com/wp-content/uploads/2017/02/Release-version_Owning-the-Sky-with-Agile.pdf

Geis, C., Bierhals, R., Schuster, I., Badke-Schaub, P., and Birkhofer, H. (2008), "Methods in Practice – A Study on Requirements for Development and Transfer of Design Methods," *DESIGN 2008*, International Design Conference.

Gericke, K., Roschuni, C., and Kramer, J. (2015), "Discovery and Evaluation of Design Methods in Practice: An Empirical Study," International Design Engineering Technical Conferences & Computers and Information in Engineering Conference, *IDETC/CIE 2015*, American Society of Mechanical Engineers, DETC2015-47387.

Gero, J., Jiang, H., and Williams C. (2013), "Design cognition differences when using unstructured, partially structured and structured concept generation creativity techniques," International Journal of Design Creativity and Innovation 1.4, pp. 196-214.

Hubka, V., and Eder, W. E. (1996), *Design Science: Introduction to the Needs, Scope and Organization of Engineering Design Knowledge*, Springer, London, England.

International Council on Systems Engineering (INCOSE) (2011), "Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities," INCOSE-TP-2003-002-03.2.2.

Jones, J. C. (1992), *Design Methods, Second Edition*, John Wiley & Sons, Inc., New York, NY.

Krishnan, V. (1993), "Design process improvement: sequencing and overlapping activities in product development," Sc.D. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA (USA).

Lahonde, N., Omhover, J.-F., and Aoussat, A. (2010), "Designers Needs Analysis for Assisting the Selection of Design Methods," IDMME – Virtual Concept 2010.

Lyneis, J., Cooper, K., and Els, S. (2001), "Strategic management of complex projects: a case study using system dynamics," *System Dynamics Review*, Vol. 17, No. 3, pp. 237-260.

Menzel, C., and Mayer, R. (2006), "The IDEF Family of Languages," *Handbook on Architectures of Information Systems, 2nd edition*, Springer, Berlin.

Pahl, G., Beitz, W., Feldhusen, J., Grote, K.-H., Wallace, K., and Blessing, L. (2007), *Engineering Design: A Systematic Approach, Third Edition*, Springer, London, England.

Ponn, J., and Lindemann, U. (2006), "CIDAD – A Method Portal for Product Development," *DESIGN 2006*, International Design Conference.

Pugh, S. (1991), *Total Design: Integrated Methods for Successful Product Engineering*, Addison-Wesley Publishers Ltd, Wokingham, England.

Raymer, D. (1992), *Aircraft Design: A Conceptual Approach*, American Institute of Aeronautics and Astronautics, Inc., Washington, D.C.

Schmidt, E. (2015), "An Approach for Structuring Methods in Product Development Processes and the Role of Uncertainties," S.M. Thesis, Department of Mechanical Engineering, Massachusetts Institute of Technology, Cambridge, MA (USA).

Stetter, R., and Lindemann, U. (2005), "The Transfer of Methods into Industry," *Design Process Improvement: A Review of Current Practice*, Springer, London, England.

Ulrich, K. (2011), *Design: Creation of Artifacts in Society*, University of Pennsylvania, http://www.ulrichbook.org/

Ulrich, K., and Eppinger, S. (2012), *Product Design and Development, 5th Edition*, McGraw-Hill, New York, NY.

Unger, D. (2003), "Product Development Process Design: Improving Development Response to Market, Technical, and Regulatory Risks," Ph.D. Thesis, Engineering Systems Division, Massachusetts Institute of Technology, Cambridge, MA.

Wynn, D., and Clarkson, J. (2005), "Models of Designing," *Design Process Improvement: A Review of Current Practice*, Springer, London, England.