Investigating Emergent Design Failures Using a Knowledge-Action-Decision Framework

by

Colin P. F. Shields

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Naval Architecture and Marine Engineering)
in The University of Michigan
2017

Colin Patrick Fleishman Shields

colinsh@umich.edu

ORCID iD: 0000-0001-8916-7250

To my parents, for teaching perseverence and work-ethic

and

to my wife, for her constant support.

ACKNOWLEDGEMENTS

I received incredible support while completing my Ph.D. As a token of my appreciation, I would like to specifically recognize:

My advisor, David Singer, for pushing me to improve and grow, for guiding me personally, professionally, and academically, and for his support and encouragement.

My committee members, for their time, expertise, and mentorship. They were invaluable in helping me reach this point.

My office mates, for their friendship and support. Also, for their help brainstorming new ideas, solving old ideas, and writing it all down.

The NICOP team for teaching me how to think about problems in new ways.

Kelly Cooper at the Office of Naval Research and the Department of Defense through the National Defense Science & Engineering Graduate Fellowship (NDSEG) Program, for providing funding.

My family, who are always loving, supportive, and interested.

My wife, Ali, for her love, encouragement, and help writing and brainstorming.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

LCS	Littoral Combat Ship
K-A-D	Knowledge-Action-Decision
TSP	Traveling Salesman Problem
ACO	Ant Colony Optimization
MLR	Most Likely Route
L-PAT	Logical-Physical Architecture Translation

ABSTRACT

Investigating Emergent Design Failures Using a Knowledge-Action-Decision Framework

by

Colin P. F. Shields

Chair: David J. Singer

Cost growth and schedule delays frequently impact the design of naval vessels as well as other physically-large and complex products. In vessel design, many instances of cost growth, performance loss, and delays occur suddenly and are unexpected, causing significant damage to an acquisition program. These outcomes are not caused by physical product failure. Instead, they are symptoms of design failures and are characterized by emergent rework, inability to integrate disparate information, and unexpected design difficulty.

Design failures emerge from learning and decision-making that occurs during a design activity. In naval acquisitions, failures are frequently caused by the complexity of the naval design activity. Design approaches, processes, methods, and tools attempt to manage this complexity, but no methods exist to address it directly. To address the root-cause of design failures, new capabilities are needed to understand the knowledge-based relationships that drive their emergence.

This thesis proposes a Knowledge-Action-Decision (K-A-D) Framework that enables a knowledge-centric perspective of design. The K-A-D Framework decomposes design into an endogenous knowledge structure that describes the changing relationships between the ideas, concepts, and evidence designers use to progress a design activity. Analysis of the K-A-D Framework through a network representation allows the importance of knowledge structure elements to be measured, the trajectory of a design activity to be identified, and opportunities to prevent design failures to be found.

The K-A-D Framework, its network representation, and network analysis provide insight into the complex design dynamics that create unexpected design outcomes. First, results and methods from design science and the study of path dependence are synthesized to create the theoretical basis of the K-A-D Framework. The framework is constructed to capture the temporal relationships in knowledge structure development that can ingrain and lock-in design outcomes through path dependence. Second, the framework is applied to investigate the complex behaviors of an agent designing a solution to the Traveling Salesman Problem. Using network analysis of the K-A-D Framework, the agent's solution development and decision-making behaviors are identified and, in some cases, predicted. Third, the knowledge-centric perspective is applied to a naval distributed system design scenario by using the framework to define an approximate knowledge structure for distributed system integration. Finally, ensemble analysis is used to investigate the potential distributed system design knowledge structure characteristics under different early-stage design conditions. Examining the knowledge structure characteristics identifies design conditions that have elevated risk for late-stage design failures. Analysis of path dependence in the knowledge structure development identifies conditions where the risk of failure may be reduced by decisions made by the designer during the distributed system design activity.

CHAPTER I

Introduction

Naval vessel design requires an integration of disciplines ranging from structures to cyber security and is spread across a sprawling body of vendors, contractors, and government entities. Successful design requires that the needs of all constituents and disciplines are balanced within a set of requirements and satisfactory vessel specification. While the scope of this is immense, the difficulty of the naval design activity is further convoluted by its 'wickedness' (*Andrews*, 2011). This means that defining achievable and affordable requirements can only be done by exploring solution concepts, which in turn can lead to requirement changes and a re-interpretation of the problem itself (*Rittel and Melvin*, 1973). In practice, the scope of the naval design activity and its wicked nature make on-budget and on-schedule design completion elusive. Engineering and design issues cause significant cost growth and schedule delays in U.S. Navy vessel procurement (*Under Secretary of Defense Acquisition Technology and Logistics (US[AT&L])*, 2013, 2014). Table 1.1 shows a number of examples.

Naval design cost growth and delays are not caused by failures of the physical vessel; they occur in the overarching design activity and arise unexpectedly during design and integration. Their root cause is emergent design failure. Design failures, in general, are characterized by excessive rework, the inability to integrate the design, and increasing design effort (*Braha and Bar-Yam*, 2007). The design failures in Table 1.1

Ship	Initial budget ($M)	Total cost growth ($M)	Cost growth as a percent of initial budget (%)
CVN 77	$4,975$	847	17
LCS 1-2	472	603	128
LHD 8	$1,893$	303	16
LPD 18-23	$6,194$	$1,548$	25

Table 1.1: Cost growth in program budgets for ships under construction in fiscal year 2007 (*Government Accountability Office*, 2007).

emerged from the history of development, integration, and decision-making preceding their failure. While some design failures are non-emergent, caused by significant events that would be expected to cause failure, emergent failures are unexpected, but potentially preventable. To understand emergent design failures and what can be done to prevent them, the underlying interactions and relationships of knowledge, decisions, and actions that produce them must be understood.

In this dissertation, formal design theories and design cognition research are synthesized into a framework for tracking how design knowledge is created and used to make design decisions through time. The framework enables complex design behaviors to be identified and measured as they occur in a design activity. The goal of this dissertation is to use the proposed framework and analysis to provide new insights into how emergent design failures arise and what designers can do to prevent them.

1.1 Background and Motivation

The motivation for this dissertation arose from discussions with Robert Keane, former U.S. Navy Chief Naval Architect, about unexpected emergent cost and work content growth in vessel design. Keane has produced a large body of work on U.S. Navy cost drivers and costing improvements (*Keane*, 2011; *Keane et al.*, 2015, 2016, 2017). This work and his experiences have repeatedly identified vessel complexity, characterized by the density of shipboard systems, as a reliable predictor of increased design, engineering, and production work content and cost *Keane* (2011).

Vessel complexity has driven cost growth for decades. *Arena et al.* (2006)'s analysis of naval acquisition cost growth, shown in Table 1.2, found that increasing vessel, system, and mission complexity has significant contributions to annual cost escalation, near 20% for surface combatants (*Arena et al.*, 2006). This corroborates findings that vessel complexity causes long-term cost growth between vessel acquisitions programs and unexpected near-term cost growth (*First Marine International*, 2005).

Ship type	Annual escalation rate due to characteristic complexity (%)	Total annual growth rate (%)	Portion of annual growth rate due to complexity (%)
Surface combatants	2.1	10.7	19.6
Attack submarines	1.6	9.8	16.3
Amphibious ships	1.7	10.7	15.9

Table 1.2: Contribution to annual cost escalation rate by characteristic complexity, 1950-2000 (*Arena et al.*, 2006).

Nearly half the cost of naval vessel is spent on its installed systems (*Miroyannis*, 2006) and these systems are significant sources of ship complexity (*Keane et al.*, 2017). The systems installed in a vessel must be designed individually and then integrated within a functional vessel design. The difficulty of this design and integration frequently leads to extreme cost growth (*Government Accountability Office*, 2007).

Design difficulty and failures in system design stem from the interaction between systems within the integrated vessel design (*Rigterink*, 2014). Changing the configuration or requirements of one system can have ramifications across the many potential interacting systems. The effect is that changing system design or replacing equipment within the system is costly and difficult (*Schank et al.*, 2009).

Typically, the cost implications of system integration are captured in vessel complexity measures. Figures 1.1 shows the relationship between outfit density, a complexity characterization, and ship production hours (*Keane*, 2011). Figure 1.2 shows compensated gross tonnage coefficient, which is based in complexity metrics, for vessel cost approximations (*Craggs et al.*, 2004). The take away of both figures is that as the systems in the vessel becomes more tightly packed the cost increases non-linearly.

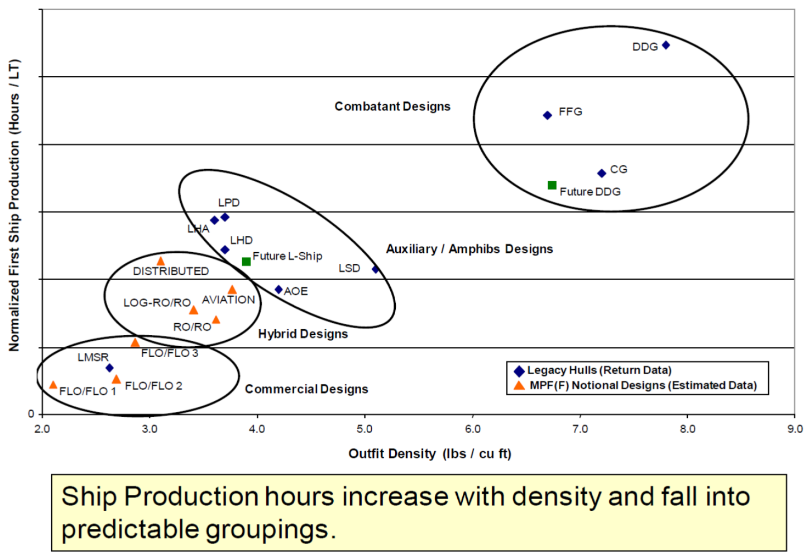


Figure 1.1: Relationship between outfit density and ship production hours (*Keane*, 2011).

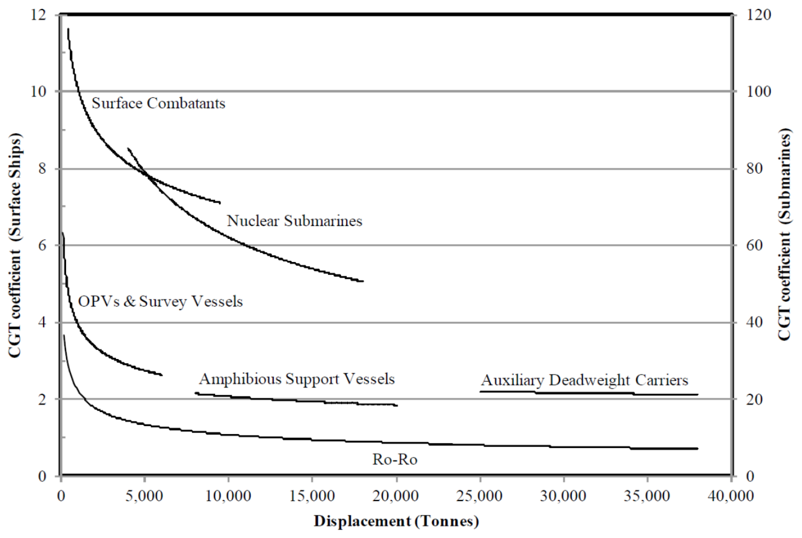


Figure 1.2: Compensated gross tonnage coefficient by ship type (*Craggs et al.*, 2004).

The complexity measures used to estimate cost implications quantify the interactions between systems within the vessel. The relationship between cost growth and these complexity measures are based on the product that is designed (a vessel in naval design). They are a surrogate for the underlying causes of knowledge-based design failures. High levels of system interaction require more designer knowledge to understand the integrated system. As the number of system interactions grow, so does the knowledge requirement on the designer. The resulting design failure mode, explosive growth in the required design effort, emerges from this relationship.

The unwanted effects of complexity are not limited to vessel design. Table 1.3 shows Bar-Yam's documentation of a number of large-scale system engineering and design failures. *Bar-Yam* (2003a) identifies that current engineering practices break-down in the face of complex system design. Specifically, designing a complex system becomes exponentially difficult as the system environment and desired system behaviors increases in complexity (*Bar-Yam*, 2003b). In essence, the design of complex systems fails because large complex systems are hard to understand. From this perspective, designing a complex system is a complex task and complex tasks can only be accomplished by another complex system (*Bar-Yam*, 2003b). Because of this, design can be understood and represented as a complex system itself.

If design is a complex system, what is it a system of, and what is a design failure? *Mavris and DeLaurentis* (2000) provide insight into the former by defining and representing design as a relationship between designer knowledge and freedom, see Figure 1.3. *Gillespie* (2012) and *Parker* (2014) identified that the knowledge-freedom relationship is complementary. Over time, designer actions generate knowledge which can be used to make decisions about the product specification. Decisions reduce the set of potential products, limiting freedom, but facilitating more targeted knowledge generation. This breaks the design complex system into interdependent relationships between knowledge, decisions, and the actions that create and use them. Thus, design failure occurs when this system is unable to converge on a fully defined and understood product.

Like all complex systems, understanding the knowledge-action-decision design decomposition must be approached from the bottom-up. This requires that each element

System function - organization	Years of work (outcome)	Approximate cost \$M=Million, \$B=Billion
Vehicle Registration, Drivers license - California DMV	1987-1994 (scrapped)	\$44M
Automated reservations, ticketing, flight scheduling, fuel delivery, kitchens, and general administration - United Air Lines	Late 1960s - Early 1970s (scrapped)	\$177M
State wide Automated Child Support System - California	1991-1997 (scrapped)	\$110M
Hotel reservations and flights - Hilton, Marriott, Budget, American Airlines	1988-1992 (scrapped)	\$125M
Advanced Logistics System - Air Force	1968-1975 (scrapped)	\$250M
Taurus Share trading system - British Stock Exchange	1990-1993 (scrapped)	\$100 – \$600M
IRS Tax Systems Modernization projects	1989-1997 (scrapped)	\$4B
FAA Advanced Automation System	1982-1994 (scrapped)	\$3 – \$6B
London Ambulance Service Computer Aided Dispatch System	1991-1992 (scrapped)	\$2.5M, 20 lives

Table 1.3: List of large-scale engineering design failures (*Bar-Yam*, 2003b).

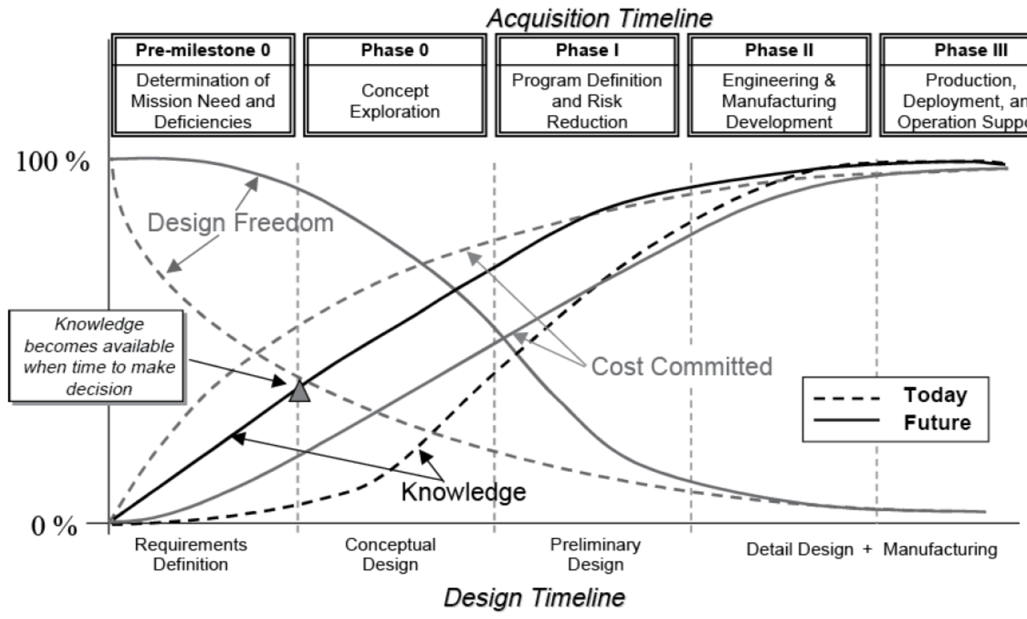


Figure 1.3: The relationship between design knowledge and design freedom through time (*Mavris and DeLaurentis*, 2000).

of the system and their potential interactions are defined and the resulting structures can be understood in the context of the macro-level design activity. However, in naval design, design knowledge is traditionally communicated through product models. To quote the *NAVSEA* (2012),

> As the design evolves from initial concept through construction, delivery, and, eventually, disposal, the type and amount of information stored in the product model is ever increasing. Specific design products are extracted from the product model in electronic format to document the design, trade studies, and analyses, and design decisions made throughout each design phase.

Product models are a reduction of the final product. The traditional approach is to capture generated design knowledge in these reductions and then use the reductions to make decisions about the final product. This approach is product-centric.

To understand if the knowledge underpinning the product models will cause emergent design failures, a knowledge-centric approach is required. How knowledge is developed and used needs to be tracked through the entire design activity. This can be accomplished with the knowledge-action-decision design decomposition. Once the system is defined and its behaviors better understood, it can be applied to naval design problems.

The knowledge-centric design perspective can provide new insights into why ship complexity causes design failures and how these failures can be addressed. The goal of this dissertation is to introduce the knowledge-based complex system design perspective, provide the theory to define the system, develop an actionable mathematical foundation, and demonstrate the theory on canonical and naval design problems.

1.2 Current Research

The main challenge of this research was to create a new way to represent design activities, analyze design activity behaviors, and predict how those behaviors could lead to emergent design failures. The created methods had to capture the complex system underlying design activities in a way that could be applied to general and naval specific cases. This research attempts to develop a method and supporting theories similar to analysis used in studies of design cognition, but applicable to large-scale and long-term design and engineering efforts. The goal is to decompose design activities into three interrelated types of elements, without necessitating the detailed classification of elements or a fixed scale of analysis. This allows the method to be used in the general study of design and provide new insight into the characteristics and behaviors of complex system design. Such a method would enable designers to identify potential causes of costly design failures in early-stage design and take action to mitigate the associated risks.

1.2.1 Scope

Three broad problems have been identified:

- Emergent and costly design failures arise from the act of designing a product, not from the product's attributes.
- Avoiding emergent design failures requires that knowledge-based design complexity is understood and analyzed; it is insufficient to only consider the product.
- Complexity in shipboard system design and integration is the proximate cause for extreme, unexpected late-stage cost and work content growth. Preventing these outcomes depends on the the knowledge-centric design perspective and requires a new analysis perspective.

These problems can be restated as three research questions:

1. Can an objective characterization of the design activities that cause or increase the chance of emergent design failures be developed?
2. Can knowledge-based design complexity be represented, understood, and used?
3. Can complex design activity dynamics provide new insight into emergent naval design failures?

The scope of this thesis is to address these questions by introducing a knowledge-based design perspective rooted in theories of design science and cognition, and facilitated by the study of complex systems. Addressing these questions represents a paradigm shift away from the traditional product-centric design perspective that is reliant on product models to a knowledge-based perspective. This shift requires a fundamental re-framing of the naval design activity as well as the supporting theory and analysis to make it actionable.

1.3 Contributions

This thesis expands the scope of naval design considerations and analysis from the product-centric perspective to a more encompassing knowledge-centric perspective. It is the author's belief that the expanded view allows designers to more deeply understand the causes of design outcomes and how they can affect them. That is, emergent design outcomes are a function of product information within the context of the designer's body of knowledge. As information is used and understood by designers, it becomes knowledge that expands the body of knowledge. Thus design can be leveraged towards better outcomes by examining the larger knowledge-based system. The primary contribution of this thesis is a knowledge-based design framework to represent the design activity and ground it in design science and cognition. Supporting contributions include methods for modeling the knowledge-based design decomposition, verification of the framework and model, and analysis tools for measuring design activity behavior in general and naval-domain instances. The specific contributions addressing the research question are:

1. Can an objective characterization of the design activities that cause or increase the chance of emergent design failures be developed?

    (a) Introduction of a knowledge-centric design perspective and corresponding knowledge-based description of design activities.

    (b) Recognition that the temporal emergence of interdependencies within the knowledge-based system, called knowledge structures, control design outcomes through path dependence.

    (c) Redefined design failure in terms of knowledge structure characteristics and growth.

    (d) Classification of the factors influencing knowledge structure development.

2. Can knowledge-based design complexity be represented, understood, and used?
    (a) Formulated Knowledge-Action-Decision Framework to design activities.
        i. Defined designer knowledge, action, and decisions as the fundamental interdependent elements of knowledge structure development.
        ii. Structured the definitions of knowledge, action, and decisions so that they could be studied temporally to elicit emergent behaviors in the design activity.
    (b) Demonstrated analysis capabilities of the Knowledge-Action-Decision Framework on a simulated design problem.
        i. Defined a network representation of the Knowledge-Action-Decision Framework.
        ii. Applied the network representation to track simulated design activities.
        iii. Introduced interpretations of network analysis methods to measure complex system behaviors in design knowledge structures.
        iv. Demonstrated that network-based knowledge structure analysis can identify emergent designer behavior through time, as modeled in the design simulation.
3. Can complex design activity dynamics provide new insight into emergent naval design failures?
    (a) Re-framed early-stage ship system design as an ensemble analysis of possible late-stage system configuration network representations to identify how early-stage decisions influence late-stage product characteristics.
        i. Expanded network-based models of a vessel physical architecture, logical architecture, and physical system configuration.

   ii. Created methods to generate ensembles of physical system configurations in early-stage ship design.
   iii. Formulated system density analysis to create leading indicators of late-stage design trade-offs between physical system configuration and vessel layout.

(b) Structured ship system design and integration as a design problem within the Knowledge-Action-Decision Framework.
   i. Identified the knowledge structure characteristics developed during distributed system design using the law of functional complexity.
   ii. Created analysis methods to convert distributed system network representations into knowledge structure characteristics.
   iii. Measured the probabilistic characteristics of system design knowledge

(c) Measured the potential for emergent design failures in distributed system design activities and identified opportunities for designers to prevent them.
   i. Identified the potential for late-stage design failures by applying ensemble knowledge structure analysis to relate early-stage decisions to increased knowledge generation and unpredictable knowledge structures.
   ii. Demonstrated that product-centric analysis cannot elicit the potential for design failures and that knowledge-centric analysis is required.
   iii. Measured the drivers of knowledge structure characteristics to identify opportunities for designers to avoid design failures through their actions.

1.4 Overview of Dissertation

This dissertation is divided into 8 chapters in 2 core sections. The first section, Chapters 2-5, provide the theoretical background to represent design activities, analyze design activity behaviors, and predict how those behaviors could lead to emergent design failures. The second section, Chapters 5-7, illustrates how the proposed approach can help prevent emergent failures in naval design. These chapters detail the application of the developed theory to a naval distributed systems design example.

- Chapter II introduces the knowledge-centric perspective of design activities and differentiates it from the product-centric perspective. The chapter demonstrates how a knowledge-centric perspective can provide new insights into emergent design failures and identifies the work needed to make the perspective actionable.

- Chapter III presents a theoretical framework for modeling the knowledge-centric design perspective using knowledge structure development. Knowledge structure development is decomposed into three types of interrelated elements: knowledge, actions, and decisions. It is proposed that the growing set of relationships between these elements captures how path dependence can influence design outcomes and how design failures can arise during a design activity.

- Chapter IV defines a mathematical representation of knowledge structure growth during a design activity, called the Knowledge-Action-Decision (K-A-D) Framework. A multilayer network representation of the K-A-D Framework is introduced along with a number of analysis methods. The mathematical model, network representation, and analyses are demonstrated on a small design activity.

- Chapter V applies the K-A-D Framework to an agent-based designer solving the Traveling Salesman Problem. Four computational experiments are used to model learning, decision-making, and path dependence in a design activity. Analysis of the agent's knowledge structure development shows that the modeled design behaviors can be measured and identified with the K-A-D Framework.

- Chapter VI explains the knowledge-centric perspective of naval distributed system design and details how and why it frequently causes emergent design failures. Then, the K-A-D Framework is applied to model and analyze knowledge-based design complexity in an early-stage naval distributed system design.
- Chapter VII uses the distributed system knowledge structure model to identify the potential for emergent design failure and provides new insight into how design failures can be prevented.
- Chapter VIII summarizes the body of work presented and offers suggestions for future research.

CHAPTER II

Naval Design, Knowledge-Based Complexity, and Emergent Design Failures

[1] Over the last 20 years, engineering and design issues have significantly increased cost growth and delays in U.S. Navy acquisition (*Under Secretary of Defense Acquisition Technology and Logistics (US[AT&L])*, 2013, 2014) and other large-scale system acquisitions (*Bar-Yam*, 2003b). These issues represent design failures which are characterized by unexpected excessive rework, inability to integrate a product, and increased design effort (*Braha and Bar-Yam*, 2007). Numerous studies have linked design failures to product complexity and have provided guidelines for avoiding them, for example (*Gaspar et al.*, 2012; *Keane et al.*, 2015, 2017). This is a product-centric perspective that focuses on the relationship between design outcomes and the product's constituent parts. While the product-centric view of complexity is instructive, design failures are based in the act of designing the product, not the product itself. Thus to better understand what causes design failures, the system of designing must be considered, not just the system being designed.

Fundamentally, designing is a learning and decision-making activity (*Gero*, 1990). Over time, designers generate knowledge to facilitate decisions about the product

[1]The following chapter is an adaptation of a submission to Naval Engineers Journal for publication under the title, *NAVAL DESIGN, KNOWLEDGE-BASED COMPLEXITY, AND EMERGENT DESIGN FAILURES.*

(*Hansen and Andreasen*, 2004). Together, these descriptions highlight that design creates a growing knowledge-based system. Design failures occur when the design activity enters unproductive cycles, becomes intractable, or requires too much effort to sustain. Design failure states are emergent, caused by the interdependent web of designer knowledge and their decisions, not by an overarching flaw in design practices. When decisions are made or new knowledge is created, it must be integrated into the existing designer knowledge. In extreme cases, integration causes design failures through incompatible knowledge and cascading decision changes or knowledge rework. Because of this, many root-causes of design failures can only be identified by understanding this knowledge-based complexity.

This chapter presents the theoretical basis for describing naval design as a knowledge-based system. Section 2.1 proposes a knowledge-based system model of design and provides example applications to naval design. Section 2.2 identifies emergence as the complex behavior behind design failures and Section 2.3 discusses how current design practices mitigate unwanted emergence. Section 2.4 presents conclusions and future work.

This chapter seeks to answer the following:

- What drives design failures in the design of complex naval systems?
- What strategies are available to mitigate and prevent design failures?
- How can these strategies be improved?

While the knowledge-centric perspective provides new insights into the research questions, further studies into the methods of implementing possible improvements will be needed. The research questions and their answers aim to advance the broader question of how the current U.S. Navy design practices can be augmented to avoid extreme, unexpected cost growth and delays.

2.1 Naval Design as a Knowledge-Based System

Theoretical models of design can help introduce a knowledge-centric perspective of naval design. *Mavris and DeLaurentis* (2000) describe the system acquisition process as a relationship between design knowledge and design freedom. Design knowledge is the information and ideas used by the designer. Design freedom is how much of the product is undefined, or conversely, how much of the product specification has been decided. Design knowledge and decision are created by designer actions that progress the design (*Oxman*, 1990).

Shown in Figure 2.1, which expands on the knowledge-freedom relationship, the design starts with limited knowledge which grows through learning. When decisions are made based on design knowledge, the freedom of potential design outcomes decreases. The resulting relationship is circular - gain knowledge, make a decision, repeat until convergence or failure (*Gillespie*, 2012; *Parker*, 2014). In formal theories of design, this relationship is described as a co-evolution of the problem and solution concept knowledge (*Dorst and Cross*, 2001; *Maher et al.*, 1996).

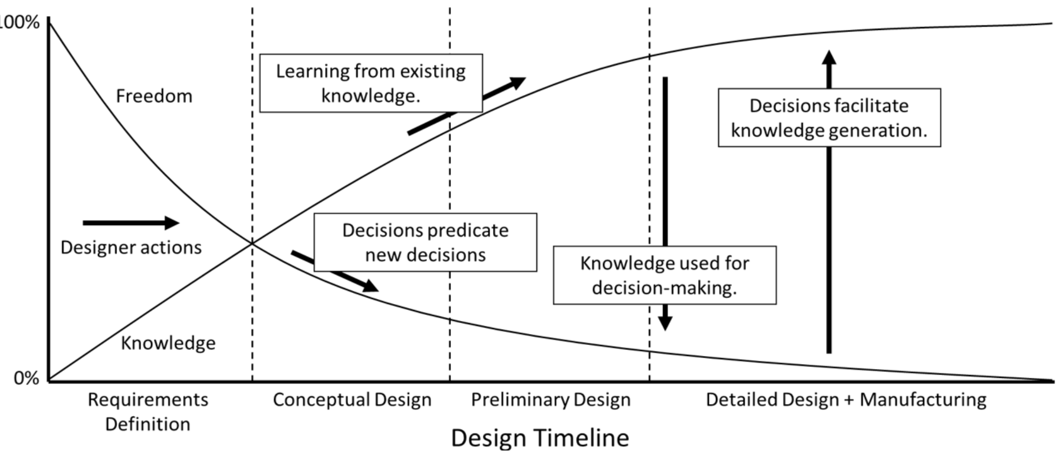

100%
0%
Freedom
Knowledge
Designer actions
Learning from existing knowledge.
Decisions predicate new decisions
Knowledge used for decision-making.
Decisions facilitate knowledge generation.
Requirements Definition
Conceptual Design
Preliminary Design
Detailed Design + Manufacturing
Design Timeline

Figure 2.1: Progression of design knowledge, freedom, and action. Knowledge grows through learning actions and is used for decision-making to specify parts of the solution. Decisions reduce the freedom of future concepts to facilitate further development and may also predicate additional decisions.

During the design activity, designers use their knowledge and solution concept ideas to learn about the design problem and support further design development (*Hatchuel and Weil*, 2003, 2009). The patterns of knowledge generation and usage create a set of relationships between ideas, evidence, and concept functions, called the knowledge structure (*Dong*, 2016).

Interdependencies within the knowledge structure create path dependencies towards certain solutions. A path dependent process is one where the order of events in the system change the probability of future outcomes (*Page*, 2006). In design, this means that the sequence of actions that design a technology will ultimately affect the defined problem and its solution (*Dosi*, 1982). As path dependencies grow, they create preferences toward concept ideas and the knowledge to support them (*Rycroft and Kash*, 2002).

This dissertation refers to cascading design changes that result from information, knowledge, and decisions as path dependent. In some cases, a single decision, piece of information, or introduction of a new technology could be classified as tipping points. This would be the case if the distribution over possible designs changed dramatically as opposed to the more gradual accumulated changes described in the thesis *Lamberson and Page* (2012). For example, the introduction of knowledge about a flawed reduction gear in the Freedom-class Littoral Combat Ship design would be an example of a tipping as it resulted in fundamental changes to the ships design, integration, and manufacturing plan.

The relationship between knowledge structure development and solution concepts can be thought of in terms of fitness landscapes. On a fitness landscape, the quality of each solution for a given problem is represented by its "height". To represent a design problem, each concept is evaluated against the designer's current problem understanding (which includes the current design requirements).

Easy problems have peaked landscapes, shown in Figure 2.2a, with a distinct

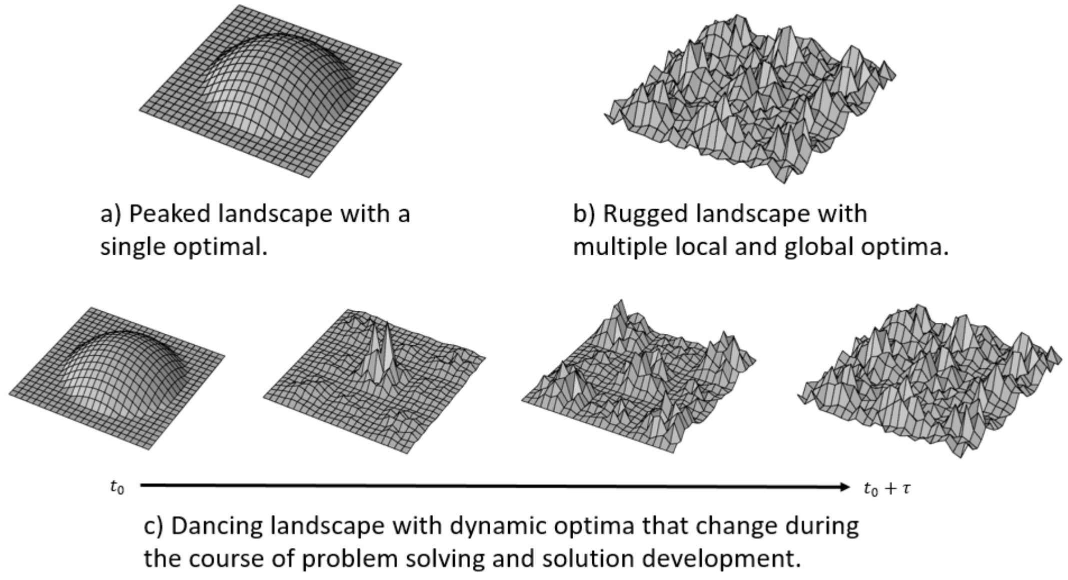

a) Peaked landscape with a single optimal.
b) Rugged landscape with multiple local and global optima.
t_0
$t_0 + \tau$
c) Dancing landscape with dynamic optima that change during the course of problem solving and solution development.

Figure 2.2: Classes of design problem landscapes. Each point on the landscape represents a solution concept and the point's height represents its fitness based on the designer's problem understanding (*Shields et al.*, 2016a).

optimal solution that is readily found. Hard problems have rugged landscapes, shown in Figure 2.2b, with many locally good solutions that obscure the true optimal (*Grim et al.*, 2013; *Page*, 2010).

The study of complex systems distinguishes complicated problems, that can be completely understood given sufficient time and resources, from complex problems that cannot. Complex problems can never be solved because they have dancing landscapes with dynamic optima (Figure 2.2c), which are effected by interacting, diverse variables that change over time (*Page*, 2008).

Nontrivial design, where problems are ill-defined and solutions are more than combinations of existing technologies are complex problems with dancing landscapes. Here, concept quality may change due to forces outside the designer's control, the way the problem is approached, and the sequence of actions used to approach it. These three factors - external, internal, and temporal - can be applied to naval design by considering events in a notional ship design process:

- External factors: Changes in the landscape due to forces outside of the designer's control. For instance, congressional budget changes or new mission requirements may alter a vessel's fitness by making it unaffordable or ill-suited for planned operations.

- Internal factors: Changes in the landscape due to the designer's understanding of the problem or decisions. This is the wicked problem effect; efforts to solve a problem may reveal, create other problems, or contradict existing decisions (*Andrews*, 2012). For instance, refining a concept may reveal a new technical challenge that was previously unknown, such as wet deck slamming in multi-hull vessels.

- Temporal factors: Changes in the landscape due to path dependencies. These are often created by the build up of internal and external factors over time. For instance, a promising high-voltage weapon system leads to all-electric ship development and expertise, which motivates further weapons systems development and exploration.

Unlike static problems, complex problems have unpredictable optima and cannot be solved in the traditional sense of locating a highest point. In the best possible case, the nature of the complexity can be understood and the actors, interactions, and processes that produce it can be harnessed (*Axelrod and Cohen*, 2000). During a design activity, designers try to guide knowledge-based dynamics to define a satisfactory problem and produce a solution. Thus, understanding these dynamics requires that the underlying knowledge-based complex system is investigated. To make this idea more concrete, the knowledge-centric design perspective is applied to two examples. These examples illustrate the general influence of the three factors of complexity on design activities.

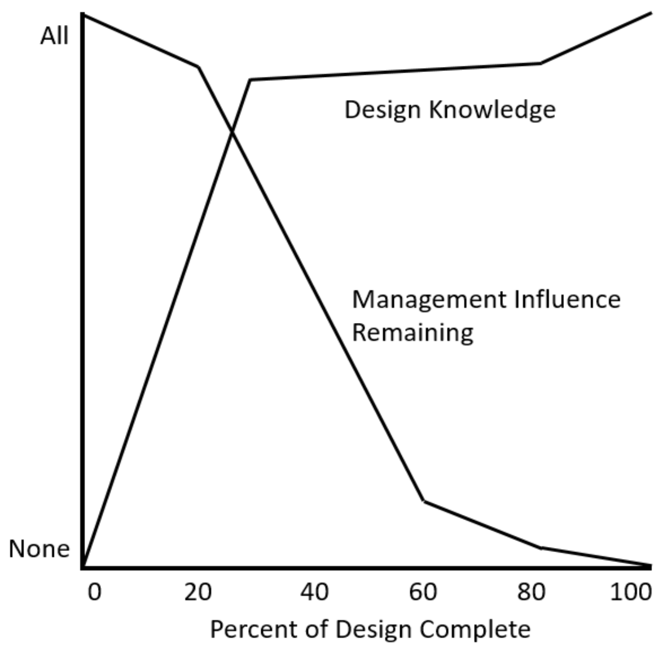


Figure 2.3: Relationship between design knowledge and management influence, adapted from (*Wheelwright and Clark*, 1992).

Temporal Complexity: Design Manager Influence

Knowledge generation, decision-making, and path dependencies explain the changing influence managers have on design outcomes. Figure 2.3 illustrates the phenomenon. At the outset of design, the manager leading the design effort has significant power to influence the design's trajectory. However, as the design progresses influence is reduced (*Wheelwright and Clark*, 1992). While this has been well studied from process and organizational standpoints, a knowledge-centric perspective can provide additional context.

When the knowledge supporting design is first created, the manager can guide what is investigated and control decision-making. Over time, the knowledge structure builds to support and refine a particular problem and corresponding concepts. The result is path dependencies towards a particular set of design outcomes. Reworking the knowledge structure requires revisiting a large body of interdependent knowledge and decisions.

This accounts for both the early manager control and later lack thereof. Guiding the creation of path dependencies that define the problem and solution concepts gives managers significant influence. Once the path dependencies are created, the influence is diminished and changing course becomes more difficult. The influence of path dependence, as well as the other factors influencing design, can be seen in the U.S. Navy's Littoral Combat Ship acquisition program.

Factors of Complexity in the Littoral Combat Ship Design Changes

Announced in 2001, the Littoral Combat Ship (LCS) was to be a speed and stealth oriented vessel as part of the US Navy's DD(X) surface combatant family (*Work*, 2014). LCS concept development continued from 2001 to 2003 and a number of key elements were defined. Among them was the belief that the LCS would rely on integration with the broader fleet architecture for its effectiveness and survivability (*Work*, 2014).

However, as vessel requirements developed there was a new emphasis placed on temporary aviation capabilities as well as reduced manning goals (*Work*, 2014). Additionally, the original survivability requirements were increased to make individual vessels more survivable as opposed to relying on the fleet (*United States Navy*, 2003).

While the above is a significantly abridged section from the LCS design history, internal, external, and temporal factors are apparent. Much of the concept evolution was internal, caused by a better understanding of how the LCS would be used. Some of the evolution was external, the defense community required more survivability. Finally, development was clearly temporal, building off of the initial concept instead of restarting at each change.

2.1.1 Complexity and Design Failures

Knowledge-based complexity can cause design failures when internal, external, or temporal factors lead to knowledge structure rework or the inability to integrate information. Broken down by factor, these failure modes are:

- External factors: New requirements or knowledge can cause successive knowledge generation. The resulting knowledge has to integrate with the existing knowledge structure which may cause integration difficulties and increased design effort.

- Internal factors: Changing design understanding and decision-making can invalidate previous knowledge or decisions. When large portions of the knowledge structure are effected it can cause significant delays and prevent design completion.

- Temporal factors: Path dependencies can lock-in undesirable or unstable solutions. Redirecting these path dependencies may cause design failures because rework has implications on previous design decisions. Additionally, changing course away from well-developed knowledge and concepts may require significant design effort.

Defining design as a knowledge-based complex system introduces a new way of looking at design and its failures. Using ideas from the broader study of complex systems, the three causal factors of complexity can be translated into a single core behavior, called emergence, that drives failures.

2.2 Emergent Design Failures

Miller and Page (2008) define a complex systems as a system with interactions between diverse actors with interdependent behaviors who adapt to one another.

Emergence describes how system-level properties can arise from these lower-level interactions.

Design outcomes and failures are caused by behaviors of a knowledge-based complex system. This implies that design failures are negative behaviors that arise from underlying interactions between knowledge, decisions and how they are used by the designer. The idea of emergence helps explain how those behaviors can be studied. Emergence describes the system as a result of its parts. This is described in Bedau's definition of weak emergence (*Bedau*, 1997):

> Macrostate, P, of system, S, with microdynamic, D, is weakly emergent if and only if P can be derived from D and S's external conditions but only by simulation.

Emergent properties are greater than the properties of single subsystems and result from interactions between system elements, people, and subsystems. In naval architecture, emergence is seen in many properties of the physical vessel, e.g. cascading system failures, but emergence in vessel design stems from designers and teams that work to create a product.

Emergence properties do not have governing equations, they must be composed from the interaction of multiple elements and simulated. The same logic applies to the knowledge-centric design perspective and its knowledge and decisions.

Design failures are strongly affected by the relationships of the information and tasks in a design activity (*Braha and Bar-Yam*, 2007). In some cases, design failures can be understood and addressed through the way knowledge is used and created during the design activity (*Braha and Reich*, 2003). For example, emergent design rework can be prevented by controlling knowledge generation (*Mihm and Loch*, 2006; *Smith and Eppinger*, 1997) and design innovation difficulties can be mitigated by managing the patterns of knowledge structures (*Dong*, 2016).

Preventing failures depends on the emergent knowledge structure and corresponding design landscape behavior. The design knowledge dynamics that drive emergent failures can only be observed through the underlying system. Because of this, emergent design behaviors and failures need to be identified through the knowledge-based system. This knowledge-centric perspective is in contrast to the traditional product-centric perspective. The product-centric perspective focuses on the level of solution definition that is defined within the context of an actual instantiation of the material solution. The following examples illustrate this point by comparing the product-centric and knowledge-centric perspectives of emergence.

Knowledge-Based Emergence: LCS Modifications

As a result of new requirements and budgetary pressures, the Navy announced plans to modify the LCS designs in 2014. The modified vessels were designated as frigates, with enhanced survivability and lethality. In many ways, this appears to a product-based change, the LCS vessel is slightly altered to a frigate. However, there has been a significant design effort required to generate the knowledge to understand how the new requirements impact the vessel design *United States Government Accountability Office* (2015). This is an example of knowledge-based emergence.

In 2010, the Navy planned to test two LCS variants: a trimaran designed by General Dynamics and a monohull designed by Lockheed Martin (*Work*, 2014). The most desirable variant would be contracted for extended production and the less desirable would be phased out of production (*Work*, 2014). From the product-centric perspective, the goal was to identify all emergent performance characteristics and select the best concept.

However, the more relevant emergent behavior was knowledge-based. Operating the two vessels showed designers that increased survivability and lethality were needed for both variants (*O'Rourke*, 2017b). This represents an internal factor of complex design behavior. When the design problem was better understood, the fitness of vessel concepts changed. The design outcome will depend on the integration of the LCS knowledge structure with the knowledge structure development that supports the modifications. This is far removed from the product-based outcome, which is dependent on the differences between the parts that compose the original LCS and its frigate modification.

Knowledge-Based Emergence: Design Instability

The Navy's CVN-78 design had a goal to integrate 16 new and untested technologies into its as-built design (*Government Accountability Office (GAO)*, 2009). These technologies were designed, built, and tested concurrently with the vessel design (*O'Rourke*, 2017c). This introduced significant risk into the vessel design and system integration, leading to considerable cost and schedule growth (*O'Rourke*, 2015).

The Navy's testimony indicated that the cost growth was driven by design requirements changes associated with the developing technologies (*O'Rourke*, 2017c). The negative effects of design changes are not limited to the CVN-78. They have been seen repeatedly in studies of over-budget and behind schedule acquisition programs (*Government Accountability Office (GAO)*, 2009).

From a product-centric perspective, design changes are relevant when they change the attributes of the material solution. From a knowledge-centric perspective, design changes can cause emergent behavior in the entire design activity. Knowledge structure growth means that initial ideas and evidence are used to create knowledge to facilitate future designer actions. This creates patterns of interdependent knowledge. If parts of the knowledge structure change, other parts must also change. The scope of this propagation can cause emergent design difficulty and rework.

Attempts to prevent emergent failures driven by design changes have been noticeably product-centric. During the CVN-79 design, there was a program-wide effort to complete 100% of the product models and 80% of initial engineering drawings by the time of construction contracting (*O'Rourke*, 2017a). This drive to reach 100% solution definition is product-centric and representative of the belief that design outcomes are controlled by the product that is created. However, due to the complexities of large-scale design, it is impractical to reach 100% product definition. Additionally, the product model cannot account for the knowledge and decision interdependencies that created it. This leaves design outcomes vulnerable to knowledge-based emergence.

Knowledge interdependencies and the internal factors of complexity mean that fixing a large portion of the product does not guarantee the corresponding knowledge is also fixed. New design knowledge may propagate across the knowledge structure and result in significant rework of the "fixed" product. Thus, without the capability to directly consider the knowledge-centric perspective, it is impossible to evaluate if the remaining 20% of the CVN-79 drawings have the potential to cause emergent design failures.

2.2.1 Linking Product and Knowledge-Based Emergence

Emergence illustrates why both product-centric and knowledge-centric perspectives must be used to ensure positive design outcomes. Product-based analysis provides valuable information about how alternatives will perform and their associated characteristics. Knowledge-based analysis can help designers understand the overall design activity. While not explicit, there has been a push in naval design to link the product to its knowledge-centric implications.

Measures of product and process complexity help to bridge the gap between the product- and knowledge-centric design perspectives. Keane has identified, through multiple studies (*Keane et al.*, 2015, 2016, 2017), the importance of minimal vessel (product) complexity in acquiring vessels at cost. He and his team identified that the issue is that complex products have substantial cost growth associated with their design and production. This insight is a surrogate for the underlying knowledge-based system.

Complexity measurements quantify the amount of information required to understand a system (*Mitchell*, 2009). Applied to product concepts, complexity measures allow a designer to ask, "how much knowledge will I need to generate to understand this alternative?" In essence, this question helps designers translate the product-centric perspective to a knowledge-centric one.

Complexity measures are also used to translate design processes into a knowledge-centric perspective. Analyses can translate interactions between process activities into a quantification of the process complexity (*Doerry*, 2009). This is another surrogate of the knowledge-based design description. More complex processes require more generation effort to achieve a particular level of designer knowledge. The key insight behind this analysis is that the knowledge structures designers create can significantly impact design outcomes.

Product and process complexity help navigate the design of complex products by providing information about emerging behaviors in the underlying knowledge-based system. However, there are cases where product and process level analyses are unable to explain emergent design behaviors. In the face of emergent design failures, this leaves designers to fall back on design practices that mitigate the negative effects of design complexity.

2.3 Mitigating Knowledge-Based Complexity

The effects of design complexity did not become a problem in naval programs suddenly; they have grown with the increasing scale of naval products and the interdependencies of systems and stakeholders behind them (*Arena et al.*, 2006; *Dobson*, 2014; *Kana et al.*, 2016). Over time, practices have evolved to manage complex design behaviors and mitigate the risk of failure. Using the McKenney-Singer design taxonomy (*McKenney and Singer*, 2014), these practices can be broken into parts:

- Design Approach: Overarching guiding principle of a design effort.
- Design Process: A series of structured steps to implement the design approach.
- Design Method: The way in which design alternatives are understood, analyzed, and selected for a particular approach and process.
- Design Tool: In support of a design activity, tools provide information that enables designer decision-making.

All four parts of the taxonomy must exist to complete a design activity. When implemented correctly, each of the four parts can help prevent emergent design failures. This section will discuss how Set-Based Design design activities and Systems Engineering design activities implicitly use different parts of the taxonomy to avoid knowledge-based complexity.

2.3.1 Set-Based Design

Set-Based Design was developed at The Toyota Motor Corporation (*Ward et al.*, 1995) and have since been applied to aerospace (*Bernstein and Deyst*, 1998) and naval ship design (*Singer et al.*, 2009). The premise of Set-Based Design is product discovery by way of elimination instead of selection. The method is characterized by: (1) communicating broad sets of design values, (2) developing sets of design solutions, and (3) delaying design decisions until adequate information is known (*Singer et al.*, 2009).

Set-Based Design, visualized in Figure 2.4, uses separate groups of experts to continuously provide opinions for design values within the design space. Intersections between these groups' feasible and preferred regions are identified and retained. Infeasible or dominated regions are removed from further consideration. Using the more concentrated set of alternatives higher quality information is created (*McKenney*, 2013). When all sets are feasible, and all tradeoffs are explored, the best possible design can be selected (*Bernstein and Deyst*, 1998).

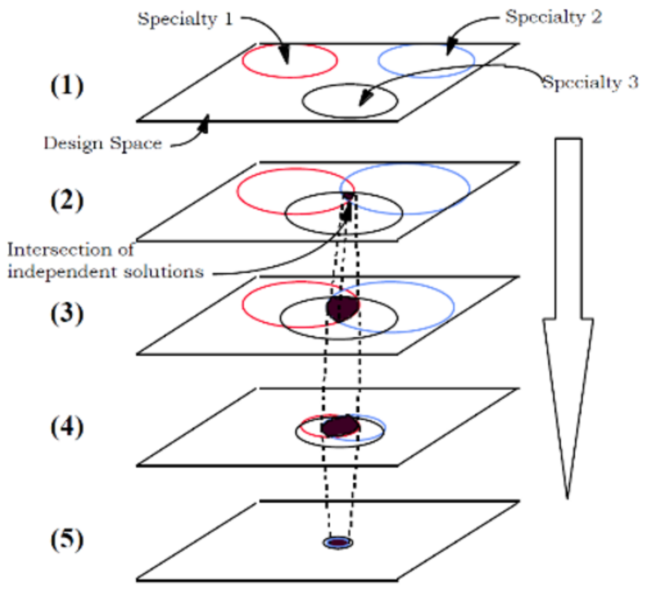


Figure 2.4: Set-Based communication and convergence *Bernstein and Deyst* (1998).

Design activities using Set-Based Design implement a Concurrent Engineering approach. Concurrent Engineering advocates for considering all aspects of a products lifecycle from the outset of design. This enables designers to identify errors and redesign early when the project is flexible (*Kusiak*, 1993). The Set-Based Design method uses the knowledge generated in Concurrent Engineering to enable the elimination of infeasible and dominated sets of alternatives. Over time, this reduces and refines the number of alternatives until a fully understood concept is converged.

Set-Based Design is often facilitated by specific execution tools that negotiate knowledge generated by traditional engineering tools. However, Set-Based Design does not specify a design process for its implementation. This leaves the designer to develop a tailored process for each design activity. For example, the Ship to Shore Connector design activity used the Decision-Oriented Systems Engineering process to develop the execution steps and tasks needed to implement Set-Based Design (*Buckley*, 2009; *Singer et al.*, 2009), while the Amphibious Combat Vehicle used the U.S. Navy Concept Design Process to develop the process needed to execute Set-Based Design (*Doerry et al.*, 2014).

Set-Based Design controls temporal factors that can lead to emergence by accounting for unexpected internal and external factors. The set-based convergence method accomplishes this by guiding how design knowledge is generated. This creates two unique knowledge structure properties:

1. Eliminating dominated or infeasible regions promotes design progression from well-understood regions. In Set-Based Design, solutions can only be eliminated if no new information would change the outcome. Thus, only low-risk knowledge is used to generate new design knowledge. This reduces the probability that internal or external factors of complexity will cause emergent behavior.

2. Knowledge about overlapping regions between sets is generated in parallel. This prevents temporal factors from creating path dependencies towards knowledge about a single well-developed solution concept and accounts for internal and external factors.

Implementing the set-based convergence method requires that sets are understood and defined. Sets are typically established at the outset of the design activity, after knowledge interdependencies between relevant disciplines are identified. The Concurrent Engineering design approach and a tailored process facilitates this step.

In terms of the knowledge structure, Set-Based Design creates relevant knowledge interdependencies early in the design activity. The knowledge structure is first created using groups of experts to generate a large body of initial knowledge. This refines the design problem and creates a knowledge structure encapsulating relevant disciplines, their interdependencies, and their possible external factors. In Set-Based Design, the initial knowledge is used to generate sets which control future knowledge generation and decision-making actions.

During concept development, experts repeatedly evaluate and refine the established sets. Regions within a set are classified as dominated or infeasible when they are un-preferred by designers and no new knowledge is likely to change the evaluation. When this occurs, designers decide to stop developing the associated knowledge. Over time, decisions converge the knowledge structure on a stable defined solution. Elimination decisions create a low-risk knowledge structure. Only making decisions when the supporting knowledge is well-understood and is unlikely to change leaves stable knowledge to be further developed. In the face of internal or external factors of complexity, this knowledge is unlikely to need to be refined to support a design shift. In this way, set-based convergence builds the knowledge structure from a diverse and stable body of knowledge. While this requires a significant amount of knowledge generation and decision-making, it enables designers to prevent design failures by leveraging temporal factors.

When Set-Based Design is properly executed, knowledge is generated from all remaining and overlapping sets. This eliminates path dependencies created by knowledge generation focused on a specific part of a solution concept. Generating knowledge from all overlapping sets also helps account for internal and external factors of complexity. If the design problem changes (internal) or something outside of the designers control changes the design landscape (external), its effects are considered for all remaining sets. Because the sets are supported by low-risk knowledge, it is less likely that complexity will drastically change the sets. Additionally, it should require less knowledge generation to adjust the sets because they are refined in parallel, as opposed to being focused on a single concept. This enables designers to manage uncertainty or changes in the design problem while maintaining the developed solution concepts.

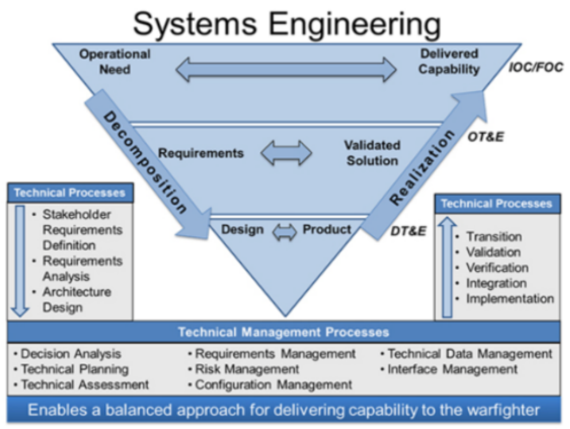


Figure 2.5: Department of Defense Systems Engineering Process Model (*ACQuipedia*, n.d.).

Set-Based Design delays the development of path dependencies while inherently accounting for complex design behaviors. This helps designers understand the changing design landscape and reduces the knowledge generation effort required to converge on a high-quality concept. Compared to traditional selective methods where knowledge structures evolve to support previous decisions, set-based elimination methods create path dependencies away from risky alternatives. In doing so, Set-based Design delays temporal factors. This helps mitigate internal and external factors of complexity. When the design landscape shifts, a flexible knowledge structure allows designers to quickly transition knowledge generation towards more promising areas. In effect, this lifts the "freedom line" in Figure 2.1 so that the designers can better respond to the dancing landscape in Figure 2.2.

2.3.2 Systems Engineering

Systems Engineering focuses on defining customer needs and required functionality early in the development cycle, documenting requirements, then proceeding with design syntheses and system validation while considering the complete problem (*INCOSE*, n.d.). Figure 2.5 visualizes this definition in its classic Vee.

Systems Engineering design activities follow the Systems Engineering process to rigorously implement a Concurrent Engineering design approach. The System Engineering Vee guides designers through specifying a suitable product. The decomposition stage is concerned with understanding the desired product attributes and determining the resources needed to create them. Proceeding through the Vee, desired attributes are translated into the final product. In the realization stage individual attributes are integrated into the product. Finally, the defined product is verified and validated to insure that the integration actually produced the desired results.

Systems Engineering does not specify a design method, instead it suggests that an appropriate multicriteria decision-aiding technique be used (*Bahill and Gissing*, 1998). This leads to the use of decision-making tools that facilitate design process decisions such as Quality Function Deployment and Analytic Hierarchy Process (*Dickerson and Mavris*, 2016; *Kossiakoff*, 2011).

System Engineering decision-making tools are typically iterative, helping designers select the best alternative to further develop. This selection relies on traditional engineering tools to model and understand alternatives. Design execution uses Systems Engineering specific tools. Generated design information is used by a project managers who are responsible for frequent process validation and integrating multiple perspectives into a balanced design (*Kossiakoff*, 2011). Together, these steps can reduce the potential for unexpected landscape dynamics.

Systems Engineering controls knowledge structure development and prevents design failures through the Systems Engineering process. The process defines how design knowledge should be generated and grouped into modules to limit complex design behavior. In effect, this creates a pattern of knowledge and decisions that is repeated throughout the knowledge structure given reasonably predictable design behavior. Before the pattern can be implemented, the underlying knowledge structure must be created.

Initial design learning starts by generating knowledge about the design requirements. Following a Concurrent Engineering approach, the relevant disciplines are identified and a large body of interdependent knowledge is generated to decide on a set of requirements. To achieve this, information about product uncertainty must be well characterized. This starting condition provides a stable knowledge structure to begin generating concept knowledge.

Once concept development begins, Systems Engineering uses the repeatable knowledge generation and decision-making pattern defined by decomposition to generate the knowledge structure. Decomposition modularizes the generation of knowledge along sub-systems so that the composite system can be understood. This reduces the required knowledge generation efforts by limiting knowledge interdependencies. The realization side of the Vee uses the knowledge structure developed for each sub-system to validate the sub-system performance. The pattern is repeated for the lowest-level sub-systems and then integrated up to larger composites. Because knowledge interdependencies are limited during decomposition, the propagation of the knowledge generated during these actions is also limited. Limiting knowledge propagation prevents emergent rework and integration failures.

Knowledge modularization also guides decision-making. Using the segmented knowledge structure to make decisions limits how they propagate in the knowledge structure. This further reduces the potential for emergent failures.

Frequent design reviews and validation steps built into the System Engineering process help identify and address unexpected design behavior. When the knowledge decomposition and integration is impacted by changes to the design problem by any of the three factors, it is identified through validation. Once identified, new knowledge is generated and integrated to address any incompatibilities in the knowledge structure. When executed properly, Systems Engineering creates a repeatable pattern of knowledge generation that limits emergent design failures. The pattern and resulting knowledge structure controls external and internal factors of complexity and helps managers track temporal factors. This reduces the potential for emergent failures and limits the effect if they occur.

2.3.3 Mitigation Differences

Design activities that utilize Set-Based Design and Systems Engineering both control knowledge-based complex behavior and mitigate design failures. Set-Based Design achieves this through a method for investigating alternatives. Systems Engineering achieves this through a design process that controls how the design activity is executed.

When Set-Based Design and Systems Engineering are executed correctly, they both yield products that satisfy the design problem. However, the complexity mitigations provided by each are different. Set-Based Design controls the development of temporal factors towards robust knowledge structures. Systems Engineering controls internal factors to ensure design decisions are well-defined and understood.

For certain circumstances, one mitigation strategy may be better suited than another. So far, choosing how they are applied is based on manager intuition or organizational practice. A better understanding of knowledge-based complexity could identify when and how each strategy should be leveraged. Further, nowhere in these design practices is knowledge-based design directly addressed. Identifying emergent behaviors that cause design failures will require more explicit techniques.

2.4 Conclusions

Avoiding emergent rework, inability to integrate, and unexpected design difficulty are critical objectives in naval design practices. Preventing these failures requires that the behaviors that cause them are well defined and understood. Examining design as a knowledge-based system illustrates that many observed failures are driven by complex knowledge generation and decision-making behaviors, not from the product itself. While the current state-of-the-art design practices implicitly mitigate some of this complexity, a more thorough perspective is needed. Returning to the research questions proposed in the introduction to the chapter:

- What drives emergent design failures in the design of complex naval systems?

  Defining design as the act of generating knowledge for decision-making through time shows that emergent design failures have three potential causes: external factors that require new designer knowledge; internal factors that change designer knowledge of the problem and solution concepts; temporal factors that drive a solution and limit designers' ability to affect design outcomes. Failures occur when one or more of these factors cause a significant increase in the amount of knowledge required to complete the design activity or makes a large portion of designer knowledge incompatible. When these behaviors emerge, extensive work and rework can be required to understand and integrate the design.

- What strategies are available to mitigate and prevent design failures?

  Design practices have evolved to handle the factors that cause emergent design failures. Rigorously managing internal factors, limiting exposure to external factors, and executing design to leverage beneficial temporal factors all help prevent failures. Towards this goal, analyzing product complexity measures knowledge-centric impact and can help identify the potential for failure earlier in the design activity.

- How can these strategies be improved?

  Existing practices and analyses can implicitly address knowledge-based causes of design failures. However, as the structure of knowledge generation and decision-making increases in complexity, they become lacking. Tools and techniques to explicitly model and analyze design's knowledge-based complexity may help identify emergent failures before they develop.

To implement these improvements, a mathematical framework is needed to define design as a knowledge-based complex system. This will provide a basis for the direct study of the causal factors of design complexity. Developing these techniques and analyses will support knowledge-based design analysis as a usable perspective for improving naval design. The following chapters will develop the required framework, demonstrate its efficacy, and apply it to the naval distributed system design problem.

CHAPTER III

Knowledge Structures and Path Dependence

The previous chapter described how naval design failures emerge from the knowledge structure created during a design activity. The emergence of design failure is influenced by internal, external, and temporal factors of complexity. To concretely and quantitatively understand how design failures develop, a formal framework for studying design knowledge structures is needed. First, a definition of knowledge structure elements is needed. Second, the mechanisms relating knowledge structures and design failures must be more clearly defined. Third, a framework for tracking knowledge structures during a design activity is required.

In this chapter, knowledge structures are defined as a set of interdependent knowledge and decisions that are expanded through designers' actions. Establishing a definition for knowledge structures also requires a definition for product structures. Product structures describe a product's constituent elements and their relationships. There is not a correct form of a product structure, it only has to be sufficient for the design activity at hand. For example, a rectangle can be described by length and width elements, or it can be described as an aspect ratio and scalar. In naval design, some product elements may be length, beam, and draft variables that are related by geometric relationships. Other examples are the spaces and components within the vessel and their relationships to one another by distributed systems.

Design outcomes are the knowledge structure and product structure created during a design activity. Together, they describe how a product was developed and the products definition. However, design activities are typically approached solely from a product perspective. The product perspective is concerned with the quality of a product structure and the best approach to define it. Systems Engineering is a prime example - it prescribes how requirements should be decomposed directly into product elements and relationships. This product perspective neglects the knowledge-based behaviors that drive the product structure definition. In contrast, the knowledge perspective is concerned with the development of the information, ideas, and decisions that allow the designer to define a product structure.

The purpose of this chapter and Chapter IV is to provide clarity on form and function of knowledge structures so that a knowledge perspective can be applied to design activities. This chapter will discuss similar and overlapping research on knowledge structures. Previous research will frame the current research and allow recognition of its unique attributes. Additionally, the review will help establish the components of a knowledge structure so that a more rigorous definition can be presented in the next chapter.

The first subject is the general application of knowledge structures, Section 3.1, which have been used heavily to study human cognition and the behavior of business firms. Understanding this background will help build the foundation for understanding how knowledge structures can be applied to naval design. Previous design research will also be discussed and relevant knowledge structure insights are identified. Section 3.2 discusses the relationship between knowledge structures and path dependencies. Path dependence provides a mechanism to understand how and why knowledge structures influence design activities. Section 3.3 identifies a number of research gaps that limit the scope of knowledge structure analysis and application in design. Section 3.4 takes a step towards bridging those gaps by describing how design activities can be recorded and understood through their knowledge structures. A brief summary concludes the chapter.

3.1 Knowledge Structures

The concept of knowledge structures was developed to improve understanding of how individuals recall information based on their knowledge and its cognitive organization (*Chi*, 1978). This made knowledge structures an important idea in education, where the goal was to help individuals acquire and use new knowledge (*Bernstein*, 1996). In tangential research, investigations occurred to ascertain how knowledge structures developed. It was found that different experiences could shape in individual's knowledge structure that would in turn influence how they perceived the world. This was well illustrated in a study of the relationship between knowledge structures and gender (*Markus et al.*, 1982).

More generally, the concept of the knowledge structure linked how information was stored, to how it would be used (*Schank and Abelson*, 2013). The relationships between an individual's knowledge grow over time (*Maton*, 2007) and significantly influences the way new information is processed, understood, and used (Bernstein, 1999). This relationship makes it possible to consider how an individual will solve problems based on the development and composition of their knowledge structure.

Knowledge structures have similar implications for organizations. Organization strategy studies the relationships between people, teams, and entities within a firm (*Miles et al.*, 1978). These works found that the ideas behind studying an individual's knowledge structure could be applied to organizations and the people within them (*Lyles and Schwenk*, 1992; *Sparrow*, 1999). Primarily, this idea was leveraged to measure a firm's ability to innovate and learn (*Cohen and Levinthal*, 1990; *Galunic and Rodan*, 1998). These studies found that knowledge structures influence learning and decision-making behaviors at the individual, team, and organizational levels.

Knowledge structures create an interesting linkage between the past, present, and future. Experiences and the acquisition of knowledge shape an individual's knowledge structure. That knowledge structure then affects how new knowledge is acquired and used. The result is a relationship and feedback between what is known now and how an individual or organization might behave in the future.

Applied to naval design failures, the knowledge structure encapsulated in the Navy and participating entities influences the outcomes of naval acquisition and design. The knowledge structure built from past experiences will change how new ship design activities will be approached, how new information will be used, and what decisions will be made. Because of this, design failures are not just a function of an unfortunate knowledge structure shift, they can be primed by existing knowledge and emerge from knowledge structure growth.

The study of knowledge structures in design activities provides some insight into how this occurs. Design research that investigates how individuals or teams design often implicitly rely on the idea of a growing knowledge structure that influences design outcomes.

It is well documented that design activities begin with some initial knowledge based on designer experience and subject matter expertise (*Laxton*, 1969). Over the course of the design activity, designers build on their existing knowledge. For example, concept exploration is used to grow and refine knowledge about the problem and possible solutions (*Goldschmidt and Weil*, 1998). This describes an initial knowledge structure the designer brings to a design activity that is incrementally developed. Studying how this growth occurs has been a fruitful branch of research in design cognition, named design studies, that examine how individuals and teams design (*Cross*, 2001).

Generally, design studies observe, encode, and analyze human behaviors during a short design activity (*Cross et al.*, 1996). For example, studying how knowledge is developed can provide insight into how the designer approached the design activity (*Goldschmidt*, 1995; *Goldschmidt and Weil*, 1998; *Kan and Gero*, 2008). Similar methods investigate growing knowledge relationships to qualitatively describe how the design activity was completed (*van der Lugt*, 2001; *Cash et al.*, 2014) and quantitatively measure how knowledge is used by the designer (*Kim and Kim*, 2015; *Cash and Štorga*, 2015).

Design studies and their employed analyses provide two insights into studying emergent naval design failures. First and fortunately, an entity's or individual's knowledge structure does not need to be defined a priori. Instead, tracking how elements of the knowledge structure are used during the design activity can capture the relevant behaviors. This means that the Navy's knowledge structure does not need to be defined before design failures can be prevented.

Second, the actual product being designed is largely irrelevant to knowledge structure analysis. Design analysis extracts information from the design activity by examining the knowledge structure's development, not the product structure's (the specified product's elements and relationships). Knowledge about the product structure is used by the designer with other knowledge structure elements to further develop the design. As a corollary, tracking product models will not describe the complexities of a design activity because they only describe a subsection of the knowledge structure. Thus, product models are not likely to provide real insight into potential design failures. The question is, how does the knowledge structure provide this insight? This can be addressed through the concept of path dependence.

3.2 Knowledge Structures and Path Dependence

Casually, path dependence implies that history matters (*Pierson*, 2000, 2004). Described in Chapter II as the temporal factor of design complexity, path dependence means that the sequence in which design is approached will influence design outcomes.

Specifically, the order of historical events matters, not just the set of historical events (*Page*, 2006). For path dependence to exist, externalities give past events influence on future events is necessary (*Page*, 2006). This means that recognizing temporal factors and path dependence in design requires that design externalities are identified.

The realities of path dependence can be observed in large- and small-scale design activities. Dosi describes technology arcs, the path by which a technology is developed, as path dependent (*Dosi*, 1982). He observed that path dependence towards ideas and solutions become encoded in the technology design knowledge structures. Creating the knowledge to support technology innovation requires costly research and development efforts. Once development for an idea begins, the relative value of alternative ideas decreases. This is can be a simultaneously a self-reinforcing positive externality and a negative externality on other parts of the knowledge structure that were not developed. These externalities also establish the necessary conditions for path dependence (*Page*, 2006; *Mahoney*, 2000). The same feedback mechanism occurs in small-scale design.

In studies of individual designers, Lawson identified that the sequence of design can change the designer's solution quality and difficulty of reaching that solution (*Lawson*, 1979). This is caused by the knowledge relationships created during a design activity. If a part of the knowledge structure is used frequently, it will be better developed and thus may have more value to the designer relative to an unused part. This conclusion is supported by multiple experiments observing that well-developed knowledge structure elements are likely to be used repeatedly to progress the design activity (*Goldschmidt*, 1990; *van der Lugt*, 2000; *Cross*, 2001; *Akin and Lin*, 1995; *Kroll et al.*, 2014). *Parker and Singer* (2015) identified similar behavior in structured design processes. For example, Parker found that a few design parameters were capable of driving ship design processes towards certain configurations. These highly-influential elements are called *design drivers*. Knowledge structure development, design drivers, and path dependence creates a way to understand how the design activity history will influence future design outcomes.

In the early-stages of design activities, many outcomes are possible. These outcomes can be classified as successes or failures. Design success occurs when a satisfactory product is defined and understood. Design failure occurs when there is excessive rework, the inability to integrate the design, and significantly increasing design effort. Reaching a successful outcome requires design development, that the designers take actions to generate knowledge and make decisions. Some of these actions will influence future actions. Others will cause design failure outright. However, in most instances, actions themselves will only shift the probability of reach the possible design outcomes. This is the condition that creates path dependence. Emergent design failures occur when the set of past actions shift the design outcome into failure. This is different than non-emergent failures when an action causes failure outright, no matter the circumstances. Shifts toward emergent failure can happen slowly, incrementally narrowing in on failure, or quickly, resulting in sudden failure. *Lamberson and Page* (2012) differentiate between path dependence, where the shift towards outcomes happens slowly, tipping points, where the shift happens quickly, and contextual tipping points, where stage is set for a shift to occur quickly.

This dissertation focuses on the role of path dependence in design and emergent failures, but path dependence, tipping points, and contextual tipping points can all shift design activities towards emergent failure. For example, much of the cost overruns and delays for the LCS-1 have been attributed to an incorrectly made reduction gear (*O'Rourke*, 2017b). The reduction gear appears to be a tipping point that suddenly caused massive rework and other suboptimal design choices. However, preceding the reduction gear flaws, a contextual tipping point was created by the LCS programs attempt at a rapid acquisition strategy where ships were being constructed before the design was completed. Once that acquisition strategy was decided on, it is likely that any flaw or delay in a semi-major or major component would cause the tip into design failure. Looking back further into the LCS acquisition history, the decision to pursue a rapid acquisition strategy was a result of path dependence that can be tracked back to the early vessel concepts. One key aspect of the LCS was that it would employ a Speed-to-Fleet strategy of using many quickly produced LCSs to fill the Navys capability gaps *Work* (2014). This contributed to a long series of design actions and decisions that led the LCS program to pursue the high risk acquisition strategy that may have ultimately set the stage for a tip into failure. In isolation, the faulty reduction gear is not a cause for design failure, massive cost overruns, and significant delays. Instead, the LCS design failure was emergent. The failure was the result of the knowledge, actions, and decisions that make up the LCS design history. In order to understand how this type of failure develops and, more importantly, move towards preventing emergent failures in the future, a knowledge-based perspective of naval design is required.

Path dependence in large- and small-scale design activity can arise from the history of knowledge structure growth. In a design activity, the designer incrementally grows the knowledge structure through their actions. The developed knowledge and relationships defines how future action will alter the current state of the design (*Levinthal and Warglien*, 1999). This feedback can develop path dependencies that favor certain outcomes and the knowledge that supports them (*Rycroft and Kash*, 2002). Applying the lessons learned from design studies, it is suggested that design drivers and path dependencies are fundamentally intertwined.

Path dependence and design drivers introduce a number of opportunities to understand and analyze emergent naval design failures:

- Because the relative value of knowledge and its relationships increases as it is used, shifting away from an undesirable outcome may require massive design rework to develop and use other knowledge.

- Naval design happens across many entities, each with their own pieces within the design knowledge structure. If two entities have incompatible design drivers, integration may be impossible.
- Some paths may be more arduous, require more knowledge structure growth, than others. Taking a certain series of actions early in the design activity may increase the probability to future design difficulty.

Knowledge structure development is far from the only way path dependencies could arise in design, but it exists in all non-trivial design activities. Because of this, it is critical to understand how knowledge structures can create path dependencies that will allow design failures. Creating tools and methods to consider these failures requires that a number of research gaps are filled.

3.3 Research Gap

Predicting and preventing emergent design failures hinges on two concepts. First, that knowledge structure growth accurately describes design activities and second, the history of growth influences design activity outcomes. However, the implementation and application of these concepts is not within the scope of current theories and methods. There are three primary gaps:

1. No method exists to track knowledge structure growth within a design activity and to relate that growth to design outcomes.
2. There is no analysis for capturing the path dependence of knowledge structure growth.
3. No methods exist to investigate how knowledge structure growth occurs during a design activity.

Until these gaps are filled, designers will not be able to systematically understand how they can influence design outcomes and prevent design failures. These gaps are amplified in their application to naval design. Naval design requires the integration of many disciplines, concurrent technology development, and designing with limited information over long time periods. This means that the knowledge structure is large, distributed across many entities and individuals, and its growth may be highly variable. These complications are most relevant in ship systems design and integration.

The complexity measures used to estimate cost implications, shown in Chapter I, quantify the interactions between systems within the physical product. Product-based complexity represents the underlying requirements on knowledge structure growth. High levels of system interaction require more knowledge structure growth to understand the integrated concepts. As the number of system interactions grow, so does the number of size of the knowledge structure that must be developed.

Unfortunately, knowledge relationships are solidified by earlier decisions about the vessel's configuration and components. By the time information about the system design knowledge interdependencies are available, it may be too late to change the knowledge and product structure to accommodate them. This is a fundamental issue in applying knowledge structure analysis to naval design. Early decisions are made with limited information, but can predicate future knowledge-based failures. The issue is compounded by the uncertainty of knowledge growth tied to technology development and the integration of multiple disciplines and entities.

Emergent naval design failures need to be considered early, before path dependencies ingrain undesirable outcomes and can only be found through knowledge structure development. The remainder of this dissertation defines a framework for tracking knowledge structure growth and knowledge-based path dependencies and then applies the framework to consider emergent failures in distributed system design. The following section will step towards that goal by describing the proposed framework.

3.4 Recording Design Through Knowledge Structures

Knowledge structures have been described and represented in a myriad of ways depending on the discipline using them and the objective. Here, the objective is to track the history of a design activity through knowledge structure growth so that the distribution of design outcomes can be considered. In this section, that objective will be decomposed to identify what elements of knowledge structure growth need to be tracked. To facilitate this, the following definition of design will be used:

Design is the act of generating knowledge for decision-making through time.

This characterization has been observed and stated in various ways in a multitude of design theories (*Gero*, 1990, 1996; *Hatchuel and Weil*, 2003), design studies (*Visser and Trousse*, 1993; *Oxman*, 1990; *Ward et al.*, 1997; *Kim and Kim*, 2015), and design process analyses (*Mavris and DeLaurentis*, 2000; *Gillespie et al.*, 2013; *Parker*, 2014; *Parker and Singer*, 2015). It can be separated into relevant parts of a growing knowledge structure (bolded):

Design is the ***act*** *of generating* ***knowledge*** *for* ***decision-making*** *through time.*

Knowledge is the building blocks that allows the designer to progress the design activity. The design activity is progressed through designer actions, which use existing knowledge to grow the knowledge structure and make decisions. Decisions commit the design to product structure elements and relationships. Used through time, knowledge and decision elements compose the design knowledge structure that grows through designer actions and create design outcomes.

Knowledge is defined as, “what a system has that allows it to attain its goals” (*Reich*, 1995). In design activities, the goal is to define a satisfactory solution to a design problem (*Simon*, 1969). To achieve this goal, designers use product concepts, evidence, and ideas to progress towards a solution (*Hatchuel and Weil*, 2003, 2009). Thus, design knowledge can be defined as the ideas, concept elements, and evidence used by the designer.

There are three different ways a designer can possess or acquire knowledge: it can already exist in the designers' knowledge structure, it can be external to the design activity, and it can be generated by the designers. Pre-existing knowledge, or precedents, allows designers to negotiate problems by providing references to previous solutions, problem-solving elements, and context to help understand the design situation (*Oxman*, 1990; *Visser and Trousse*, 1993; *Visser*, 1995). External knowledge is simply put into the design activity, for instance a manager says there is an additional budgetary constraint. Generated knowledge is created by the designer. For example, when the designer is learning, exploring concept ideas, or refining previous knowledge (*Gero*, 1996; *Sim and Duffy*, 2004; *Kim and Kim*, 2015). Refining knowledge is a specific case of knowledge generation, where existing knowledge is adjusted or adapted, as opposed to being novel to the designer.

Knowledge is created through designer actions. Designer actions can generate, inspect, and adjust designer knowledge (*Goldschmidt and Weil*, 1998; *Goldschmidt*, 2014). Other actions introduce precedents or external knowledge into the knowledge structure. Over time, actions enable the design to progress by building on each other throughout the design activity (*Cash and Štorga*, 2015). Knowledge added by one action will be used in future actions and so on. Thus, actions are incremental steps that may use elements of the knowledge structure to create or refine new knowledge elements.

Actions that add to designer knowledge can be differentiated from design decisions. Decisions are a choice between alternatives (*Hazelrigg*, 1996; *Hansen and Andreasen*, 2004). Decisions imply a commitment to using the chosen alternative in the future.

This can be the decision to use specific knowledge structure elements in the design activity. For example, doing system analysis with a specific model or calculating ship resistance with regression versus standard series. Decisions also apply to committing to product structure elements. Designers define a product structure by deciding on a set of elements or relationships in the product structure. For example, deciding on the length of a ship. Decisions are the knowledge structure elements that interface between ideas, concepts, and evidence and the product solution and arise in two ways. Designer create some decisions through decision-making choices that leverage existing knowledge and decision. External factors can also mandate decision. Both types of decisions can create new product structure elements or relationships or refine existing decisions.

To summarize, knowledge structures are composed of knowledge and decisions. Knowledge structures are developed by actions and decision-making. The definition of each type of element is:

- Knowledge: the ideas, concept elements, and evidence used by the designer.
- Actions: incremental steps that may use elements of the knowledge structure to create or refine new knowledge elements.
- Decisions: commitment to a segment of the knowledge structure, this often applies to knowledge about the elements and relationships in the product structure.

Definitions for knowledge, action, and decision elements and their relationships enable a description of knowledge structure development in design:

1. Design begins with knowledge of previous solutions, problem solving elements, and an initial understanding of the design problem.

2. Design is progressed through actions that use existing and external knowledge and decisions to learn, explore, refine, and make decisions about the design solution, which are then accumulated in the knowledge structure.

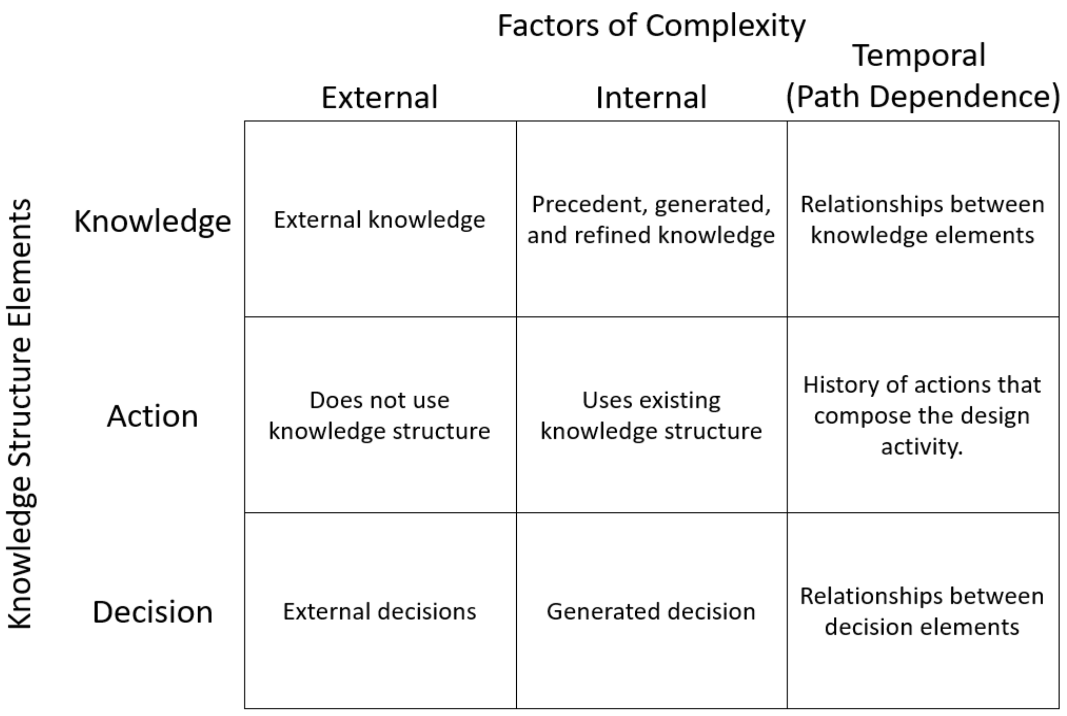
3. Design is completed when decisions specify a solution and it is known that the solution satisfies the design problem, or it is decided that no satisfactory solution exists. The design outcome is design activity's knowledge structure and the committed product structure.

This description provides a path towards defining a formal framework that captures knowledge structure development and the factors that influence it. Relationships between elements can be tracked through the actions and decisions that use and create the knowledge structure. Applied over the course of a design activity the resulting knowledge structure growth can capture the factors of design complexity that influence outcomes, see Figure 3.1.

Discussed in Chapter II, knowledge structures are influenced by external, internal, and temporal factors of complexity. External factors are outside of the designer's control, but must be integrated into the design activity. Internal factors arise from the knowledge structure development, especially when design drivers or ingrained knowledge is refined or revised. Temporal factors are path dependencies which ingrain solutions and ideas over time. Representing these factors through knowledge structure development facilitates a knowledge perspective of design that can enable the identification and prevention of design failures.

3.5 Conclusion

This chapter provided the necessary background to establish how and why knowledge structures can be used to understand design activities. The review of similar and overlapping research framed the current research and its unique perspective. Previ-



Knowledge Structure Elements	External	Internal	Temporal (Path Dependence)
Knowledge	External knowledge	Precedent, generated, and refined knowledge	Relationships between knowledge elements
Action	Does not use knowledge structure	Uses existing knowledge structure	History of actions that compose the design activity.
Decision	External decisions	Generated decision	Relationships between decision elements

Figure 3.1: Relationship between knowledge structure development and factors of design complexity.

ous applications of knowledge structures to design science have been largely implicit, relying on the fact that the content of, and relationships between, an individual's knowledge will influence design outcomes. Research in path dependence provided insight into why this influence exists. This thesis proposes that explicit consideration of the development of knowledge structures can reveal the path dependence of decisions and help reduce design failures.

The distinguishing characteristics of this approach include explicitly tracking knowledge structure growth and its relationship to design outcomes, so that the trajectory of a design activity can be understood. The second distinguishing feature is analysis based on knowledge structure growth to identify the path dependencies influencing outcomes. The third feature is a new perspective on how to explore the relationship between potential knowledge structure growth and future design outcomes. The objective of these features is to link the past, present, and future of a design activity, which will ultimately enable designers to measure the potential for design failures and leverage the complex factors of design towards better outcomes.

CHAPTER IV

Formal Framework for Mapping Design Knowledge Structures

This chapter introduces a formal framework for describing knowledge structure development during design. Application of this framework, its network representation, and network analyses can quantify how a knowledge structure grows through time and identify design drivers in the design activity. Thus, this framework provides the foundation for investigating the potential for design failures. Section 4.1 introduces the formal K-A-D Framework for describing design knowledge structure development. Section 4.2 defines a network representation of the framework that enables knowledge structure analysis, which is introduced in Section 4.3. Section 4.4 demonstrates the K-A-D Framework, its network representation, and analyses on an example design activity.

4.1 Knowledge-Action-Decision Framework

When designers solve design problems, they develop a knowledge structure of the knowledge they use and decisions they make. The composition and development of this knowledge structure creates design outcomes; the specified solution and the decisions that define the specification. Tracking how a knowledge structure develops can help measure design drivers, identify path dependencies, and predict the emergence of design failures. This is enabled by the K-A-D Framework, which describes knowledge structure development in context of external, internal, and temporal factors of design complexity so that the emergence of design outcomes can be understood.

The previous chapter defined three primary types of knowledge structure elements and their sub-types:

- Knowledge: the ideas, concept elements, and evidence used by the designer.
  - Precedents: previous solutions, problem solving elements, and an initial understanding of the design problem.
  - External: information added to the design activity from outside of a designer's control.
  - Generated: new ideas, concept elements, and evidence created by using the existing knowledge structure.
  - Refined: new knowledge that is a small change to existing knowledge.
- Actions: incremental steps that may use elements of the knowledge structure to create or refine new knowledge structure elements.
- Decisions: commitment to a segment of the knowledge structure, this often applies to knowledge about the elements and relationships in the product structure.
  - External: decision added to the design activity from outside of a designer's control.
  - Generated: decision that uses the existing knowledge structure to commit to a new product structure element or relationships
  - Refined: decision that is a small change to an existing decision.

Recording how these types of elements are used by the designer during the design activity captures how the factors of complexity impact design. External factors are captured in the use of external knowledge and decisions. Internal factors are captured by the actions that generate knowledge and decisions. Temporal factors are captured by the growing relationships between knowledge and actions and their use in designer decisions.

Quantitatively investigating this description is best done with a formal framework. The goal of this section is to define the Knowledge-Action-Decision Framework that provides definitions of knowledge structure development through a dynamic process. The process incrementally grows the knowledge structure at discrete time steps $t = 1, 2, 3 \ldots$. The knowledge and decision elements contained in the knowledge structure in a given time step is K_t. For example, in the design of an engine room the knowledge structure may be $K_t = \{$Engine weight in range $[41500kg, 62200kg]$, power in range $[3600kW, 5400kW]\}$. The knowledge structure is acted upon by a generative function G_t, that represents the designer's action at t. The generative function adds new knowledge elements to the knowledge structure. The history of knowledge structure development is a series of the generative functions on acting on the knowledge structure. This is defined as

$$K_{t+1} = G_t(k, r) + K_t; k, r \in K_t, \tag{4.1}$$

which can be rearranged to solve for $\Delta K_t = K_{t+1} - K_t$, the knowledge structure elements added by G_t,

$$\Delta K_t = G_t(k, r); k, r \in K_t, \tag{4.2}$$

where k represents the subset of the existing knowledge structure used by the action and r is used to denote if the added element is a refinement of a previously existing element. For instance, if some designer knowledge is k_i="Engine weight in range $[41500kg, 62200kg]$" and an action uses k_i to create a new element k_j="Engine weight is range $48400kg$," then k_j could be represented as a refinement of k_i. In this case, the growth equation would be $K_{t+1} = G_t(k_i, k_i)$, where K_{t+1} would contain the knowledge element k_j. This general equation can be applied to the sub-types of knowledge structure elements.

Precedent knowledge, which the designers bring to the activity are not generated by an action, but they are introduced to the design knowledge structure before they are used. In terms of the generative function, introducing a precedent is

$$\Delta K_t = G_t(\emptyset, \emptyset). \tag{4.3}$$

Generated knowledge is created when the designer uses the existing knowledge structure. Creation of a new idea, concept element, or piece of evidence, creates the mathematical form

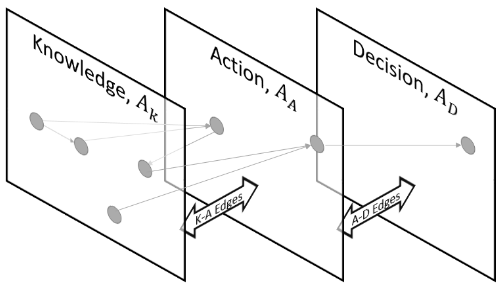
$$\Delta K_t = G_t(k, \emptyset). \tag{4.4}$$

Knowledge refinement uses the existing knowledge structure to make an adjustment to a knowledge element. This is defined above in Equation 4.2 and is applied when $r \neq \emptyset$.

Decisions are mathematically described in the same way as knowledge generation, but are denoted as a generative function D_t. The knowledge structure change equation for a decision is thus

$$\Delta K_t = D_t(k, r); k, r \in K_t. \tag{4.5}$$

Decisions follow the logic and definitions for design knowledge. External decisions are defined as in Equation 4.3. New decisions affecting non-specified parts or elements



$$A_{KAD} = \begin{pmatrix} A_K & A_{KA} & A_{KD} \\ A_{AK} & A_A & A_{AD} \\ A_{DK} & A_{DA} & A_D \end{pmatrix}$$

Figure 4.1: Structure of the K-A-D Framework network representation and its supra-adjacency matrix A_{KAD}. Layers A_k, A_A, A_D contain nodes of individual design elements and in their relationships. Inter-layer connections are contained on the off-diagonal elements.

of the product structure are defined as in Equation 4.4 and refinements on previous decisions as in Equation 4.2. Through the same reasoning described above, new decisions and decision refinement are internal factors that define how the knowledge structure is used to generate a product.

4.2 K-A-D Framework: Network Representation

Mathematical definitions of knowledge structure growth describe the design activity history. Representing the growth functions as a growing network of knowledge, action, and decision elements and their relationships can help analyze the mathematical description. Networks are defined as a collection of objects, or nodes, and the connections between them, called edges, that represent a system (*Newman*, 2003). Network science provides a mathematical framework to study structure and creation of these relationships (*Newman*, 2003). Systems can then be studied by analyzing the structure of the network representation (*Newman*, 2010). Applied to the K-A-D Framework, network representations and analyses provide a powerful toolset for quantifying a design activity.

Mathematically, networks are defined by an $[N \times N]$ adjacency matrix A where N is the number of objects represented in the network (*Newman*, 2010). Relationships between objects, called edges, are denoted by non-zero matrix elements. Networks with undirected edges represent only that two objects are related and thus have a symmetric adjacency matrix. Directed edges represent that one node acts on another. For example, if node i influences node j, the matrix element $A_{ij} = 1$, but $A_{ji} = 0$.

Networks may also be layered into multilayered networks to encompass multiple types of relationships, temporal dynamics, and other complicated connectivities (*Kivelä et al.*, 2014). Multilayer networks enable the analysis of multiple channels of connectivity, each channel is represented by a layer with its own respective nodes and edges (*Boccaletti et al.*, 2014). Each layer is represented by an $[n \times n]$ matrix A_α where n is the number of objects in layer α. Layer adjacency matrices are on the diagonal of the larger multilayer adjacency matrix, called the supra-adjacency matrix. Relationships between layers are off-diagonal; edges from the elements of channel α to β are in the sub-matrix $A_{\alpha\beta}$ of the supra-adjacency matrix.

The K-A-D Framework can be represented as a growing multilayer network. Types of framework elements - knowledge, actions, and decisions - are represented in separate layers and their relationships are connections within or between layers. In this network, knowledge, actions, and decisions nodes represent individual elements of the knowledge structure. The corresponding knowledge, action, and decision layers are represented in submatrices A_K, A_A, and A_D, respectively. Edges within layers represent refinement connections, when one element is a small change to a previous element. Edges between layers represent the relationships created by actions that use knowledge and decision elements to develop the knowledge structure. Figure 4.1 shows the general form of the resulting K-A-D network A_{KAD}.

Design activities and their knowledge structure need to be decomposed in to elements to be represented by the K-A-D Framework and network. Knowledge structure elements are roughly self-contained sections of the knowledge, actions, and decisions that are defined by the person investigating the design activity. Referring to definitions used in the studies of design cognition, the types of elements are:

- Knowledge element: coherent segment of knowledge used by the designer (*van der Lugt*, 2001).
- Action element: discrete step taken by the designer that changes the state of the knowledge structure (*Goldschmidt*, 1995; *Cash et al.*, 2015).
- Decision element: commitment to a cohesive segment of the knowledge structure for use in future development, this often applies to knowledge about the elements and relationships in the product structure.

The application of these definitions is subjective and tied to the fidelity of knowledge structure description. For fine-grained descriptions, knowledge elements may represent a word, action elements are captured in sentences, and decision elements describe a verbal commitment. In large-scale design activities, this level of detail is impossible to achieve and a coarse-grained description is necessary. For example, in the design of a naval vessel's hullform, knowledge elements may represent monohull and multihull concepts, hull parameters, and resistances, action elements may represent creating a hullform model or doing engineering calculation, and decision elements may represent the choice to move forward with a particular hullform concept. While this decomposition is subjective and requires a degree of common sense to define elements, it has proven to be an effective method in the study of design cognition (*Dinar et al.*, 2015) and thus is sufficient for modeling knowledge structure development.

The K-A-D network is created through the series of designer actions that occur during the design activity as described by Equation 4.1 and its variants. The general growth functions for knowledge and decisions define how knowledge structures develop by using existing elements and external factors to create new elements. The K-A-D network represents each instance of 4.1 and 4.5 with the following sequence:

1. Designer's action: create a new node in the action layer i.
2. Knowledge structure element usage: add edges A_{ji} from each node, $j = 1 \dots k$, to i.
3. Generated element: create new node l in the knowledge or decision layer corresponding the growth function; G_t for an action or D_t for a decision.
4. Growth relationships: add edges A_{il}, from the designer's action to the new element, and A_{rl} from the refined element (if $r \neq \emptyset$).

If the knowledge structure growth equation introduces a precedent or external knowledge ($k = \emptyset$ and $r = \emptyset$), the representation is simplified to just add the node l without an action node or edges (only step 3). The K-A-D network representation describes the knowledge structure growth through time as a series of interdependent relationships between knowledge, decisions, and the actions that use and create them. The network structure can then be analyzed using the network science toolset.

4.3 K-A-D Framework: Network Analysis

Network analysis quantifies the structure of a network to help understand the represented system's behavior. It is important to note that this means network analysis is only meaningful in the context of what the network represents. With this in mind, network analysis can be applied to the K-A-D network to help understand the design activity through its knowledge structure. Network science provides an expansive toolset of analysis theories and methods that could be applied to the K-A-D network. Thorough reviews can be found in (*Boccaletti et al.*, 2014) and (*Newman*, 2010).

Analysis in this thesis leverages the concept that networks can be measured to identify design drivers. Applying network measures of importance, called centrality metrics, to the K-A-D network enables knowledge, action, and decisions design drivers to be identified. The most basic centrality measure is the degree of a node - the number of edges associated with it. *Out-degree* measures importance of a node by the number of edges going out of it and *in-degree* measures the number of edges pointing towards it (*Newman*, 2010). These can be calculated using the network adjacency matrix and edge notation, $A_{ij} = 1$ if there is an edge from i to j. In- and out-degree can be written

$$k_j^{in} = \sum_{i=1}^{n} A_{ij}, \qquad k_i^{out} = \sum_{j=1}^{n} A_{ij} \tag{4.6}$$

Variants on this type of edge count were used by *Goldschmidt* (1990), *Cross* (1997), *van der Lugt* (2001), *Kim and Kim* (2015), and elsewhere to identify actions or knowledge that drove the design activity. Degree measures of the K-A-D network tells us which aspects of the design process are important to the knowledge structure growth:

- Out-Degree: The number of edges pointing away from the node describes how many times a knowledge structure element is used to grow the knowledge structure.

- In-Degree: The number of edges pointing at a node describes how many knowledge structure elements were used to generate that element.

In-degree and out-degree are straightforward measures of importance, but do not provide any quantification of the temporal nature of the knowledge structure. For example, a knowledge element could be used very frequently in the beginning of a design activity, but then not used later. Early in the activity, that knowledge is likely a design driver, but later it likely is not. However, node degree does not capture that time-dependent dynamic. Hyperlink-Induced Topics Search (HITS) is an extension of degree centralities that quantifies the information content of a node (*Kleinberg*, 1999). In the K-A-D network, HITS can help quantify the importance of knowledge structure elements based on the temporal growth of relationships.

HITS measures importance with reinforcing ideas of hubs and authorities. When the HITS algorithm is run, each node is given a hub and authority score. Important hubs are nodes that point to important authorities. Conversely, important authorities are nodes that are pointed at by important hubs. The corresponding centrality metrics, hub centrality and authority centrality, are the hub and authority scores normalized over all nodes in the network. Authority centrality x and hub centrality y are calculated using

$$x_i = \alpha \sum_j A_{ij} y_j, \qquad y_i = \beta \sum_j A_{ji} x_j, \tag{4.7}$$

where and are constants and A is the network adjacency matrix. Applied to a matrix notation, Equation 4.7 becomes

$$\bar{x} = \alpha \boldsymbol{A} \bar{y}, \qquad \bar{y} = \beta \boldsymbol{A} \bar{x}, \tag{4.8}$$

which can be combined to obtain

$$\boldsymbol{A}\boldsymbol{A}^{\boldsymbol{T}} \bar{x} = \lambda \bar{y}, \qquad \boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A} \bar{y} = \lambda \bar{x}, \tag{4.9}$$

where $\lambda = (\alpha\beta)^{-1}$. This defines $\boldsymbol{A}\boldsymbol{A}^{\boldsymbol{T}}$ and $\boldsymbol{A}^{\boldsymbol{T}}\boldsymbol{A}$ as having the same eigenvalue λ, with their eigenvectors being the authority and hub centralities, respectively.

Identifying network hubs and authorities show the flow of information within its structure. Applying this measure to the K-A-D network identifies parts of the knowledge structure which create important elements and those that use important elements:

- Hub: Measures node importance by the importance of authorities it points to. Thus, in the K-A-D network identifies elements of the knowledge structure that have been used to generate other important elements.
- Authority: Measures node importance by the importance of hubs that points to it. Within the K-A-D network, this identifies knowledge structure elements that were generated by other important elements.

Hub and authority centralities provide measure of recent importance to the design activity. Hub centrality identifies nodes that are important contributors to recent knowledge structure growth. Authority centrality identifies nodes that were recently created by important nodes. For example, the design action of parametrically calculating resistances for a hullform would increase the hub centrality of length, beam, and draft knowledge and create new resistance knowledge with elevated authority centrality.

When nodes are first created, they can only be pointed at by other nodes, increasing their authority centrality. As these nodes are used, their hub centrality increases, but their authority centrality is likely to decrease. If a node is heavily used through a recent part of the activity, it will have high hub centrality because it generates many nodes. These nodes will have high authority scores as a result. However, if that node is no longer used, its hub score will decrease as other nodes are used to grow the knowledge structure. This dynamic allows hub and authority centralities to capture the design drivers for a snap shot of the knowledge structure and measure temporal design driver dynamics when applied to a series of snap shots.

Both degree centrality and HITS analysis enable designer drivers to be measures and path dependencies to be identified. In a design activity, design drivers are elements that dominate the creation of design outcomes. Design drivers are repeatedly used knowledge structure elements that arise during the design activity. Path dependent influences develop when design drivers become ingrained, committing design outcomes to particular solution elements or ideas. In terms of the described network analysis, design drivers are expected to be identified as important nodes in the K-A-D network. Path dependencies are expected to take the form of design drivers that maintain or grow in importance over an extended period of time. For example, knowledge about two battery technologies that are being tested for use in a new vessel would be used frequently, giving them a high hub centrality. The decision to use one battery technology over the other would refer to the knowledge developed about each technology, giving the decision-making action a high authority centrality.

4.4 K-A-D Framework Demonstration

To demonstrate the K-A-D Framework, its network representation, and the described analyses, they are applied to a small design activity. The following example is replicated from (*Goldschmidt*, 2014) and is derived from a recording of an architect named Martin, who was describing his thought process while designing a library site. The activity is broken into three spoken sentences each representing a designer action:

1. G_1: "We start by creating a hierarchy: the large trees, the parking lots, the pedestrians, an entry axis."
2. G_2: "I would then look for a direct relationship between entrance and exterior, because here, the real edge is not this [edge of building], for me it's that [edge of site]."
3. D_3: "I would try to have an important element; would therefore make the axis I mentioned before, this one [points to sketch]."

Goldschmidt (2014) used this example to demonstrate how relationships between designer actions can be described. Here, the example is reused to show how the relationships between knowledge, actions, and decisions can be described. When the

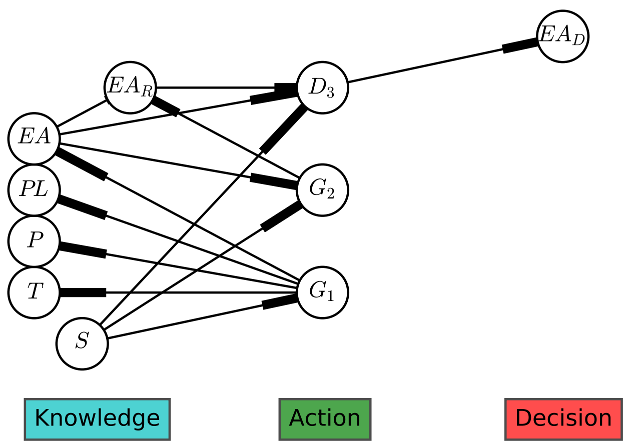


Figure 4.2: K-A-D network visualization for the library design activity. Nodes for each layer are positioned over their respective label.

K-A-D Framework is applied, the activity is first decomposed into elements of knowledge, actions, and decisions. For this activity, the decomposition of knowledge and decision elements is described in Table 4.1. The K-A-D Framework decomposition and growth equations is mapped to a K-A-D network in Figure 4.2. The K-A-D network is analyzed using out-degree, in-degree, hub centrality, and authority centrality whose results are shown in Table 4.2.

Label	Decomposition description	Growth equation
S	*Site architecture* precedent knowledge, generated by an initializing action G_0.	$G_0(\emptyset, \emptyset)$
T	*Trees* existence and location, generated during sentence 1.	$G_1(S, \emptyset)$
P	*Pedestrian* existence and location, generated during sentence 1.	$G_1(S, \emptyset)$
PL	*Parking lot* existence and location, generated during sentence 1.	$G_1(S, \emptyset)$
EA	*Entry axis* existence and location, generated during sentence 1.	$G_1(S, \emptyset)$
EA_R	*Entry axis* refined in reference to exterior, generated during sentence 2.	$G_2([EA, S], EA)$
EA_D	*Entry axis* location decision, generated during sentence 3.	$D_3([EA_R, EA, S], \emptyset)$

Table 4.1: Decomposition of knowledge, action, and decision elements in the library design activity.

Node (layer:label)	Out-degree k^{out}	In-degree k^{in}	Hub y	Authority x
$K:S$	3	0	0.366	0.0
$K:T$	0	1	0.0	0.0
$K:P$	0	1	0.0	0.0
$K:PL$	0	1	0.0	0.0
$K:EA$	3	1	0.385	0.0
$K:EA_R$	1	2	0.165	0.188
$A:G_1$	4	1	0.0	0.145
$A:G_2$	1	2	0.08	0.3
$A:D_3$	1	3	0.0	0.366
$D:EA_D$	0	1	0.0	0.0

Table 4.2: Network analysis results for the library design activity.

K-A-D network degree analysis is straightforward for such a small network, but still useful. Out-degree quantifies how many times an element was used to build the knowledge structure and identified three major contributors: knowledge elements S and EA, as well as action G_1, which were connected to 3, 3, 4 other knowledge structure elements, respectively. Essentially, elements that are referenced many times or actions that create many elements have a high out-degree in the K-A-D network. When out-degree is high, it indicates that the element has a large contribution to the knowledge structure development. The reverse guideline applies to in-degree. Elements that reference many other knowledge structure elements have higher in-degrees, indicating they are heavily interdependent with the existing knowledge structure. For example, D_3, that committed to the entry axis location, has the highest in-degree, $k^{in}_{D_3} = 3$, and use integrates a significant portion of the knowledge structure in a single decision.

Hub and authority centrality provide a different perspective on the knowledge structure. Hub centrality identifies elements that are important to recent knowledge structure growth and picked out two significant nodes, general site knowledge S and the initial entry axis knowledge EA, $y_S = 0.366$ and $y_{EA} = 0.385$. In terms of the design activity, these results are intuitive. Site knowledge, S, facilitates every step of the design and entry axis knowledge, making is a hub of the knowledge structure development. The entry axis knowledge, EA, is the primary designer focus and is used three times in the final two steps the design activity. Authority centrality identifies elements that were recently created from important hubs. This identified two highly scored elements, the actions $x_{G_2} = 0.3$, that refined the entry axis knowledge, and $x_{D_3} = 0.366$, that committed to its location. In both cases, these elements used the most important hubs to develop new knowledge structure elements. The high authority centrality indicates that these elements integrate important parts of the knowledge structure and represent a significant step in the knowledge structure development. It follows that the decision in the final step of the design activity appears as the most important authority.

Comparing degree analysis and hub and authority centrality highlights the differences between them. G_1 has the highest out-degree $k_{G_1}^{out} = 4$, but no hub centrality ($y_{G_1} = 0.0$). This indicates that while it generated a significant number of elements, it is not important to the current knowledge structure growth. This illustrates that degree and HITS analysis provide contrasting perspectives on the node's importance. Each perspective is based on how they quantify the knowledge structure's K-A-D network: Degree analysis is time-independent and HITS is highly time-dependent. This can be useful in understanding how and why certain knowledge structure elements are design drivers. Using the library example, in terms of contribution to the design activity G_1, which generated the initial knowledge structure, a is design driver, but the entry axis knowledge EA ($y_{EA} = 0.385$) may be a more important driver for the designer's immediate actions.

The analysis demonstrated here can be applied during a design activity to measure the importance of specific elements to the knowledge structure. This information can be tracked over time to identify when and how path dependencies develop. The following sections explore this type of temporal knowledge structure analysis.

CHAPTER V

Testing the K-A-D Framework

The primary objective of K-A-D Framework is to enable designers to understand how path dependencies will influence design outcomes towards success or failure This chapter tests the K-A-D Framework on an agent-based design simulation that learns, solves problems, and makes design decisions. This agent designer is programmed to exhibits path dependent behavior towards its solution. Analyzing the agent's design activity with the K-A-D Framework and network demonstrate that path dependent design behaviors can be identified by tracking knowledge structure development and measuring design drivers.

Simulation tools have previously been used to explore the influence of organizational structure, design approach, and problem structure on design outcomes (*Parker*, 2014; *Austin-Breneman et al.*, 2015; *Sobieszczanski-Sobieski and Haftka*, 1997). In this chapter an iterative design activity is simulated by an agent solving the Traveling Salesman Problem (TSP). The TSP asks for the shortest route between a list of cities that visit each city exactly once. This represents one of the most difficult combinatorial optimization problems and is NP-Hard. In this experiment, it provides a difficult close-form problem for the agent designer to learn and solve. Analyzing how the agent designs a solution to the TSP with the K-A-D Framework provides a baseline for one to examine design knowledge structures with the K-A-D Framework.

To simulate a design activity, the agent designer will explore routes through the cities, learn the path distance between cities, develop preferences towards promising paths, and make decisions to specify its route solution. The route solution specified by these decisions and is the agent's product structure. The agent's knowledge structure is the information it uses to learn paths, develop preferences, and make decisions. The K-A-D Framework is used to track knowledge structure development during the design activity simulation. Using the K-A-D network representation, this captures the learning and decision-making that enables the agent designer to create a solution to the TSP. Four experiments are conducted to designer behaviors with different degrees of path dependence. These experiments model design by simulating different problem-solving strategies develop knowledge to define design solutions. Analyzing the resulting K-A-D networks illustrate that design drivers can be measured and used to identify when path dependencies emerge.

Section 5.1 describes the design activity simulation and the designer agent, which is based on Ant Colony Optimization methods. Section 5.2 defines the K-A-D Framework mapping for the simulated design activity. Section 5.3 describes the expected designer behaviors and corresponding knowledge structure characteristics. Section 5.4 described and conducts four experiments that test the K-A-D Frameworks efficacy. Section 5.5 concludes the chapter.

5.1 Simulation Structure

To simulate a design activity, the agent designer is given a TSP with a known number of cities and a fixed starting point, but no information about the distance between each city. During the simulation, the agent creates a route through the TSP cities in a series of timesteps. After a route is created, the designer then calculates the route's length and records its preference towards the paths used in the route based on that length. If previously unused paths between cities are used in the route, the path distances are learned by the agent. In subsequent routes, path preferences and learned distances are used to guide the route creation. When the designer develops a strong preference for a path, it makes a decision to always include that path in its route. Over the course of the simulation, these decisions specify the product (TSP route solution) defined by the designer. The general structure of the simulation is shown in Figure 5.1.

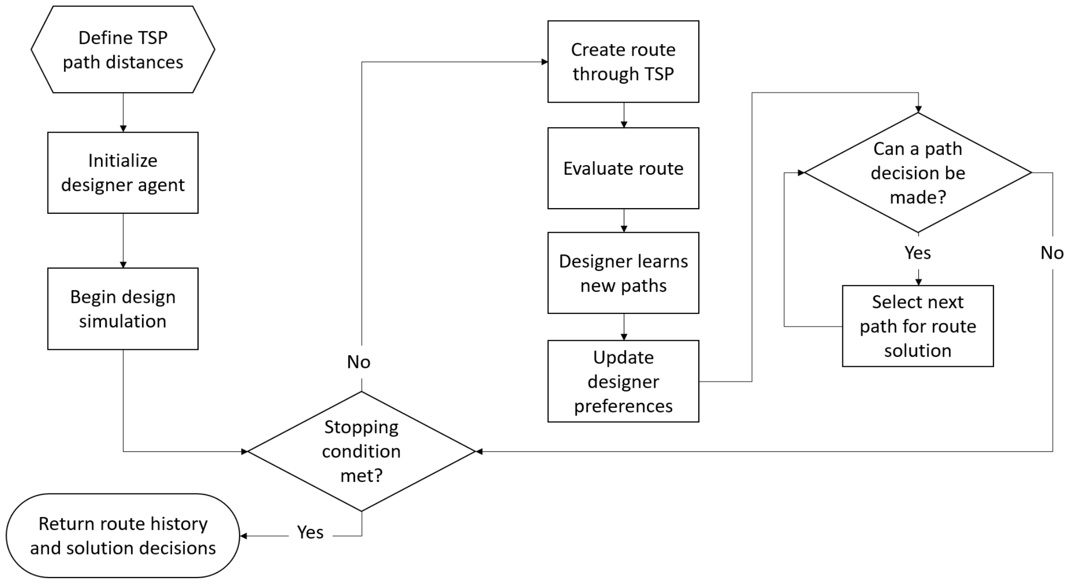

Define TSP path distances
Initialize designer agent
Begin design simulation
Stopping condition met?
No
Yes
Return route history and solution decisions
Create route through TSP
Evaluate route
Designer learns new paths
Update designer preferences
Can a path decision be made?
Yes
No
Select next path for route solution

Figure 5.1: Design simulation structure.

For the ease of K-A-D analysis and the comparison of results, the experiments use a specially formatted TSP. For a TSP with N cities, the distance d_{ij} of the path between City i and j is defined as

$$d_{ij} = \begin{cases} 1, & \text{if } i \text{ or } j = 0, \\ |i-j|, & \text{otherwise,} \end{cases} \tag{5.1}$$

where City 0 is the starting city for every route. In this TSP, the solution route must visit all cities once and does not return to the starting city. Thus, two shortest routes exist, one goes from City 0 to City 1 and moves to the city at the next largest location until arriving at City N, the other goes from City 0 to City N and moves to the next smallest city location until arriving at City 1. The length of both routes is $N-1$. Any deviation from these two routes will incur increased route distance.

Path distances are fixed, but when the design activity is initialized, the path distances are unknown to the designer. Before the designer creates a route containing path d_{ij}, the agent believes the path length is e, a simulation parameter. Once a path is included in a route, the agent learns the true path distance defined in Equation 5.1. Figure 5.2 shows how this occurs in the first simulation timestep.

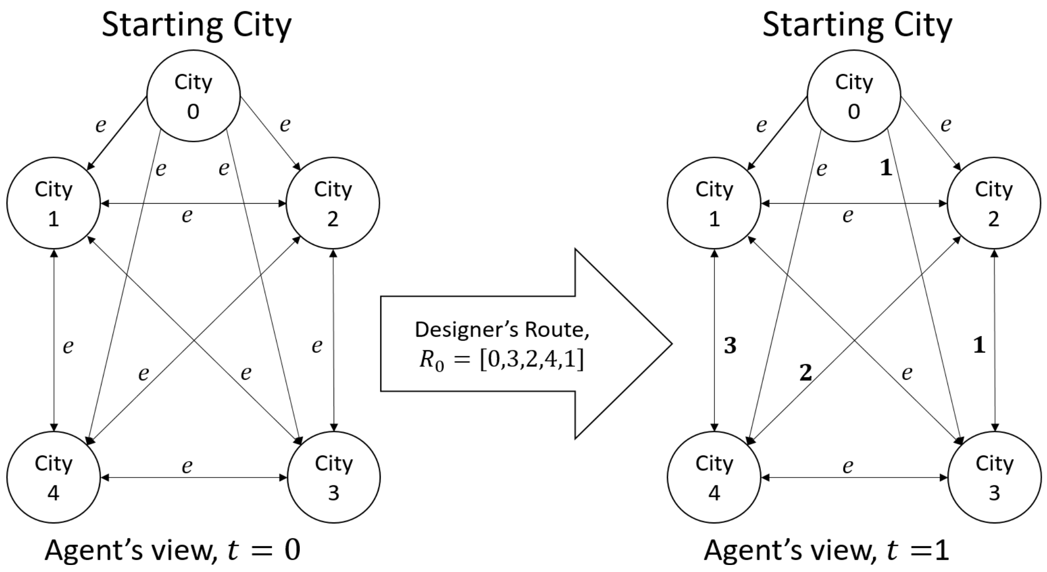


Figure 5.2: Designer path learning in the simulated design activity with 5 TSP cities. Initially, the designer believes that all path distances are e. Path distance d_{ij} is learned after a path is used in a route.

The designer agent is implemented using Ant Colony Optimization (ACO) methods. ACO methods are a class of biology inspired genetic algorithms originally developed to solve the TSP (*Dorigo et al.*, 1996). Ant agents explore the possible solution paths through the TSP based on digitized pheromones τ and local heuristics η. Here, a variation will be used to better model a designer. *Dorigo et al.* (1996) provides a thorough description of the original ACO methods and algorithms.

Pheromones are a mechanism to encode agent preference to parts of a solution. This is similar to how a designer will prefer certain solution elements based on the designer's experiences and biases, as well as the solution element's perceived quality. Each time a path is used in a route its pheromone level is increased. The pheromone is increased relative to the overall route length. This mechanism was chosen because it models designer's behavior during solution development - frequency of use indicates designer preference *Goldschmidt* (2014).

The local heuristic is based on the action of choosing a single path within a route. Given a choice of possible paths from a city, the heuristic favors shorter paths. This mechanism allows the simulation to model the effect of designer learning - if the initial path distance e is low, the designer is likely to favor exploring new paths versus exploiting known paths.

Solution routes are created through repeated probabilistic steps where the designer moves from its current city in the TSP to another unvisited city following the transition rule,

$$p_{ij} = \frac{\tau_{ij}^{\alpha} \cdot \eta_{ij}^{\beta}}{\sum_{h \in \Omega} \tau_{ih}^{\alpha} \cdot \eta_{ih}^{\beta}}, \tag{5.2}$$

where p_{ij} is the probability that the designer goes from City i to City j, τ_{ij} is the pheromone level on path (i, j) and is scaled by the factor α. η_{ij} is the local heuristic $1/d_{ij}$ which is based off of the designer current belief about the distance of path (i, j) and is scaled by the factor β. Ω is the set of cities that the designer has not visited on this route.

After an ant creates a solution, the total route distance Γ is evaluated with the true path distances in 5.1 and the designer learns the distance of any newly used paths. Then, the designer's path pheromones are updated following the rule,

$$\tau_{ij}' = (1 - \rho) \cdot \tau_{ij} + \Delta\tau_{ij}, \tag{5.3}$$

where the pheromone level on every edge is dissipated by the factor ρ and paths in the designer's route are incremented by $\Delta\tau = 1/\Gamma$. After each pheromone is updated, τ' replaces the previous pheromone values. This process reinforces pheromones for the last path based on the route length and guides subsequent transition rules.

In this simulation, the designer makes a decision about a solution element when pheromones are strong enough. Designer decisions are controlled by a probability convergence threshold c. Decisions specify a path in the agent designer's solution that will be used in all future routes. This models design decisions, which are a commitment to an element or relationships in the product structure. Generally, the decision rule for City i is defined as

$$X_1 - X_2 \geq c, \tag{5.4}$$

where X is a descending list of transition probabilities from City i, X_1 is the first elements in the list, and X_2 is the second element in the list. If 5.4 is satisfied, the designer decides that the path represented by the transition probability X_1 is an element of the route solution (design product structure).

Because City 0 is the fixed starting point, the first path decision is selected from City 0 by applying Equation 5.4. Using a modification of Equation 5.2 to only consider pheromones, the possible transition probabilities p_{0j}, $j = 1 \ldots N$, are put into the descending list X. If the first element X_1 is c greater than the second element X_2, a decision is made to always include the path $(0, k)$, where k is represented by X_1. Only pheromone values are used in this calculation to prevent the heuristic from distorting the calculation of the agent designer's preference.

When Equation 5.4 is satisfied and a decision is made to include path (i, j) in the solution, it is locked-in by adjusting path pheromones. This models the agent's commitment to the solution element. The pheromones are set to 0.0 on all paths from City i that are not path (i, j). This ensures the designer will only travel to City j

from City i because $p_{ij} = 1.0$. The decision rule is applied each timestep, after new paths are learned and the pheromones are updated.

Equation 5.4 is applied starting with City 0 and if it is satisfied, a decision is made to select the preferred city. Then, Equation 5.4 reapplied to the next city in the route. This continues until an entire route is decided or the decision criteria is not satisfied. The decision process is then applied in the next timestep starting from the last city in the decided route. If the simulation stops without completely specifying a route, the paths used in the last timestep are defined as the set of design decisions.

5.2 K-A-D Framework Network Mapping

The simulated design activity is tracked through the K-A-D Framework's network mapping. This section goes through the steps of the simulation described in Figure 5.1 and defines their mathematical growth equations in the K-A-D Framework. The K-A-D network can then be constructed from those equations and analyzed during the design activity simulation. Figure 5.3 visualizes the application of these growth equations in steps 2-5.

1. *Initialize agent*: The designer begins the simulations with knowledge that N cities exist and there are paths between each city with distance e. This is modeled as precedent knowledge in the K-A-D Framework, $[P_{01}, \ldots, P_{ij}] = G_0(\emptyset, \emptyset)$, where P_{ij} represents the agent designer's knowledge about the path distance between City i and City j.

2. *Create route*: In the simulation, a route is created by iterating Equation 5.2 to select paths through the TSP. At each path selection, the agent designer uses knowledge for about path distance corresponding to the set of unvisited cities in the route Ω and knowledge about the previous paths selected in the route. The selection action for path s in the current route generates knowledge

about the route created at the current timestep t, R_{st}. Using this description, path selection s from City i to City $j \in \Omega$ for timestep t is modeled as, $R_{st} = G_{st}([R_{1t}, \ldots, R_{(s-1,t)}, P_{ij}, \ldots, P_{i\Omega}], \emptyset)$.

3. *Evaluate route*: After a route is completed, the designer evaluates its distance using the series of path selection steps s that defined the path. This action generates knowledge about the route score for that timestep and is modeled as, $S_t = G_{St}([R_{1t}, \ldots, R_{st}], \emptyset)$. If a decision has been made about a path in the route, the decision element is used in place of the corresponding route selections and path knowledge.

4. *Learn new paths*: If the agent uses a previously unvisited path in the route for timestep t, the agent learns the path's true distance. For the path from City i to City j, this is modeled as a knowledge refinement that uses the previous path knowledge P_{ij} and the route selection knowledge R_{st} that used the path and is thus, $P_{ij} = G_{ijt}([P_{ij}, R_{st}], P_{ij})$.

5. *Make decision*: If a path selection has sufficiently converged for the agent to make a decision, the agent takes a decision-making action that creates a decision committing it to a path in the TSP solution. If this is the first decision, a new decision node is created. If it is not the first decision, the previous decision is refined. The decision-making action to include the path from City i to City j as the dth path in the solution route uses the set of path from i to all cities Ω that are not included in decided paths. Using this description the action that creates the decision for the dth path is, $\Phi_d = D_{dt} = ([P_{ij}, \ldots, P_{i\Omega}], \Omega_{d-1})$, where Φ_{d-1} is the previous decision or $\emptyset$ if Φ_d is the first decision.

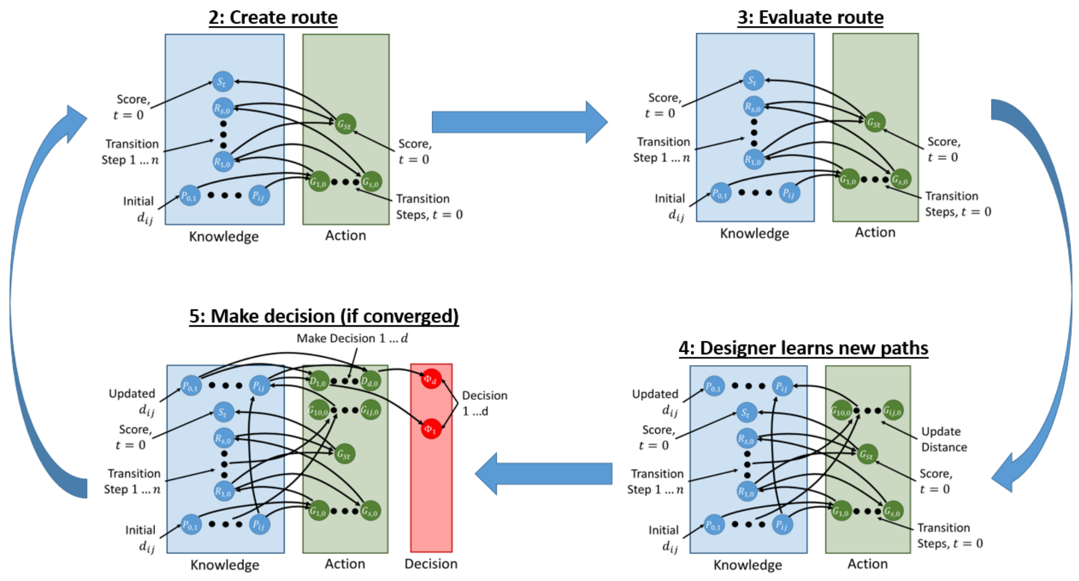


Figure 5.3: Development of the K-A-D network for simulation steps 2-5. These steps are repeated during each simulation timestep.

5.3 Expected design drivers and path dependencies

The definition of TSP cities, distances between cities, and agent behavior encode a repeatable pattern of path dependencies that ingrain design activity outcomes. During the design activity, the pheromone mechanism creates preferences toward certain solution elements. This process is path dependent, which should be reflected in the design drivers, knowledge structure elements that are important to knowledge structure development. If analysis of the K-A-D network can identify path dependencies through the design drivers, it will be a step towards demonstrating that the proposed methods are able to help designers understand how their design outcomes develop. In larger and more complex design activities, this modeling and analysis will enable the measurement of design outcomes and the prediction and prevention of design failures through the identification of path dependencies.

In the simulated design activity, solution path dependence comes from the ACO pheromone mechanism. Paths with high levels of pheromones are more likely to be used throughout the design activity than paths with low levels of pheromones. This makes convergence on a route solution a path dependent process, which is governed by the ACO pheromone update in Equation 5.3. To see this, consider a path that is used once in two sequential route creations. Assuming both routes are of equal length and dissipation factor, ρ, used in Equation 5.3 is not 0.0 or 1.0, the pheromone level on that path will be different if the path is used in the first route or in the second. For example, if $\tau_0 = 0.5$, $\Gamma = 5$, $\rho = 0.2$; $\tau_2 = 0.48$ if the path is used first and $\tau_2 = 0.52$ if the path is used second.

Path dependencies propagate in the sequence of route creation during the simulation. Based on the order of route creation, certain paths will be more likely to be used in subsequent routes. In the timespan of the design simulation, this defines the route generation and solution convergence as path dependent processes. In terms of the design activity simulation, designer knowledge corresponding to preferred paths are expected to be design drivers. Similarly to a designer preferring to use knowledge about favorable solution, **the knowledge associated with paths that have a high level of pheromones will be used more frequently**. As path dependencies that converge the ACO develop, the knowledge used to create those path dependencies is expected to maintain their importance to knowledge structure development. This should be reflected in the K-A-D network analysis.

Setting a fixed starting city creates a secondary type of design driver. The agent's first step in every route is to move from City 0. To make this move, the designer takes an action using its knowledge about the paths that originate from City 0. Because this exact action happens every time a route is created, designer knowledge and actions pertaining to starting paths should also be design drivers. However, when the designer makes a decision, this set of design drivers should shift. In the simulation, a decision removes the reliance on knowledge for a path selection action. The first decision always specifies the path taken from City 0. Thus, this decision will shift importance from knowledge about starting paths to knowledge about paths leaving from the second city in the route.

The following section will implement the design activity simulation in four cases that exhibit different degrees of path dependence. Examining the corresponding knowledge structures and comparing their differences will demonstrate that the K-A-D Framework and analysis can identify path dependencies in the simulated design activity by measuring design drivers in knowledge structure development.

5.4 Results and Discussion

Design simulation was used to conduct four experiments that illustrate how the K-A-D Framework can be applied to identify design driver and path dependencies in the knowledge structure. The experiments are described below and their parameters are shown in Table 1:

1. Random: Parameters are set so the agent designer creates random routes at each timestep, simulating an unintelligent designer that defines a product randomly. This provides a baseline analysis of the K-A-D network, which will be used as a comparison for other designer behaviors.

2. Initial Route Dependence: Parameters are set so the agent designer immediately locks into the first route taken and only uses that route in subsequent timesteps. This simulates a designer that fixates on the first solution they come across. The result is an extreme case of path dependence that illustrates how path dependencies can be identified through K-A-D network analysis.

3. No Decision: Parameters are set so the agent designer learns paths, explores possible routes, and converges on a solution, but does not make decisions committing to a solution route until the final timestep. This enables analysis of

design drivers as they reach a converged steady state, without the distortion of decisions on the knowledge structure development.

4. Decision: Parameters are set the same as the No Decision experiment, except the agent designer makes path decisions when preference reaches a threshold. Comparing the Decision and No Decision experiments demonstrates that the K-A-D network analysis can identify shifting design drivers and path dependencies.

Simulation name (label)	N	c	ρ	e	α	β
Random (R)	6	1.0	0.0	1.0	0.0	0.0
Initial Route Dependence (IRD)	6	1.0	1.0	1.0	1.0	1.0
No Decision (ND)	6	1.0	0.2	1.0	1.0	1.0
Decision (D)	6	0.7	0.2	1.0	1.0	1.0

Table 5.1: Designer agent behavior parameters.

The first two experiments bound the study by exploring edge cases, one where path dependencies do not exist and the other where they immediately exist. The third experiment looks at a case where the designer is learning and problem-solving. In this experiment, the emergence of design drivers and path dependencies can be observed. The last experiment demonstrates how decisions can shift design drivers and how this shift can be measured. Testing under these parameters showed that the agent converged to a solution by the 50th timestep. To ensure that convergence was captured, each experiment instances were run for 60 timesteps. For each set of experiment parameters, the experiment was simulated 100 times so that trends in agent behavior could be identified and compared statistically. Figure 5.4 and the optimality results in Table 5.2 demonstrate the agent designer's ability to generate an adequate solution to the TSP, when simulation parameters allow. Each K-A-D network created by a simulation run is analyzed with the four centrality metrics described in Section 4.3: In-degree, out-degree, hub centrality, and authority centrality. Results are aggregated over the 100 simulation instantiations.

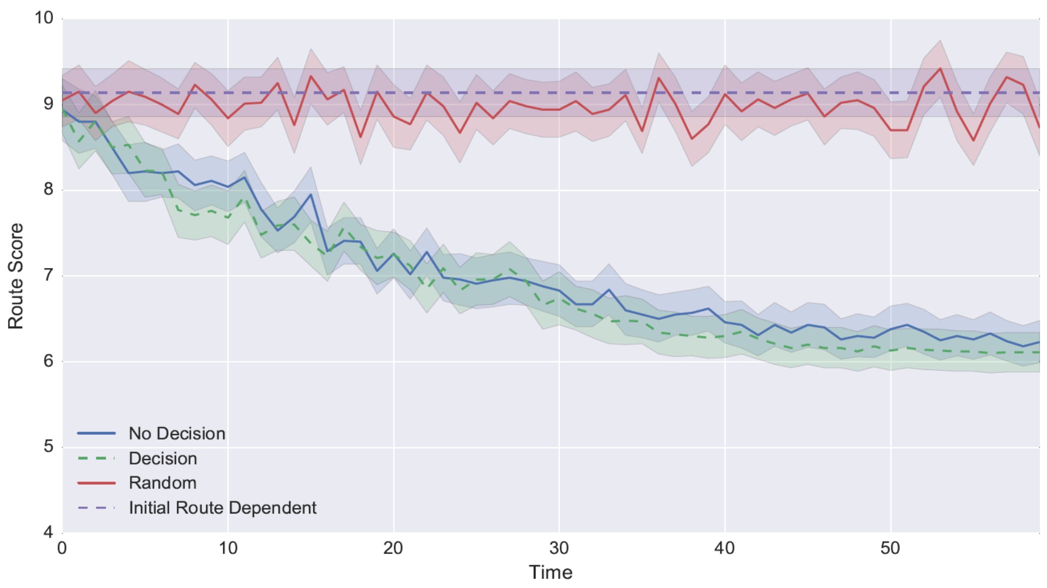


Figure 5.4: Agent's solution quality in 6-City TSP (95% confidence interval (CI) for mean); progression of the routing length during the design activity timesteps.

Simulation name (label)	Runs converged on the optimal (%)
Random (*R*)	2
Initial Route Dependence (*IRD*)	1
No Decision (*ND*)	37
Decision (*D*)	45

Table 5.2: Optimality results for experiments after 60 timesteps.

Figure 5.4 demonstrates that each experiment is behaving as intended. In the 6-City TSP with distances defined by Equation 5.1, the two optimal paths are $0-1-2-3-4-5$ and $0-5-4-3-2-1$, with a total length $\Gamma = 5$. In general, the agent is not guaranteed to converge on the optimal due to the nature of the ACO heuristic (*Dorigo et al.*, 1996), which is known to get stuck in local optima. Furthermore, the parameters in the *R* and *IRD* both prevent the designer from improving its solution by preventing the agent from learning. In contrast, *ND* and *D* experiments showed that the agent is capable of improving its routing, converging to the optimal solution 37% of the time in *ND* and 45% of the time in *D*. The optimization improvement seen in *D* is attributed to the decision-making behavior, which allows more focused optimization on the a smaller selection of paths. Both the *ND* and *D* results show that the design can uncover TSP path distances and use them to incrementally reduce the route length over the simulation. This indicates that the knowledge structure captured in the K-A-D Framework represents the design behaviors of an agent that is learning, problem-solving, and making informed decisions. The remainder of this section discusses the knowledge structure patterns that arise from these behaviors.

Discussed in Section 5.3, the TSP solution design activity is expected to have two types of design drivers. First, as the designer converges on a solution, the knowledge elements associated with preferred paths will be drivers. Second, knowledge and action elements associated with the starting paths will be drivers. After the first path decision is made, it is expected that these drivers will shift from starting paths to other TSP paths.

The first type of design drivers can be measured by the distinction between knowledge for TSP paths in the designer's final solution and paths that are not in the final solution. To properly compare these segments, the K-A-D network knowledge layer nodes P_{ij} are separated into the following groups based on their location in the TSP and if their represented paths were included in the designer's final route R_f:

- Starting Path, Solution: Path that leaves from City 0 and is in the final route, $P_{(0,\cdot)} \in R_f$.
- Starting Path, Non-Solution: Paths leaving City 0 and not included in the final route, $P_{(0\cdot)} \notin R_f$.
- Non-Starting Path, Solution: Paths that do not include City 0 and are in the final route, $P_{ij} \in R_f; i, j \neq 0$.
- Non-Starting Path, Non-Solution: Paths that do not include City 0 and are not in the final route, $P_{ij} \notin R_f; i, j \neq 0$.

These segments were evaluated using out-degree and hub centrality for the first three experiments, Random, Initial Route Dependence, and No Decision, which each elicit different convergence behaviors. The network measures at each timestep were averaged over the nodes within the segment and the results were aggregated over all simulation runs.

In the Random experiment, the agent creates a solution at random. Thus, there is no convergence in the agent's behavior and there no difference between the knowledge represented in the solution and the knowledge that is not represented in the solution. The result is seen in hub centrality analysis in Figures 5.5 and 5.6 that show near identical results for the importance of solution and non-solution segments to knowledge structure development. Because the designer is unable to use previous routes to learn and improve route generation, solution and non-solution knowledge is used identically by the designer agent. This means that, besides variations caused by the number of knowledge elements in each segment, there no meaningful differences between them.

In Figure 5.5, starting path knowledge has low initial importance to knowledge structure development. Its importance grows as the knowledge is used for route generation and distance learning. The agent's distance learning actions use the initialized path knowledge to learn the true distance between cities. These learning action increase the importance of the knowledge they use and cause the ruggedness in the curves that are highlighted in the red circle. Starting path knowledge importance levels off as the knowledge structure development reaches a steady state, where the knowledge is used equally in design development. The initial difference in starting path importance between the two segments is caused by the number of knowledge elements in the segments. In each experiment, the starting path solution segment has one knowledge element and the non-solution segment has four. Thus, it is more likely that the agent uses path knowledge from the non-solution segment than the

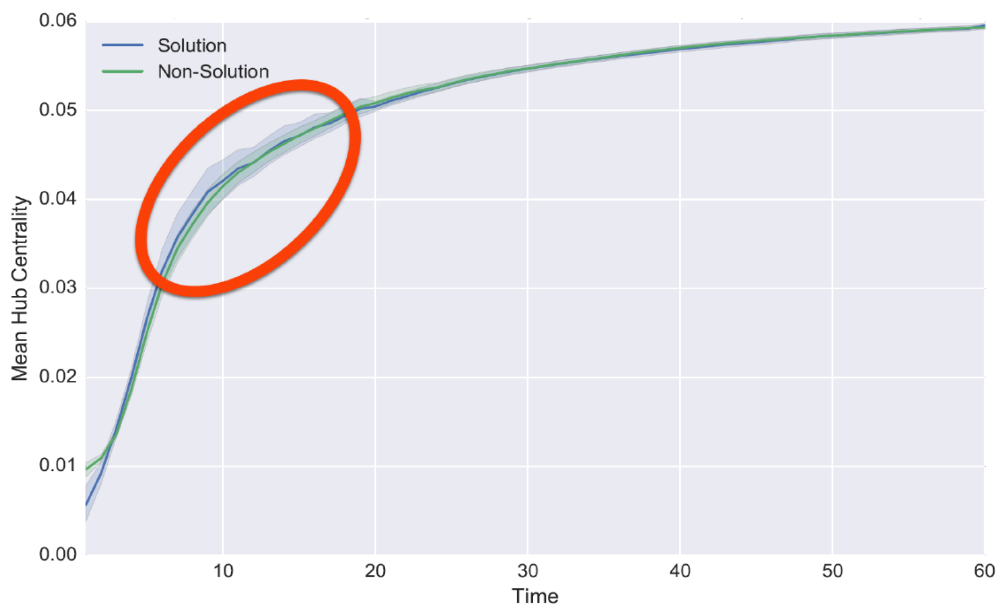


Figure 5.5: Importance of starting path knowledge (95% CI for mean); hub centrality in the K-A-D network based on inclusion in the agent designer's solution for the R experiment. The area circled in red highlights where learning actions are influencing the importance of knowledge elements.

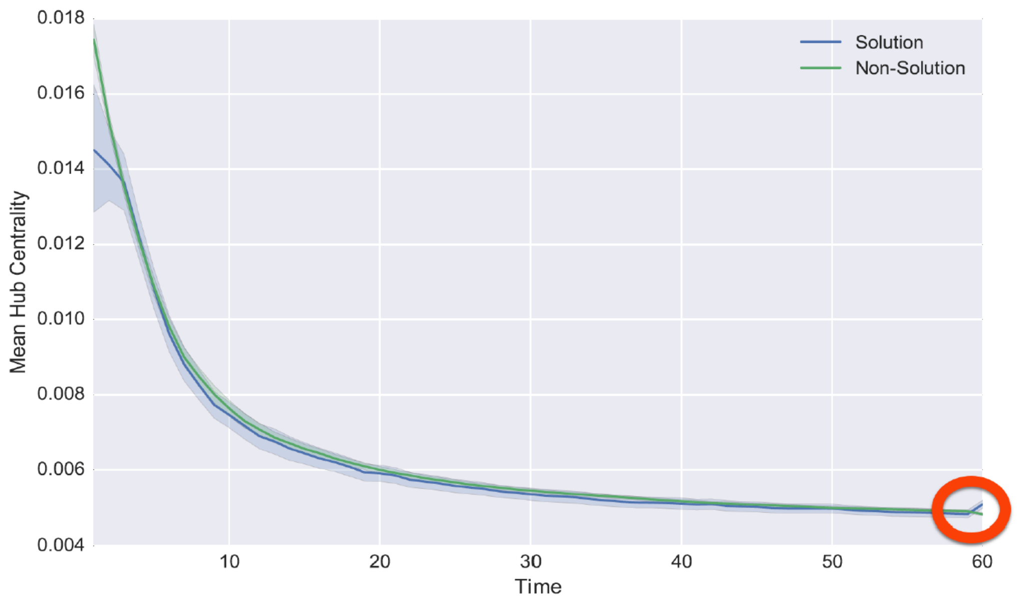


Figure 5.6: Importance of non-starting path knowledge (95% CI for mean); hub centrality in the K-A-D network based on exclusion from the agent designer's solution for the R experiment. The area in red highlights where the stopping criteria raises the importance of the knowledge elements in the solution segment during the last timestep.

solution segment. This causes the non-solution segment to grow in importance faster than the solution segment for the first few timesteps. Once, knowledge elements from both segments are used, they become identical.

In Figure 5.6, non-starting path knowledge has relatively high initial importance that decreases over time. The initial importance is caused by the small amount of knowledge in the knowledge structure when the design activity begins. When there is a small number of knowledge elements, those that are used have a high importance. However, as the knowledge structure develops, the non-starting path knowledge is used less often than the starting path knowledge. This lowers the importance of non-starting path knowledge to the knowledge structure development, which is reflected in the centrality results. The difference in means and 95% confidence intervals between solution and non-solution knowledge, is caused by the difference in the number of knowledge elements in each segment. The solution path knowledge segment contains four knowledge elements and the non-solution knowledge element segment contains 16. This reduces the variance of the non-solution segment relative to the solution segment and can produce differences in segment means over short timespans. The difference between segments in the final timestep, highlighted in the red circle, results from the route solution being defined by the final route produced in the design activity. This means the knowledge that supports the final route will have relatively higher centrality for the last timestep.

In the Initial Route Dependence experiments, there is an extreme difference between segments because the solution is completely path dependent based on the first route. This immediately makes the path knowledge that creates the first route knowledge structure design drivers, see Figure 5.7. Hub centrality analysis shows marked difference between solution and non-solution knowledge because the agent fixates on a solution after the first route. The solution path knowledge clearly appears as a design driver with high network importance relative to non-solution path knowledge.

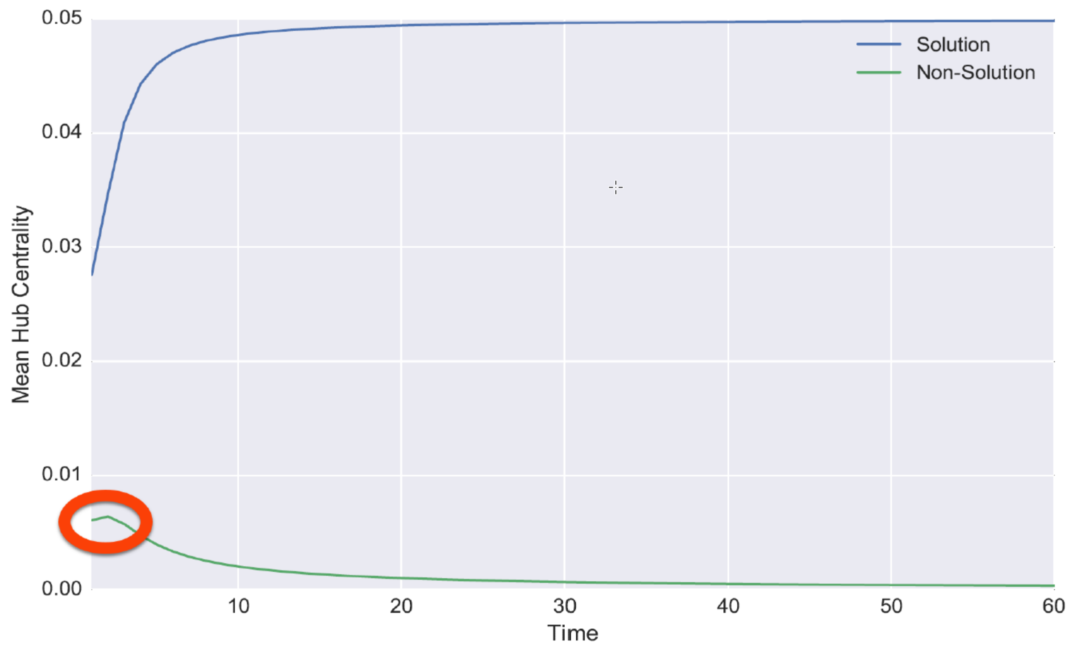


Figure 5.7: Importance of starting path knowledge (95% CI for mean); hub centrality in the K-A-D network based on inclusion in the agent designer's solution for the *IRD* experiment. The area highlighted in red shows where the creation of the first route leads to briefly sustained importance in the Non-Solution segment.

While the non-solution starting path knowledge is used by the agent to create the first route, that knowledge is not used in subsequent routes. This creates the briefly sustained importance, highlighted in red, that quickly decreases as the non-solution knowledge is goes unused. Similar design driver behaviors in the non-starting path knowledge is shown in non-starting path knowledge. However, in the non-starting path knowledge segment, the large number of knowledge elements reduces the initial importance of non-solution knowledge seen in Figure 5.7.

The No Decision experiment exhibits controlled solution convergence, which is reflected in the K-A-D network analysis. Figure 5.8 demonstrates that hub centrality analysis captures the divergence of knowledge importance as the agent develops path preferences. When the agent develops a preference towards a path, the corresponding knowledge is used more frequent to develop solutions. This causes that knowledge to increase in importance within the knowledge structure development. The result is that knowledge structure development is highly influenced by the knowledge that is included in the solution. This is reflected in the statistically significant difference between solution and non-solution segments for the 95% confidence intervals shown in the graphs. The larger confidence interval for the non-solution elements in Figure 5.8 is caused by variations in the decrease of non-solution knowledge importance. Some knowledge may become immediately unused by the agent and other knowledge may be used frequently until convergence occurs. Conversely, in Figure 5.9 the solution path knowledge has a larger confidence interval than non-solution knowledge. This occurs because the agent converges on non-starting paths over different timespans. The agent may develop preferences toward some non-starting paths quickly and other preferences may be delayed.

Figure 5.10 and 5.11 shows similar results in out-degree analysis. The divergence of knowledge importance occurs as the designer develops path preferences. The results show that designer actions used solution path knowledge often than non-solution path knowledge. This difference starts early in the design activity and increases during the design activity. This is seen though the statistically significant difference between solution and non-solution segments for the 95% confidence intervals shown in the graphs.

In both starting and non-starting path knowledge, there are no clear design drivers when the design activity begins. This resembles the Random experiment results. As the agent designer develops preferences, design drivers emerge in the knowledge structure development. This captures path dependence in the solution generation. When the path dependencies develop, the network importance begins to move towards the Initial Route Dependence results, where solution knowledge has higher importance than non-solution knowledge.

Design drivers can be shifted by designer decisions that commit to the product

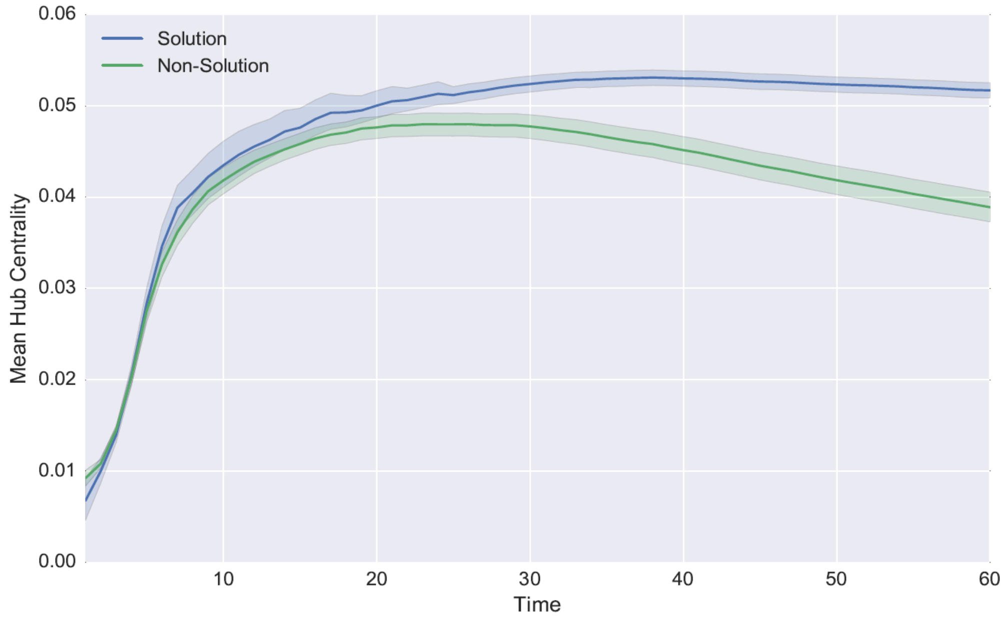


Figure 5.8: Importance of starting path knowledge (95% CI for mean); hub centrality in the K-A-D network based on inclusion in the agent designer's solution for the *ND* experiment.

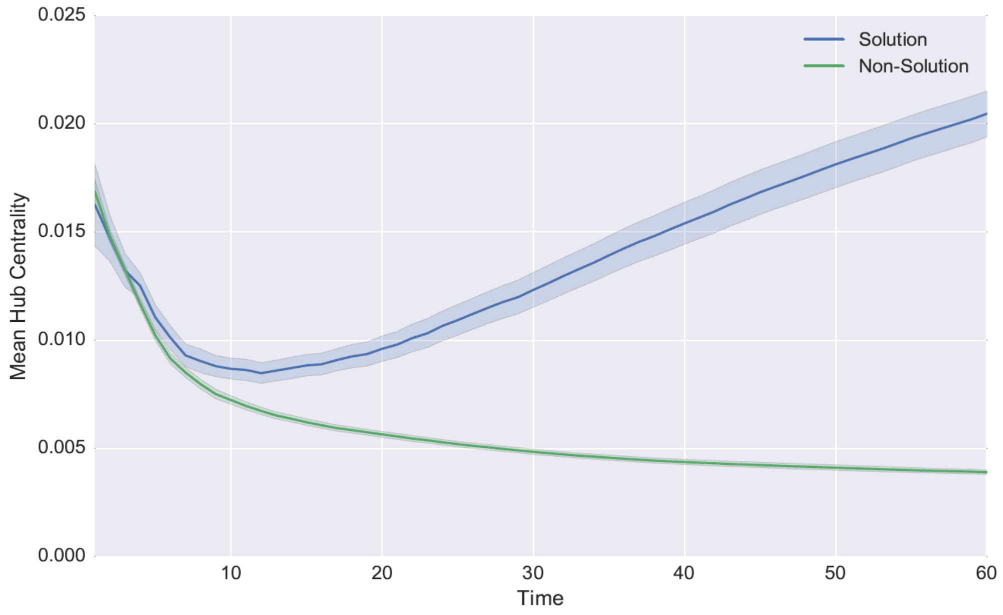


Figure 5.9: Importance of non-starting path knowledge (95% CI for mean); hub centrality in the K-A-D network based on exclusion from the agent designer's solution for the *ND* experiment.

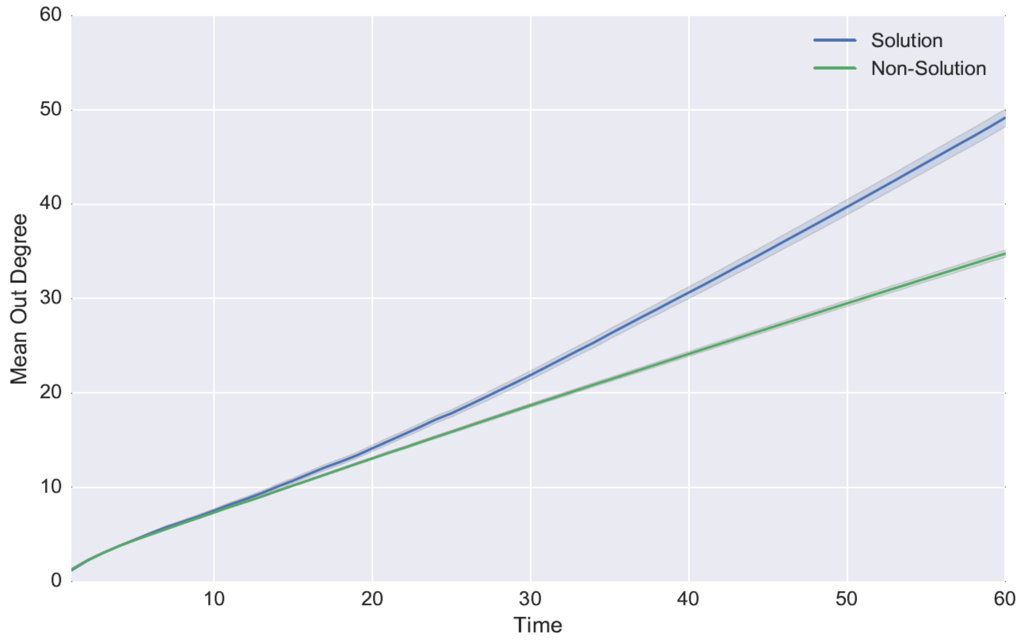


Figure 5.10: Importance of starting path knowledge (95% CI for mean); out-degree in the K-A-D network based on inclusion in the agent designer's solution for the *ND* experiment.

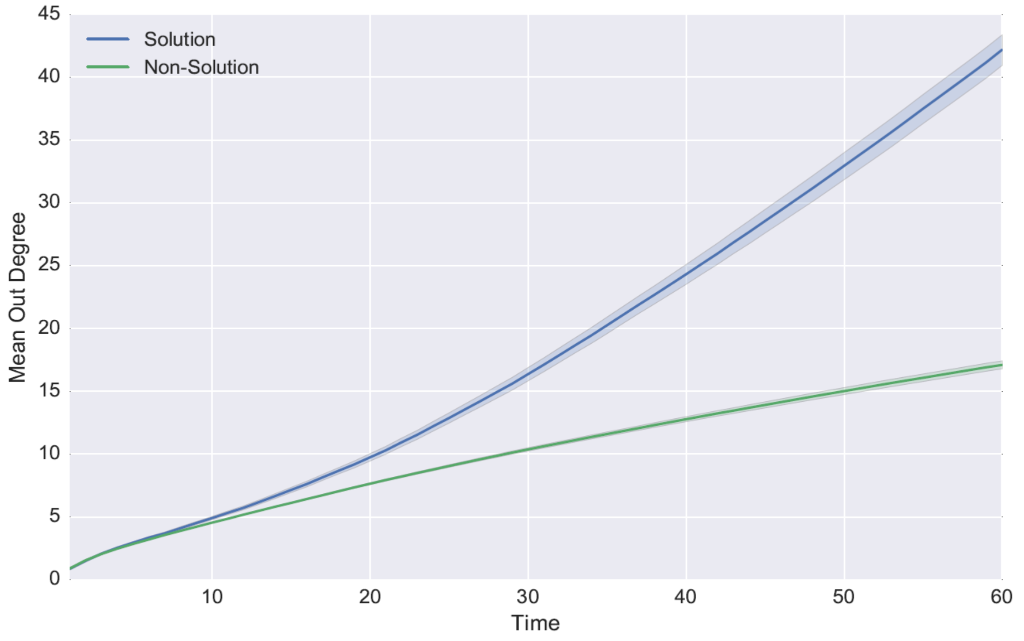


Figure 5.11: Importance of non-starting path knowledge (95% CI for mean); out-degree in the K-A-D network based on exclusion from the agent designer's solution for the *ND* experiment.

structure and change the knowledge structure development. In these experiments, this design driver behavior is created by the order of path selection actions. Because the agent always starts the route at City 0, the path knowledge $P_{0\cdot}$ has greater influence than non-starting path knowledge. This difference can be confirmed by measuring starting and non-starting path knowledge and actions that generate that knowledge. Comparing these types of knowledge structure elements is best done by grouping parts of the K-A-D network into four segments:

1. Starting Path Knowledge - knowledge layer nodes representing paths $P_{0\cdot}$.
2. Starting Path Actions: action layer nodes representing learning action for starting paths. These nodes represent the action $G_t([P_{0j}, R_{st}], P_{0j})$, as described by K-A-D network mapping in Section 5.2.
3. Non-Starting Path Knowledge: knowledge layer nodes representing non-starting paths $P_{ij}; i, j \neq 0$.
4. Non-Starting Path Actions: action layer nodes representing learning action for non-starting paths. These nodes represent the action $G_t([P_{ij}, R_{st}], P_{ij})$, where $i, j \neq 0$ as described by K-A-D network mapping in Section 5.2.

To quantify segment differences, hub and authority centralities were applied to the K-A-D network and their results were averaged over the nodes in each segment. Figures 5.8 and 5.10 show the results for the No Decision experiment. The authority centrality results for knowledge layer nodes and hub centrality results for action layer nodes are not displayed because they are negligible. In Figure 5.12, mean of authority centrality for actions that learn path distances have some initial importance, but decrease quickly as the design activity progresses. Early in the design, actions that learn the true path distances are identified as important actions because they use important knowledge. However, the actions quickly became unimportant as the

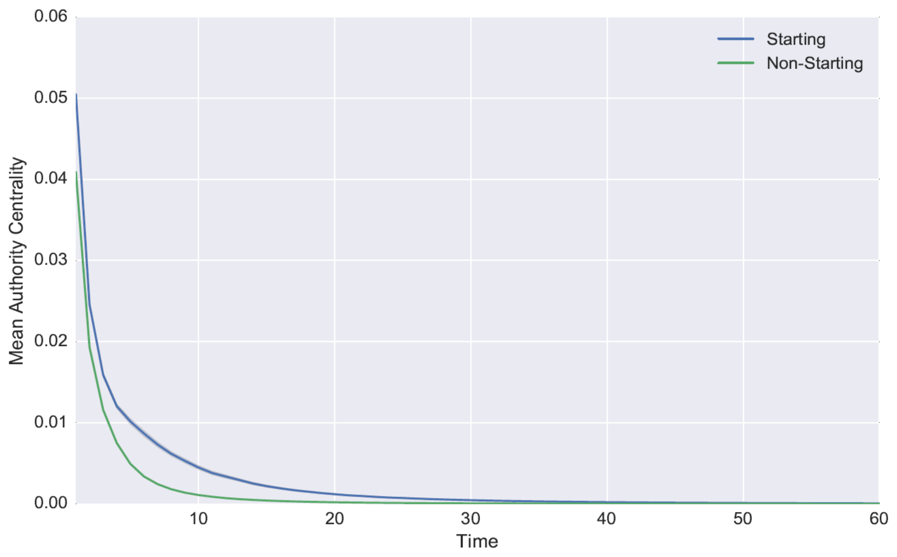


Figure 5.12: Importance of path learning (95% CI for mean); authority centrality of action layer nodes that represent path learning in the *ND* experiment.

knowledge they used is ignored in subsequent actions. This occurs because the designer stops using their initial path knowledge ($d_{ij} = e$), after the true path distance is learned.

Figure 5.13 shows how the designer uses the path knowledge generated by the actions in Figure 5.12 to solve the design problem. Mean of hub centrality shows that after the first few timesteps, starting path knowledge is significantly more important to the knowledge structure development than non-starting path knowledge. In simulation *ND*, starting path knowledge guides route generation during every timestep. This makes it considerably more important to the designer's actions than non-starting path knowledge, which is only used in some routes.

In simulation *D*, a different design driver dynamic occurs. Shown in Figure 5.14, the agent's decision to commit to a starting path from City 0 shifts the design drivers. This decision is denoted in the figure by the vertical red line. Compared to simulation *ND*, starting-path knowledge decreases in importance and non-starting path knowledge increases in importance. This is shown by the statistically significant divergence between the importance of starting and non-starting knowledge segments. The results

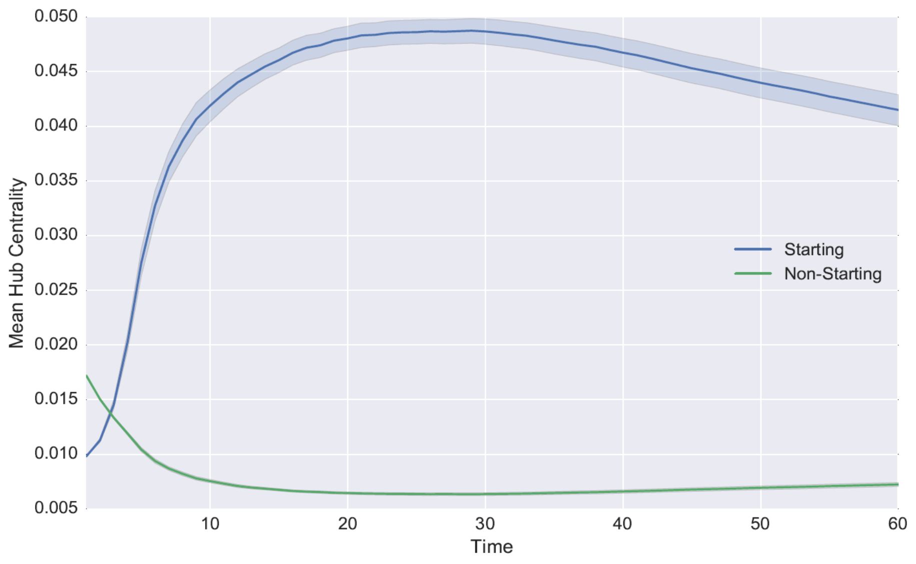


Figure 5.13: Importance of path distance knowledge (95% CI for mean); hub centrality of designer path knowledge for the *ND* experiment.

indicate that the agent is using knowledge about the undecided paths more heavily in the route creation process. It is suggested that this shift in importance accounts for the agent converging on the optimal path more than 20% more frequently in the *D* experiment than in the *ND* experiment.

In the Decision experiments, the decision shifts knowledge usage from the starting path knowledge to the non-starting path knowledge. This decreasing the importance of starting path knowledge to the knowledge structure development and increases non-starting path knowledge. The changing importance is shown in the hub centrality for the segment in Figure 5.14. Variations in when the first decision occurs causes the increased confidence interval in the Decision experiment compared to the No Decision experiment. The inflection points and subsequent divergence between the two experiments indicate a noticeable shift in what is important to the agent designer.

Both sets of design driver analyses match the expected design driver behavior. When the designer begins to develop certain elements of the knowledge structure during the design activity, it becomes a driver that can be identified in the K-A-D Framework.

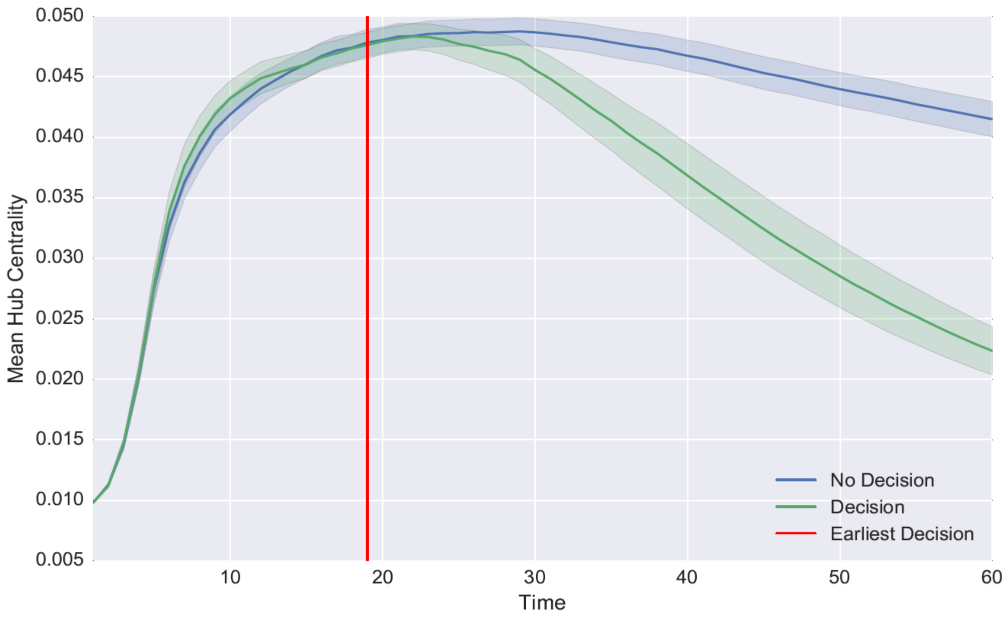


Figure 5.14: Importance of starting path knowledge (95% CI for mean); hub centrality comparison of starting path knowledge that is represented in the agent's solution for D and ND experiments.

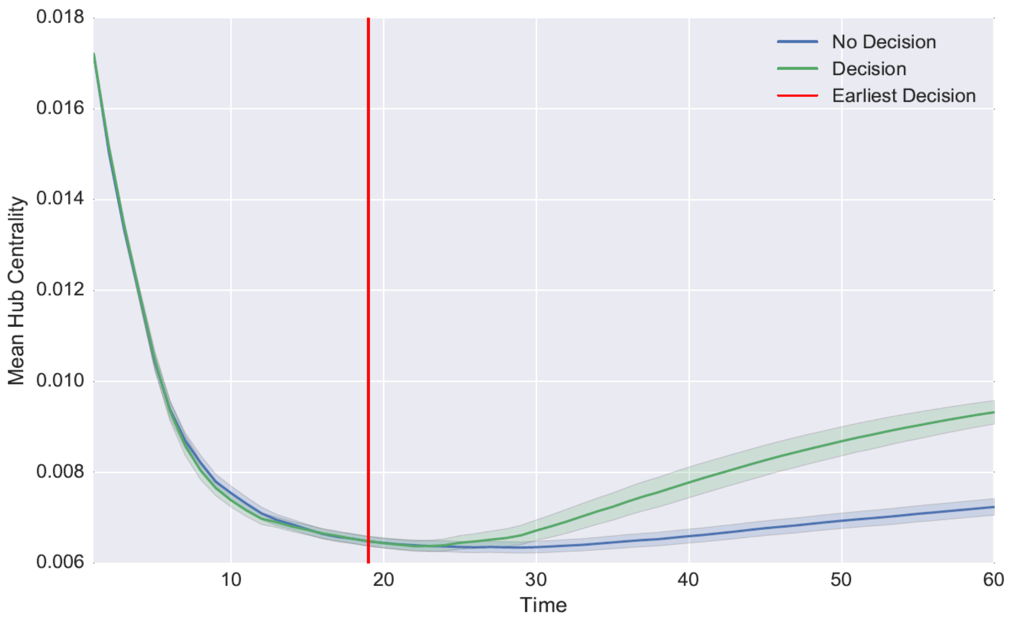


Figure 5.15: Importance of non-starting path knowledge (95% CI for mean); hub centrality comparison of non-starting path knowledge that is represented in the agent's solution for D and ND experiments.

5.4.1 Identifying Path Dependence

In the simulation, the designer converges on a solution by reusing preferred paths to help refine better routes. Discussed in Section 5.3, this process is path dependent. When the path dependencies develop, they should appear as sustained design drivers that are more important than comparable knowledge elements. Figures 5.8 and 5.9 correctly identify this behavior. However, it is not just the final outcome that is path dependent, the pheromones on each TSP path are also path dependent. Thus, if the K-A-D network is properly capturing the agent designer's path dependent behavior, centrality analysis should be able to replicate the designer's pheromone mechanisms.

Network centrality results cannot be directly compared to the pheromones for the TSP paths because the pheromone dissipation factor and route length introduce distortions in how the pheromones are updated. Instead, the agent's most likely route (MLR) can be used to make a reasonable comparison. The MLR is the route that has the highest probability of being taken given the transition rule in Equation 5.2. The MLR can be constructed from pheromones or the pheromone values can be substituted for another set of values, in this case network centrality measures. The structure of Equation 5.3 and fixed TSP starting city guarantees that the highest value path selection replicates the most likely route based on a set of path values. Starting at City 0, the path with the highest value is chosen. This leads to the next city, where the highest value path to an unvisited city is chosen. The highest value selection process continues until a full route is constructed.

MLRs are compared between the network centrality of nodes representing path knowledge and the pheromones for those paths. This comparison evaluates capability of different network centralities to measure path dependencies in the agent designer's actions and the design outcomes. To quantify the comparison, the error between the two sets of values is the percent of paths in the centrality-based MLR that are not in the pheromone-based MLR. Route error E was calculated for R, ND, and D.

Figure 5.16, shows the error for the Random experiment. In this experiment, the agent does not produce design drivers and thus the network analysis cannot replicate the agent behavior. It is expected that the MLRs are not comparable between the centrality and pheromone paths. This provides the expected error of the path prediction process if paths are created at random, $E \approx 80\%$, which provides a baseline to judge the performance of ND and D experiments.

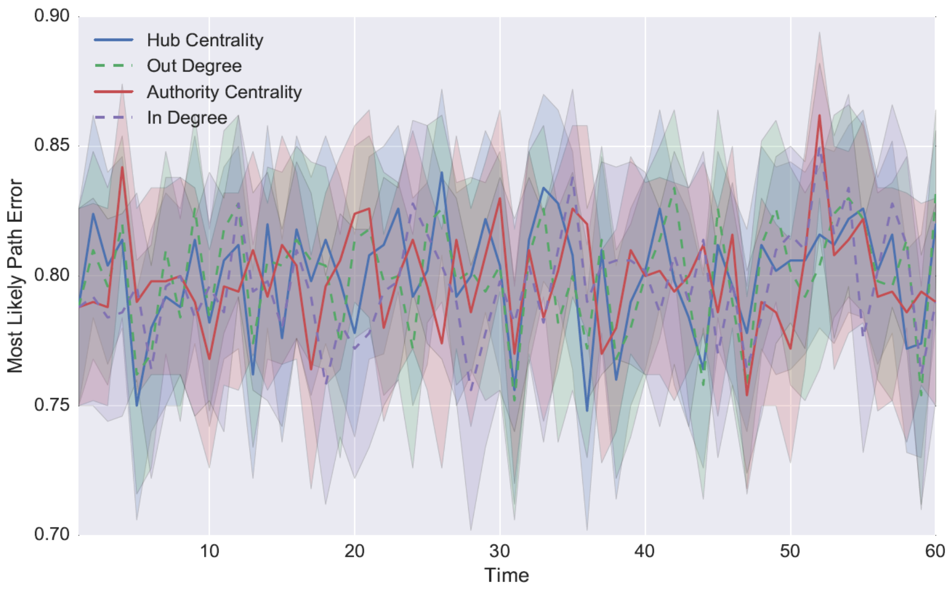


Figure 5.16: Most likely path comparison (95% CI for mean); route error between pheromone and centrality MLRs, experiment R. This identifies a baseline error rate of 79.8% for random route selection.

Figures 5.17 and 5.18 show the path error for No Decision and Decision experiments. This indicates that path dependencies in the designer's behavior can be identified and measured through out-degree and hub centrality analyses. In-degree and authority centrality do not make accurate predictors of path dependencies. This is to be expected. Out-degree and hub centrality measure importance by how a knowledge structure element is used. In-degree and authority centrality measure importance by how an element is created. Path dependencies develop from the repeated use of a knowledge structure element, thus out-degree and hub centrality are better suited for measuring their development.

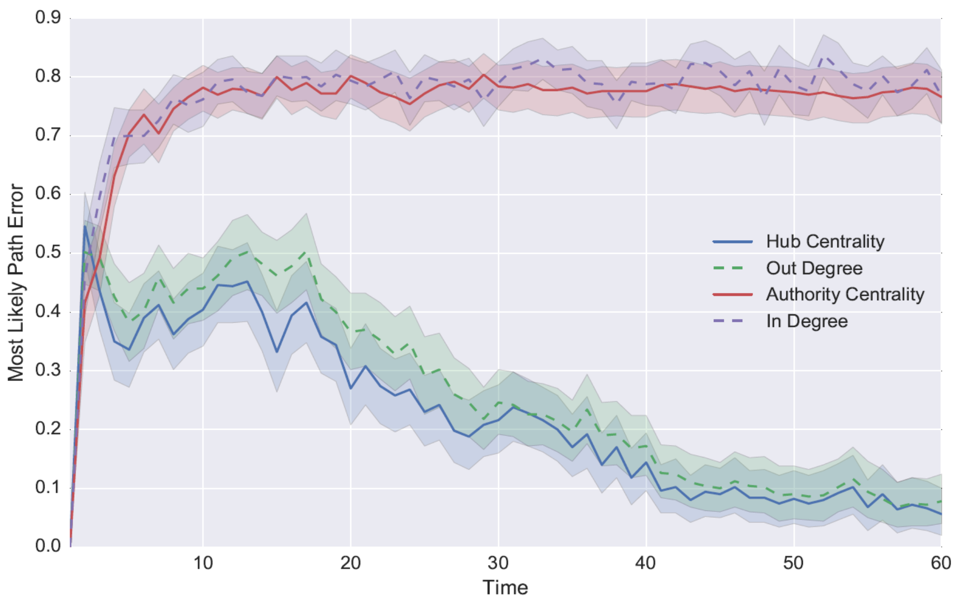


Figure 5.17: Most likely path comparison (95% CI for mean); route error between pheromone and centrality MLRs, experiment *ND*. At $t = 1$, there is no error between solutions because only one route has been created.

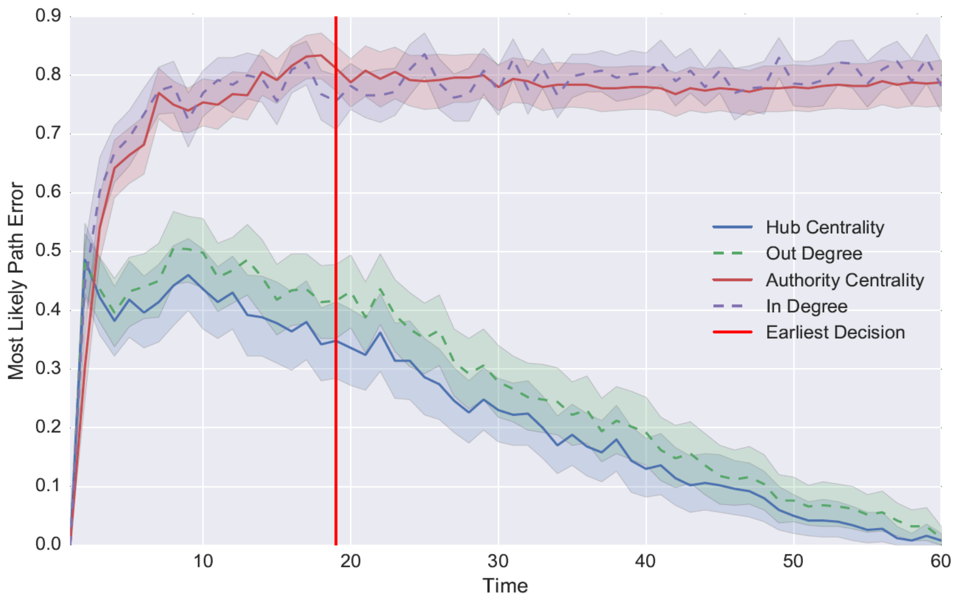


Figure 5.18: Most likely path comparison (95% CI for mean); route error between pheromone and centrality MLRs, experiment *D*. At $t = 1$, there is no error between solutions because only one route has been created.

Focusing on out-degree and hub centrality, the path dependence results are initially imperfect, only predicting approximately 50% of the correct paths in the most likely routes, which is almost 40% better than random. As path dependencies develop, the error decreases and both out-degree and hub centralities more accurately predict the strongest path. In the decision experiment, ingrained preferences are explicitly locked-in by the designer to create a fixed path in the route. This reinforces the existing path dependencies and reliably increases the accuracy of the centrality-based routes. The result is that the error rate in D decreases to $E \approx 0.0$ and does so consistently after decisions are made.

The four experiments and their knowledge-based analysis through the K-A-D Framework have demonstrated the following:

- Random Experiment: unintelligent agent randomly creates routes and does not learn or explore. Because of this, the agent is unable to develop a solution to the TSP outside of the random chance that an good solution is chosen in the final timestep.
  - When knowledge-based path dependencies do not exist, there are no design drivers in the knowledge structure development.
- Initial Route Dependence: agent immediately commits the first route to the product structure. The solution lock-in also prevents the agent from developing a good solution to the TSP because the first route is created at random.
  - The influence of path dependencies that ingrain design outcomes appear as knowledge structure elements that have sustained importance relative to other knowledge structure elements.
  - Design drivers that influence the generation of knowledge structure elements can be identified with out-degree and hub centrality analysis of the K-A-D network.

- No Decision: agent learns paths, explores possible routes, and converges on a solution, but does not commit to a product structure until the last timestep. In this experiment, the agent incremental improves its solution towards an optima by increasing its preference towards paths that are used in short routes.
  - Path dependent behaviors that converge the solution can be identified by measuring the importance of knowledge structure elements to the knowledge structure development.
  - Determining the emergence of design drivers with K-A-D network analysis indicates when path dependencies may ingrain design outcomes.
  - Sustained design drivers suggest that the design outcome is ingrained by the knowledge structure development.
  - The dynamic influence of path dependent behaviors on design outcomes can be measured with out-degree and hub centrality analysis of the K-A-D network.
- Decision: agent learns paths, explores possible routes, converges on a solution, and makes decisions committing to the product structure. In this experiment, the agent improves its solution towards an optima by creating preference towards promising paths. Heavily preferred paths are locked-in during the design activity and shift importance to undecided paths.
  - Decisions shift knowledge structure development by changing what knowledge structure elements are important to the design activity.
  - Knowledge structure development shifts caused by decisions can be identified with the K-A-D network.

5.5 Conclusion

This chapter demonstrated the K-A-D Framework on a design activity simulation where an agent designs a solution to the TSP. The simulations generated data to test the framework's ability to describe a design activity in such a way that design drivers could be measured to identify the influence of path dependencies. The results presented here demonstrate the capabilities of the K-A-D Framework in a repetitive and well structured environment. While this does not necessarily represent all of the complexities of a real-life scenario, the simulation does model some of the unique aspects of design that create emergent design outcomes.

During the simulation, the agent designer learns, exploits knowledge to create solutions, and uses knowledge to guide future learning, problem-solving, and decision-making. This captures fundamental design behaviors. Using the K-A-D Framework to describe these behaviors in terms of the knowledge structure development was able to create network modeling of the design activity. Design drivers and path dependencies were measure by analyzing the corresponding networks. These results are an example of how the K-A-D Framework can elicit how knowledge structure development influences design outcomes.

From a naval design perspective, the K-A-D Framework approach is useful in two ways. First, it introduces the concept that knowledge structures drive design, not the product being designed. This alone is a critical step to preventing design failures caused by knowledge-based emergence. Second, the K-A-D Framework provides a formal framework for describing a knowledge structure, measuring its development, and identifying emergent influences on design outcomes. Application of the K-A-D Framework to a naval design activity may enable the identification of design failures and subsequent prevention measures. Demonstrating how this goal can be achieved is the subject of the following chapters.

CHAPTER VI

Distributed Systems Design Failure Analysis Using the K-A-D Framework

Shipboard systems, associated engineering, and design work account for almost half of the cost of a naval vessel (*Miroyannis*, 2006). Over the last 50 years, shipboard systems have accounted for nearly 20% of the annual cost increase for procuring a naval vessel (*Arena et al.*, 2006). Within a naval procurement program, the design and engineering of shipboard systems is volatile. Small decisions can propagate into significant design, engineering, and construction work (*Schank et al.*, 2009), which often drives cost overruns and delays (First Marine International, 2005). These issues are emergent design failures that start in early-stage design, but are realized in later-stages.

During early-stage design, distributed systems are considered at low-fidelities to measure concept feasibility and requirement satisfaction (*Andrews*, 2016). This allows designers to make sizing and layout decisions with confidence that an acceptable distributed system can be designed for the vessel. Detailed system configurations are designed later, using better defined layout, components, and operational specifications. This means that early-stage decisions about component layout and system routings have considerable influence on the late-stage distributed system design activity.

Late-stage system design integrates separately engineered systems so that they operate interdependently, fit together in the same physical space, and work correctly in their locations. This design activity creates a knowledge structure supporting how each system and its routings integrate with other components, systems, and spaces within the vessel. Design failure occurs when developing this knowledge structure is too difficult to complete in the allotted timeframe. Because early-stage design determines many of the systems, interdependencies, and physical interactions, early-stage decisions may increase the potential for emergent late-stage design failure. However, current early-stage distributed system analysis only considers the attributes of the product structure. Without considering the knowledge structure implications of early-stage decisions, designers cannot address the risk of emergent design failures.

In this chapter, the K-A-D Framework is applied to the distributed system design activity to introduce potential late-stage knowledge structure development into early-stage analysis. To facilitate this, Section 6.1 introduces a method for generating ensembles of many distributed system design representations for early-stage design analysis. Section 6.2 proposes mathematical definitions for distributed system design knowledge and actions that convert the design representations into approximate knowledge structures. Section 6.3 discusses knowledge structure analysis for investigating emergent design failures. Section 6.4 demonstrates the importance of knowledge structure analysis for considering design failures by comparing knowledge and product structure analysis methods.

In Chapter VII, these definitions and methods are implemented to simulate knowledge structure growth during system design for a multi-system vessel concept. Analysis of the generated knowledge structures identifies decisions with a high risk for design failures, design drivers behind the failures, and opportunities to leverage path dependencies to prevent potential emergent failures.

6.1 Generating Distributed Systems in Early-Stage Design

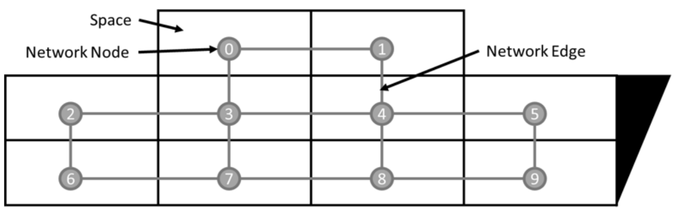
To create knowledge structures that accurately represent distributed system design three types of information are needed: information about knowledge that will be used, the actions that will use them, and the knowledge that will be created. These elements define the pattern of knowledge structure development that may create design failures. However, in early-stage design, information about these elements is limited by the level of available detail. Rough ideas of components and systems are often known, but the way knowledge will be used by designers is not. This makes it difficult to anticipate how a knowledge structure will develop during system design. Because knowledge structures determine the design outcomes, the ability to know how knowledge structures unfold may be more important than knowing about potential product structures.

The problem is two-fold. First, knowledge that is used in the design is dependent on the interactions of systems within the vessel. Therefore, the layout of a vessel concept may have a significant impact on the design knowledge structure. The distributed system configuration details that define these interactions are not available until late-stage design, when it is too late to avoid design failure. To develop the appropriate knowledge structures in early-stage design, so that they can be used to decrease the chance of early-stage design failure, configuration and integration information need to be available earlier. The second problem is that there is a combinatorial explosion of possible configurations as the level of design detail increases. Thus, a single distributed system configuration is unlikely to represent the knowledge structure that will develop. To remedy this, many possible configurations need to be investigated and a method to evaluate multiple configurations efficiently needs to be established.

Shields et al. (2017) presented a method for generating early-stage distributed system configurations, called physical system solutions, that solves the aforementioned problems. The method represents both the vessel layout, (physical architecture) and the systems' functional interdependencies (logical architecture) as networks. The Logical-Physical Architecture Translation L-PAT algorithm, was developed to quickly convert these networks into physical system solutions through a stochastic routing method. This approach creates ensembles, many randomly drawn instances from the same conditions, of physical system solutions. Ensemble of configurations can be analyzed to statistically quantify the expected configuration properties. This enables the creation and analysis of potential future knowledge structures for a distributed system design activity.

In this chapter, the ensemble approach is extended to knowledge structure analysis to investigate the expected properties of the future design activity. Ensembles of simulated physical system solutions provides the system interaction information needed to approximate knowledge structure creation. This enables characteristics of late-stage knowledge structure development to be considered in early-stage design. The complete methodology for generating physical system solutions and a demonstration can be found in *Shields et al.* (2017). The outline of the method follows (configuration will be used in place of physical system solution).

Distributed system configurations can be thought of as a set of paths for resource distribution between components within a vessel. A feasible configuration must fit within the vessel geometry and enable all components to function. The physical architecture of a vessel defines the vessel geometry and spatial definition. In this thesis, the physical architecture is represented as the component layout and arrangement of spaces. This constrains where the distributed system can pass through and where it needs to connect to components. The nature of these connections is described in the logical architecture. Here, the logical architecture is considered to be the functional connections between components within the vessel (e.g. a computer needs electrical power from a generator). This description reflects the findings of an ongoing research program studying the design of naval distributed systems. Research collaborators



$$V = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 & 0 \\ 0 & 1 & 0 & 1 & 0 & 1 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 1 & 0 \end{bmatrix}$$

Figure 6.1: Physical architecture composed of spaces within the vessel and its network representation. Nodes in V represent spaces. Edges between nodes represent adjacencies between spaces.

include students, researchers, and faculty at The University of Michigan, Virginia Polytechnic Institute and State University, The University College London, and Delft University of Technology.

Both the physical and logical architectures can be represented as networks. V represents the physical architecture as a single-layer network that describes the relationships between vessel spaces, see Figure 6.1. S represents the logical architecture as multi-layer network that describes the relationships between components within different system types, see Figure 6.2. Components are represented as nodes $S_{(\cdot,i)}$ that exist in each system type layer $l = 1, 2, \ldots$. Functional connections between components for a given type of system are represented as $S_{(l,ij)}$. The following figures demonstrate the physical and logic architectures with their network representations.

The distributed system configuration G is the set of paths through V that satisfy every relationship between components in S. In this implementation, each path is generated by a randomly selected shortest path routing. The selection for a single component connectivity is shown in Figure 6.3. The L-PAT algorithm executes this selection for every edge in each layer S_l. Using the K-A-D Framework, the distributed system configuration output, G, can be converted into a corresponding knowledge structure approximation.

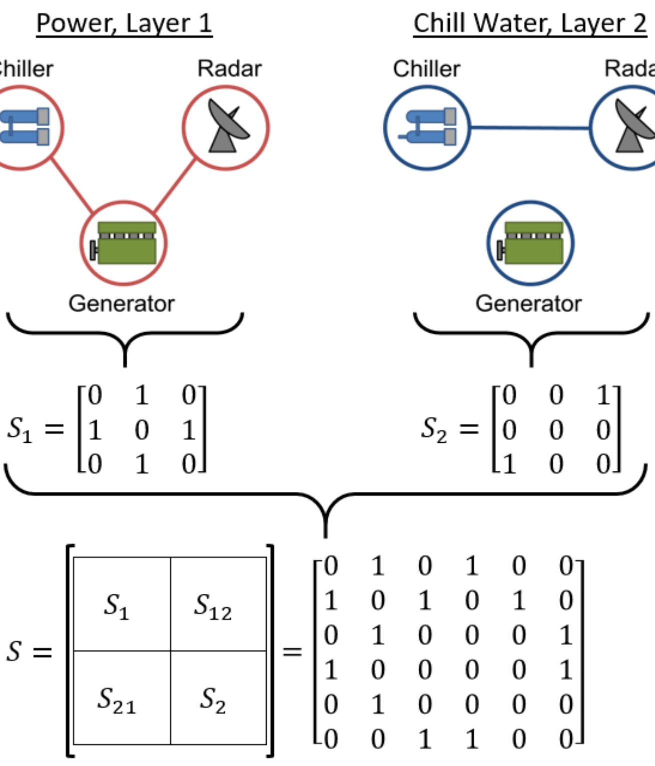

Power, Layer 1
Chiller
Radar
Generator
Chill Water, Layer 2
Chiller
Radar
Generator
$S_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$
$S_2 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 1 & 0 & 0 \end{bmatrix}$
$S = \begin{bmatrix} S_1 & S_{12} \\ S_{21} & S_2 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 & 1 & 0 & 0 \\ 1 & 0 & 1 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 & 0 & 1 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 \end{bmatrix}$

Figure 6.2: Logical architecture of two systems with three components and its network representation. Nodes in S_l represent components in system l. Edges within S_l represent connections between components in system l. Edges between system layers represent components that are shared by multiple systems (adapted from (*Shields et al.*, 2017)).

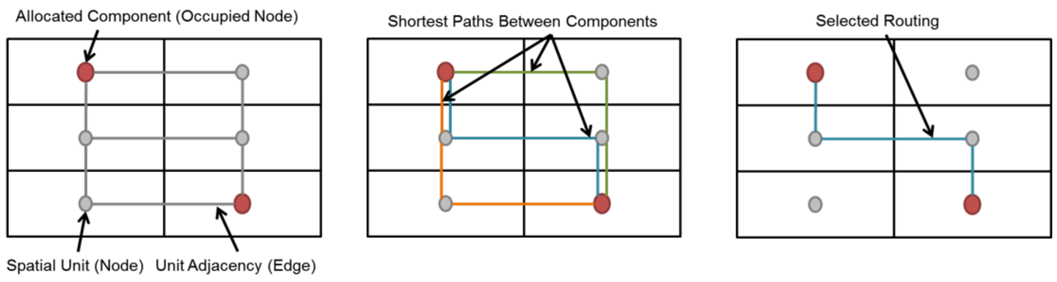

Allocated Component (Occupied Node)
Spatial Unit (Node)
Unit Adjacency (Edge)
Shortest Paths Between Components
Selected Routing

Figure 6.3: From left to right: Structure of V and its components, enumeration of possible shortest paths in V between components, randomly selected path (*Shields et al.*, 2017).

6.2 Distributed System Design Knowledge Structures

The K-A-D Framework converts a distributed system configuration into a knowledge structure by defining knowledge, actions, and their relationships in a system design activity. Once these elements are defined, knowledge structure ensembles are created from the distributed system configurations generated by the L-PAT algorithm. This enables knowledge-based analysis of design outcomes and their emergence from knowledge structure development.

The knowledge structure model does not incorporate decisions in the K-A-D Framework. In a real design activity, decision could be included by tracking how parts of the distributed system configuration are committed to product structure. This would limit the set of potential product structures as decisions were made. When a decision is made about configuration elements, the decided elements are removed from the random generation of ensembles. This models that the element is always included in the product structure. The differential between knowledge structure characteristics between ensembles with and without decisions would provide insight into how the decision influenced knowledge structure development.

The general knowledge structure model for distributed system design is rooted in the integration of multiple systems and components. To understand an integrated configuration, designers must evaluate how the distribution system routings and system components operate within vessel environment, as well as in the presence of other distribution systems. This requires that the designer develops knowledge that integrates existing information about routings, components, vessel spaces, and the connections between spaces. As the number of interactions between these elements increases, so does the number of knowledge interdependencies that the designer is expected to integrate.

The K-A-D Framework is used to model the knowledge integration actions. First, existing knowledge is defined to represent routings, components, and environments.

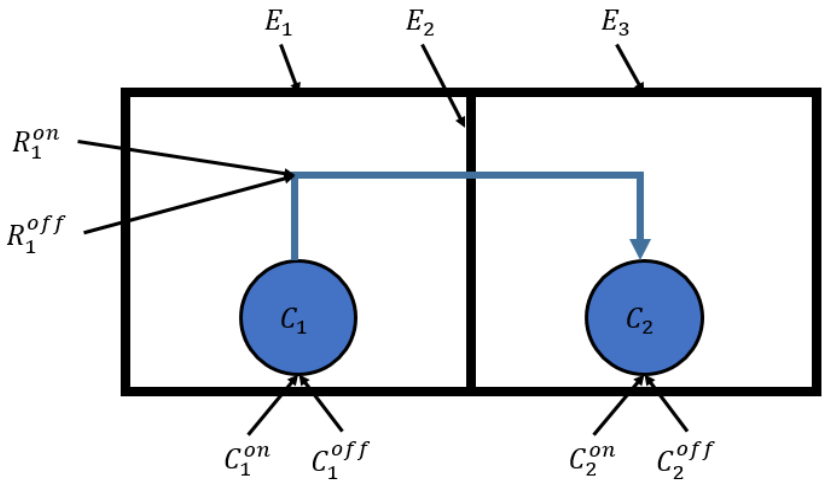


Figure 6.4: Distributed system configuration with two spaces (black boxes), two components (blue circles), and a routing from component C_1 to component C_2 (blue line). Knowledge elements for system environments E_i, system components C_i^{on} and C_i^{off}, and system routings R_i^{on} and R_i^{off} are labeled.

Environments are defined as the vessel spaces and the connections between them. Individual elements in the knowledge structure represents knowledge about the properties and functions of each routing, component and environment instance. For example, two different components would have their own knowledge structure elements. Elements for routings or components are broken into knowledge representing 'on' and 'off' states. Knowledge elements for 'on' represent functional properties and 'off' represents physical properties. Existing elements are defined as precedent knowledge in the form, $k = G_0(\emptyset, \emptyset)$, where k is routing knowledge R^{on} or R^{off}, component knowledge C^{on} or C^{off}, and environment knowledge E. Figure 6.4 demonstrates how knowledge structure elements are defined for a small distributed system configuration representation.

The knowledge structure approximation assumes that designers generate knowledge about the integrated configuration through actions that use interdependent knowledge elements. Interdependencies are defined by the configuration. For example, if a routing R_1 passes through environment E_1, then the knowledge associated with both elements are interdependent. The complete knowledge structure for the in-

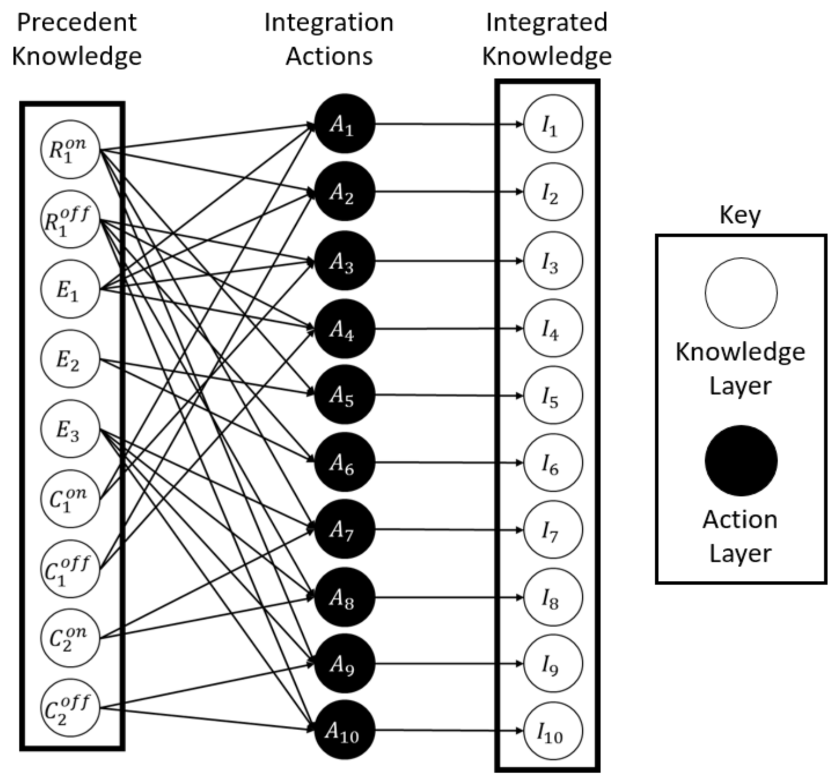


Figure 6.5: Knowledge structure approximation for a distributed system configuration in Figure 6.4. This is created by the K-A-D Framework and its network representation to model the integration of knowledge elements required to understand the configuration. Precedent knowledge is integrated though actions A_i that create integrated knowledge I_i.

tegrated configuration needs to contain elements that consider the interdependencies. In this case, $I_1 = A_1(R_1^{on}, E_1)$ and $I_2 = A_2(R_1^{off}, E_1)$, where I_i represents knowledge for a part of the integrated configuration that is generated by designer action A_i. These actions can be combined with precedent knowledge to create a K-A-D network that approximates the design activity based on distributed system configuration. Figure 6.5 demonstrates the knowledge structure approximation for the configuration in Figure 6.4.

The proposed K-A-D Framework description characterizes the potential knowledge structures in a way that is not predicated on a particular system design process. This is possible because the knowledge structure approximation captures a fundamental relationship between knowledge and system design, called the *law of functional complexity* that defines how difficult a system is to design. It can be shown that the K-A-D Framework captures this relationship by comparing the mathematical definition of the law of functional complexity to the knowledge structure model.

Bar-Yam (2003a) describes laws in complex systems as fundamental relationships between qualities of system, action, environment, function, and information. The law of functional complexity describes the amount of information needed to completely specify the function of a system (*Bar-Yam*, 1997). Functional complexity is defined as:

$$C(f) = C(a) \cdot 2^{C(e)} \tag{6.1}$$

where for a given system whose function is to be specified, the environmental (input) variables have complexity $C(e)$, and the actions of the system have a complexity of $C(a)$. When designing a system, $C(f)$ this is the amount of information needed to fully understand the system's behavior.

Functional complexity provides an intuitive check on the knowledge structure defined in the K-A-D Framework. The estimated knowledge structure captures knowledge required to understand each system environment within the vessel. Evaluated across the entire vessel, the knowledge structure estimates the amount of knowledge generated to understand the system configuration.

The K-A-D Framework for distributed system design articulates the functional complexity relationship. Applied to each system environment, $C(a)$ is the complexity of component actions and $C(e)$ is the complexity of the system routings. Specifically, $C(a)$ is the number of actions that could occur (e.g. 'off' if no component is present, 'on' or 'off' for a single component) and $C(e)$ is the number of states the environment could be in (e.g. 'off' if no routings are present, 'on' or 'off' for a single routing, 'on-on', 'on-off', 'off-on', and 'off-off' for two routings).

The network-based definition of distributed system components and routings that define the L-PAT's physical system solutions can be readily mapped to the K-A-D Framework. Each node and edge in the physical architecture representation V_{ij} is a system environment node E_{ij} in the K-A-D Framework. Each component represented by a node in the logical architecture, $S_{\cdot,i}$, is represented as its corresponding on and off component knowledge nodes, C_i^{on} and C_i^{off}. The routings between components i and j in system l, $S_{l,ij}$, are represented as on and off routing knowledge nodes, $R_{l,ij}^{on}$ and $R_{l,ij}^{off}$.

The configuration of the physical system solution defines the knowledge interdependencies created in the K-A-D Framework. For each system environment, knowledge representing the components allocated to it and the system routings that use it are integrated in the knowledge structure. The complete knowledge structure can be created by iterating through each node and edge in the physical architecture and creating the knowledge interdependencies for that element, Figure 6.6 describes this process.

6.3 Distributed System Knowledge Structure Analysis

Chapter V showed how network-based knowledge structure analysis can measure design drivers and identify path dependencies that influence design outcomes. The same concepts can be used to measure emergent design drivers for the knowledge structure developed during the design of a distributed system configuration. Applied over an ensemble of knowledge structures, this can measure expected design drivers, path dependencies, and the potential for design failure.

The distributed system design knowledge structures can approximate the knowledge integration effort for a given configuration. When the integration effort is high, the number of actions the designer must take is also high. Thus, the first measure of interest is the number of actions needed to generate a configuration's knowledge

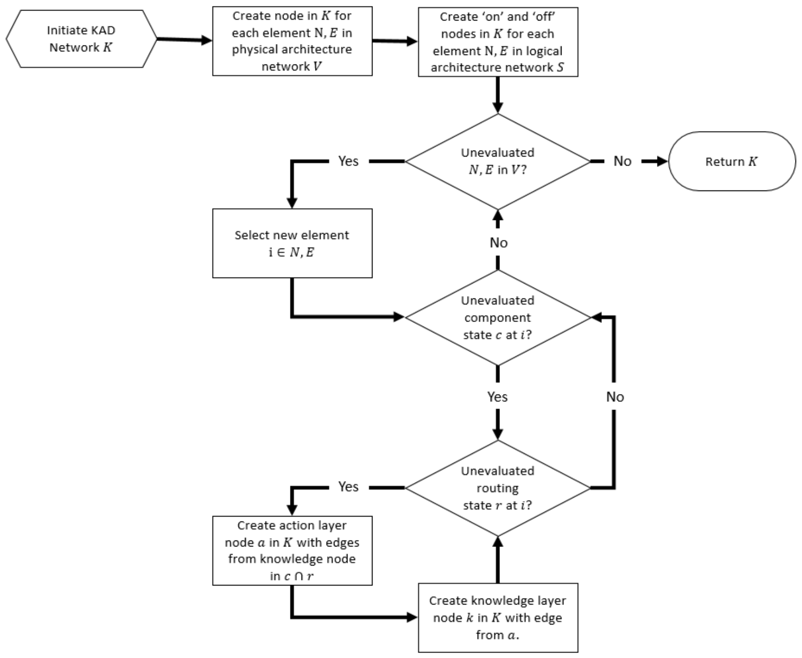

Initiate KAD Network K
Create node in K for each element N, E in physical architecture network V
Create 'on' and 'off' nodes in K for each element N, E in logical architecture network S
Unevaluated N, E in V?
Yes
No
Return K
Select new element $i \in N, E$
No
Unevaluated component state c at i?
Yes
No
Unevaluated routing state r at i?
Yes
Create action layer node a in K with edges from knowledge node in $c \cap r$
Create knowledge layer node k in K with edge from a.

Figure 6.6: Process flowchart for converting an L-PAT distributed system representations into a knowledge structure approximation using the K-A-D Framework.

structure. In the K-A-D network, this is the number of nodes in the action layer N_A. The second measure of interest is how knowledge elements contribute to designer actions. This is represented by the out-degree of nodes in the knowledge layer $\boldsymbol{k}^{out} = [k_0^{out}, k_1^{out}, \ldots, k_{N_K}^{out}]$, where N_K is the number of nodes in the knowledge layer.

The number of designer actions estimates the effort required to design a distributed system configuration. Actions integrate existing knowledge to account for all of the knowledge interdependencies created by the configuration. As the number of interdependencies increases, the number of actions will also increase, making it an estimator of design effort. Out-degree of knowledge elements quantifies how knowledge was used to complete the design activity. This measures the importance of individual elements to knowledge structure development.

Expected knowledge structure characteristics are calculated by evaluating the number of action and its distribution over an ensemble. These characteristics provide insight into how the conditions that define an ensemble influence the subsequent design activity. Comparing characteristics between different ensembles can reveal how one set of early-stage design conditions may be more likely to create late-stage design failures than another.

Unfortunately, the simple approach of measuring characteristics by generating and analyzing K-A-D networks is not always possible. The size of the K-A-D network grows exponentially with the number of routing interactions and can easily become larger than is reasonable to store and evaluate. When this occurs, a more tractable method is to measure characteristics by analyzing the distributed system design representation, without generating the K-A-D network. The law of functional complexity facilitates this analysis. The number of required actions can be calculated by iterating through the node and edge elements of the physical architecture network representation. The network-based calculation of functional complexity for distributed systems is described in *Shields et al.* (2016b). At each element V_{ij} in the vessel network, $C_{ij}(a)$ is the complexity of action for the corresponding system environment knowledge E_{ij} and is defined as

$$C_{ij}(a) = 2^{V_{ij}}, \tag{6.2}$$

where V_{ij} is the number of components allocated to that element by the L-PAT algorithm.

Complexity of the environment $C_{ij}(e)$ is the number of routing interactions that occur on the element V_{ij}. Using the routing paths G that are output by the L-PAT algorithm, $C(e)$ is a count of the number of times V_{ij} appears in G. This can be calculated as

$$C_{ij}(e) = \sum_{l} \sum_{s \neq t} \delta_{stl}(i, j), \tag{6.3}$$

where $\delta_{stl}(i,j) = 1$ if the route between components s and t in system l contains V_{ij} and $\delta_{stl}(i,j) = 0$ otherwise. Combining Equation 6.2 and Equation 6.3 into Equation 6.1 and rearranging the terms gives the functional complexity $C_{ij}(f)$ of element V_{ij}

$$C_{ij}(f) = 2^{V_{ij} + \sum_l \sum_{s \neq t} \delta_{stl}(i,j)}. \tag{6.4}$$

$C_{ij}(f)$ counts the estimated number of designer actions required to integrate the distributed system configuration at the system environment represented by V_{ij} in the vessel network. Equation 6.4 can be applied to every element in the vessel network to calculate the total number of designer actions required to integrate a prospective distributed system design. The resulting quantity $C_V(f)$ is defined as

$$C_V(f) = \sum_{i,j} 2^{V_{ij} + \sum_l \sum_{s \neq t} \delta_{stl}(i,j)}. \tag{6.5}$$

Complexity values in Equation 6.4 and Equation 6.5 are equivalent to K-A-D network characteristics. $C_{ij}(f)$ is equal to the out-degree k_{ij}^{out} of knowledge node E_{ij} in the K-A-D network. Thus, number of actions using each system environment element is calculated by applying Equation 6.4 to each physical architecture element V_{ij}. The resulting degree sequence $\boldsymbol{k}_E^{out}$ does not include component or routing knowledge elements, but it can be calculated quickly and describes a critical part of knowledge structure development. $C_V(f)$ is equal to N_A, the number of action layer nodes.

Both $\boldsymbol{k}_E^{out}$ and N_A are emergent properties of the system design activity. They arise from the knowledge interdependencies that need to be developed to understand a system design concept. The L-PAT algorithm provides the necessary information to predict how the knowledge structure properties emerge. In the following chapter, N_A will be used to estimate the effort required to design a distributed system and $\boldsymbol{k}_E^{out}$ will be used to identify design drivers. Together, they describe how early-stage decisions influence design outcomes and the potential for failure.

It needs to be reiterated that the described knowledge structure characteristics are not predicated on how the design activity is executed. The characteristics are calculated from a completed knowledge structure based on the precedent knowledge and configuration. This does not model the order in which the knowledge structure is developed. While additional details can be added to consider the execution of development for a specific tool or process, the current form is more appropriate for a general model.

6.4 Knowledge Structures Versus Product Structures

The knowledge structure created from the distributed system configuration introduces knowledge generation requirements and knowledge interdependencies into early-stage design analysis. Comparing the knowledge-based analysis to traditional product-based analysis illustrates the value of this perspective. The product perspective only considers the elements within the product and their relationships, not the ideas and information that they represent. In isolation, a product structure perspective masks the knowledge structure that support it.

Figure 6.7 demonstrates this issue with a product and knowledge structure comparison for a distributed system configuration. Comparing the knowledge and product elements in Figure 6.7 shows the difference in the scope of each perspective. The knowledge-based perspective describes the knowledge that is represented in the configuration. The product-based perspective describes the elements that compose the product. Emergent design failures arise from the interdependencies between design knowledge and thus cannot be captured in the product structure. For example, the Boeing 787 design suffered significant delays due to unexpected knowledge structure interdependencies, not because the 787 had different product structure interdependencies than other Boeing airplanes (*Kotha and Srikanth*, 2013). In Figure 6.7, the product structure has no interdependencies at the cut, only S_1 exists. The knowledge

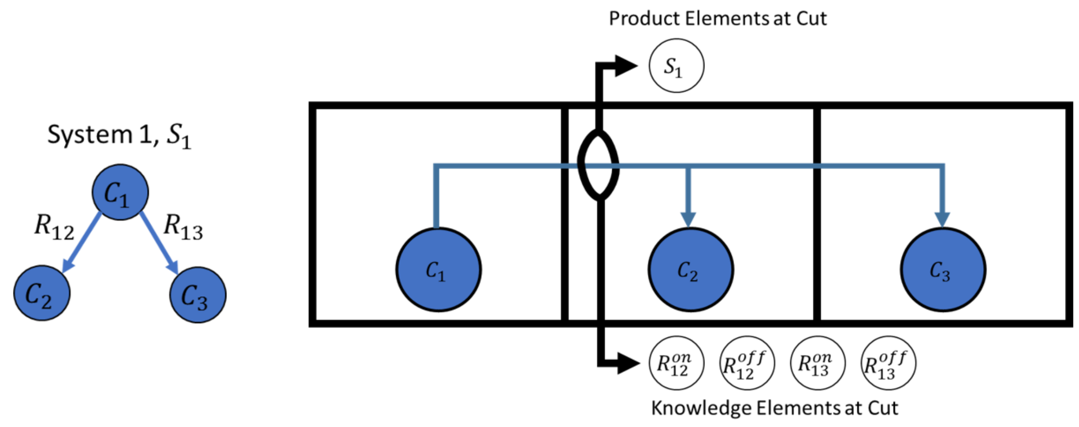


Figure 6.7: Knowledge and product structure comparison for distributed system design at the circled section. Knowledge structure describes the information that required to understand the distributed system configuration. At the cut, this is knowledge about each routing R_{12} between components C_1 and C_2 and R_{13} between components C_1 and C_3. Product structure describes the system elements at the highlighted section. This only includes that System 1 exists that location, which is denoted by S_1.

structure has four interdependencies: $R_{12}^{on} - R_{13}^{on}$, $R_{12}^{off} - R_{13}^{on}$, $R_{12}^{on} - R_{13}^{off}$, $R_{12}^{off} - R_{13}^{off}$. The inability to account for these knowledge interdependencies inhibits the prevention of emergent design failures.

Failure in distributed system design can be driven by a rapidly increasing amount of knowledge required to understand a particular design. This increase is caused by the explosion in the number of knowledge interdependencies that need to be understood. In a more intricate configuration than shown in Figure 6.7, the product interdependencies are essentially static and only vary with the number of system types. In comparison, knowledge interdependencies are highly sensitive to the configuration decisions. While this intuitively follows the modeling choices, the difference shows why a knowledge-based perspective is required to understand interdependencies that cause distributed system design failures.

This chapter leveraged the K-A-D Framework and network to approximate knowledge structures for a distributed system design activity. The following chapter will demonstrate how ensemble knowledge structure analysis can identify the potential for distributed system design failures and opportunities to prevent them. More broadly, these methods introduce the ability to consider potential design outcomes, before design activity takes place.

CHAPTER VII

Identifying and Preventing Emergent Design Failures in Early-Stage Distributed System Design

In this chapter, the K-A-D Framework and K-A-D network model developed for analyzing distributed system design activities are applied to a naval vessel concept. The results demonstrate that knowledge structure analysis can be used to predict naval design failures and identify prevention measures. Using ensemble analysis of knowledge structure development, potential design failures are identified. Exploring the design drivers behind those failures reveals how failures emerge and what can be done to prevent them. The following study is conducted on a coarse-grained naval vessel layout with a comprehensive distributed system. This model is also used in *Shields et al.* (2016b). The level of detail is appropriate for early-stage design analysis and can reveal potential late-stage design failures.

7.1 Demonstration Case

In this section, the physical and logical architectures of a naval vessel concept are represented as networks. These networks and the Logical-Physical Architecture Translation algorithm will be used for the knowledge structure analysis described in Chapter VI. The vessel concept is shown in Figure 7.1 and its physical and logical

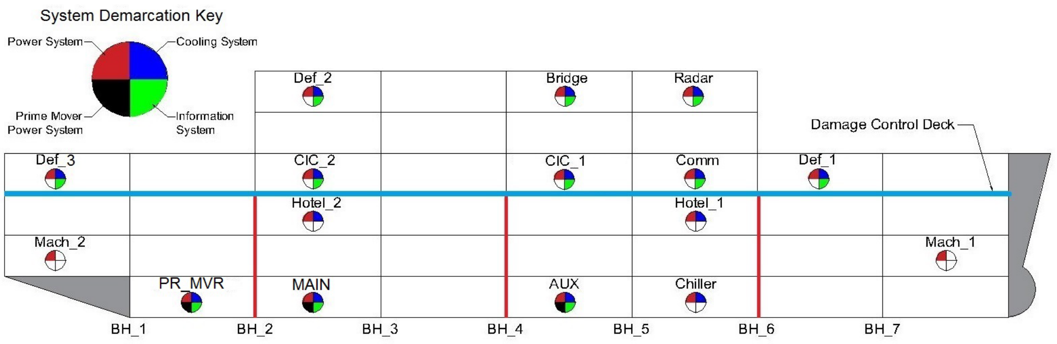


Figure 7.1: Vessel concept modeled with structural zones, component locations, watertight bulkheads (red vertical lines), and a damage control deck (blue horizontal line). Color coding on each component indicates the distributed systems the components connect with. Each component is assumed to be accessible by shipboard personnel (*Shields et al.*, 2017).

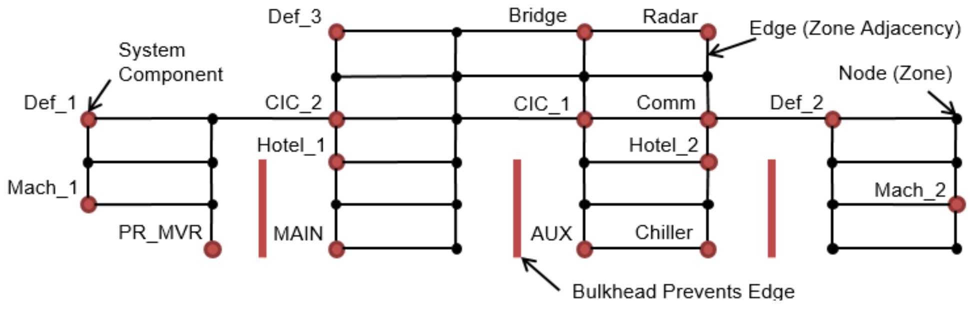


Figure 7.2: Physical architecture network representation V for the vessel in Figure 7.1. Structural zones separated by watertight bulkheads do not have edges between them (*Shields et al.*, 2017).

architectures are shown in Figures 7.2 and 7.3.

The demonstration of the K-A-D Framework methods will consider how potential future design outcomes for this vessel's distributed system design activity are influenced by changes to the physical architecture. In the vessel, three watertight bulkheads constrain the distributed system configuration. This model assumes that distributed system routings cannot pass through these bulkheads and must go around them. In the demonstration, knowledge structure analysis will be applied to a range of scenarios that relax this routing constraint to allows penetrations of watertight bulkheads.

Allowing systems to route through watertight bulkheads is not intended to capture

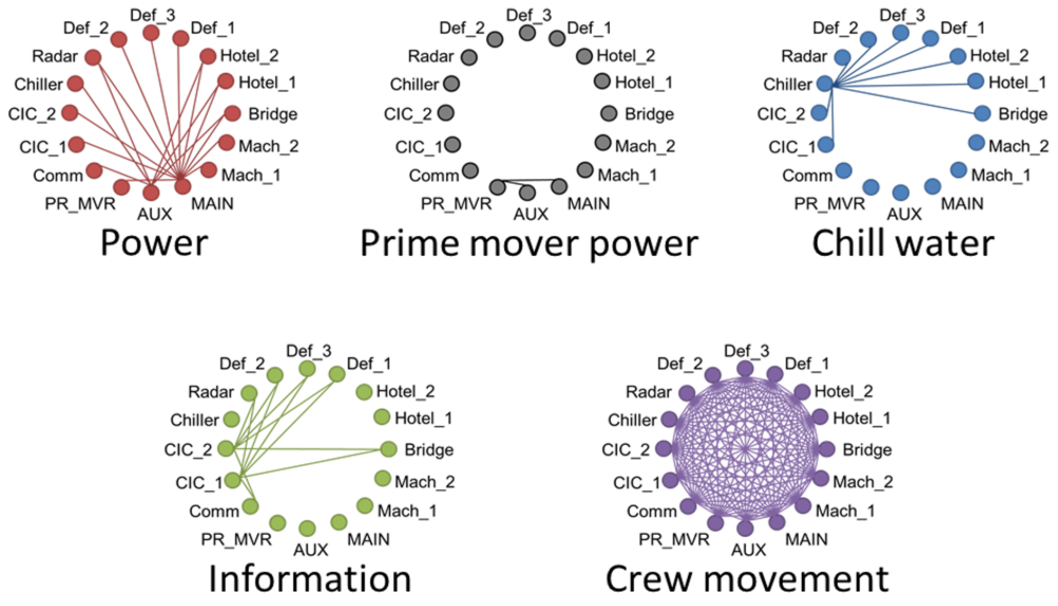


Figure 7.3: Physical architecture network representation S for the vessel in 7.1. Distributed system connectivity is defined by resource flow between components, e.g. components that need chill water must be connected to the chiller (*Shields et al.*, 2017).

a specific naval architecture rationale. Instead, it represents a level of constraint on the design activity. If the activity is highly constrained, the bulkheads are never penetrated, there is a limited number of routing options. If the activity is unconstrained, the bulkheads have penetrations, the number of routing options increases. Defining a watertight bulkhead permeability parameter p allows the level of design constraint to be controlled. Figure 7.4 shows how this is modeled in the physical architecture.

When bulkhead permeability is low, systems are unlikely to be able to pass through watertight bulkheads. This constrains routings to pass horizontally over bulkheads and makes it difficult to achieve the shortest path routings between components. When permeability is high, systems are likely to be able to pass through bulkheads. This allows routes to gravitate towards shortest path routings. In effect, permeability models constraint and objective pressures. Measuring the relationships between permeability and the number of designer actions needed to understand the integrated system can help characterize how an early-stage decision about bulkheads can poten-

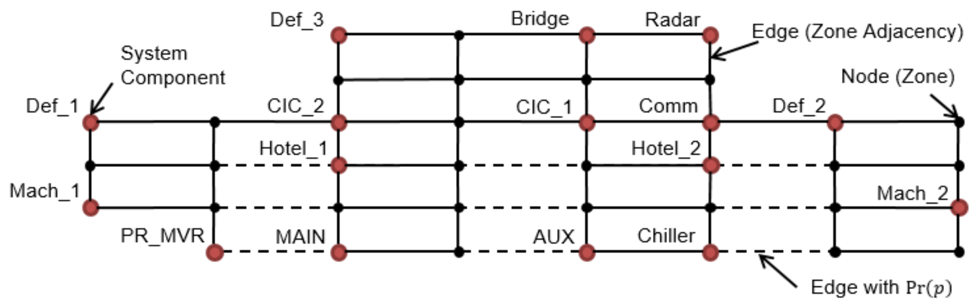


Figure 7.4: Adjusting the physical architecture network for bulkhead permeability p. For each instantiation of the physical architecture at given permeability, the penetrating edges are included in the network V with probability p.

tially cause design failures.

Bulkhead permeability defines the knowledge structure ensembles that will be analyzed. All knowledge structure instances generated for a given value of p arise from the same set of design activity conditions. Aggregating knowledge structure analyses by p provides insight into what design outcomes can be expected from those conditions. Each instance of knowledge structure analysis is generated by first permuting the physical architecture based on p (see Figure 7.5), then the L-PAT algorithm is used to generate the distributed system configuration, which is analyzed using the K-A-D network measures described in Section 6.3.

The following K-A-D Framework analyses investigate the knowledge structure characteristics of eleven ensembles and the differences between them. Ensembles of 200 instances were created for permeability parameters $p = [0.0, 0.1, \ldots, 1.0]$. Knowledge structure analyses described in Chapter VI are applied to each instance to calculate N_A, the number of designer actions needed to understand the distributed system configuration, and $\boldsymbol{k}_E$, the number of actions that use each system environment knowledge element. The results are aggregated within each ensemble and investigated with statistical methods. This enables probability of a potential design failure to be calculated and possible prevention measures to be identified.

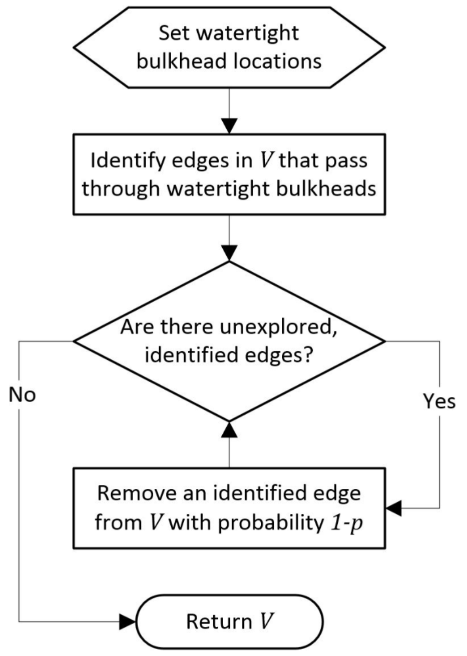

Set watertight bulkhead locations
Identify edges in V that pass through watertight bulkheads
Are there unexplored, identified edges?
No
Yes
Remove an identified edge from V with probability $1-p$
Return V

Figure 7.5: Flowchart for adjusting the physical architecture V to account for bulkhead permeability parameter (*Shields et al.*, 2016b).

7.2 Emergent Design Failures

Design failures are caused by increasing design effort. Emergent design failures arise when history of designer actions and decisions leads to an unexpected increase in design effort. Chapter VI proposes that the design effort for a distributed system design activity can be estimated using the K-A-D Framework. Interactions between components, routes, compartments, and connections between compartments create knowledge interdependencies. These interdependencies are understood through designer actions that integrate the interdependent knowledge elements. The total number of actions need to integrate the interdependencies in a vessel's distributed system configuration is measured by N_A, to number of nodes in the action layer of the K-A-D network model, which is calculated using Equation 6.5. An increase in the number of integration actions between two configurations suggests the design activity with the larger number of actions will require more effort to complete. Applied to ensembles of

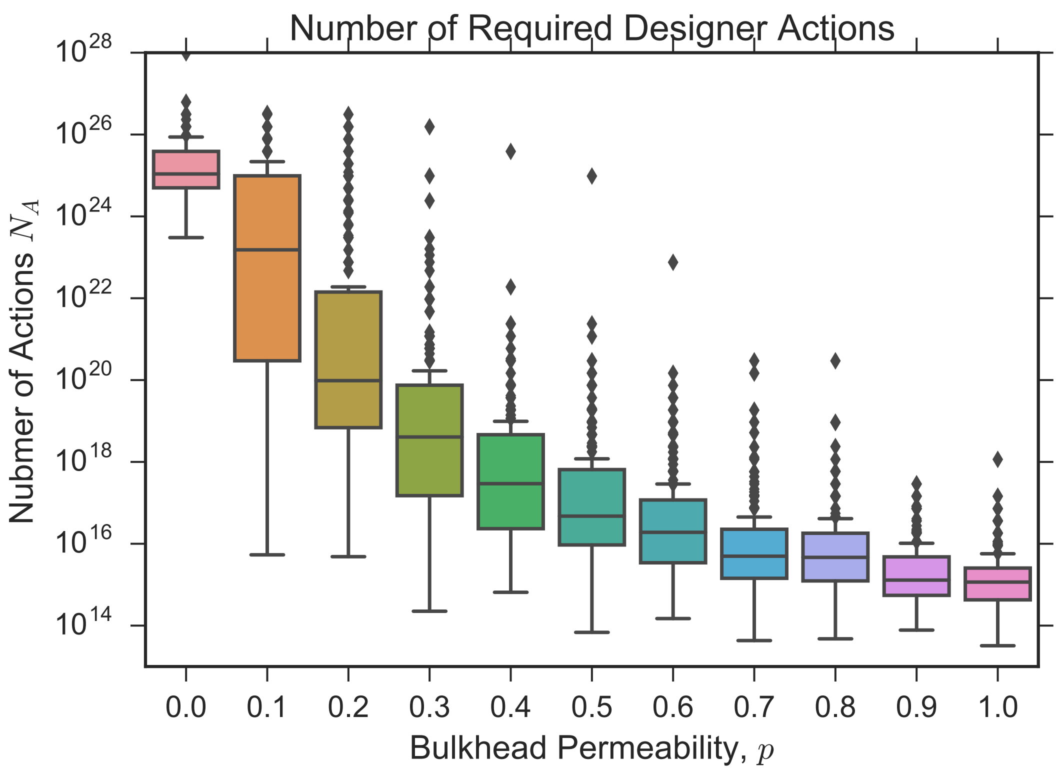


Figure 7.6: Number of actions for distributed system design at each ensemble defined by the level of watertight bulkhead permeability. Higher values of N_A show that more actions are required to complete the design activity and indicate increased design effort.

configurations, this approximates the effort for designing a distributed system given the level of bulkhead permeability.

Number designer action analysis was performed for 200 distributed system configurations in ensembles denoted by bulkhead permeability $p = [0.0, 0.1, \ldots, 1.0]$. In this study, the number of systems, high level of system connectivity, and size of the vessel model created a large number of designer actions. In some cases, $N_A \approx 10^{28}$. Due to the magnitude of number of actions in the design activity, it is more intuitive to explore the number of designer actions on by order of magnitude, O_A, where $O_A = log_{10}(N_A)$. The number of actions results are shown in Figure 7.6 and the order of magnitude descriptive statistics are shown in Table 7.1.

p	$\bar{O}_A$	σ_{O_A}
0.0	25.10	0.68
0.1	22.64	2.57
0.2	20.62	2.57
0.3	18.69	2.04
0.4	17.58	1.57
0.5	17.03	1.55
0.6	16.45	1.33
0.7	15.87	1.06
0.8	15.72	1.00
0.9	15.28	0.78
1.0	15.11	0.71

Table 7.1: Mean and standard deviation of $O_A = log_{10}(N_A)$.

The first observation that can be made is the large number of actions required to create a distributed system design knowledge structure. For the examined model, the number of actions can approach 10^{28}, which is large, but not unreasonable considering the scope, timeline, and sophistication of naval design. The importance is that the ranges observed in Figure 7.6 describe design activities that require significantly different numbers of actions to complete. While N_A is far from an exact computation of difficulty, this represents meaningful shifts in knowledge structure development. The large shift in knowledge structure characteristics indicates a significant change in the knowledge structure. This is reminiscent of critical thresholds observed in percolation and network formation, for examples those seen in *Grassberger* (1983), *Newman and Ziff* (2001), and *Radicchi and Arenas* (2013).

Table 7.1 shows there is a negative relationship between bulkhead permeability and knowledge structure size. As permeability increases and the design activity is less constrained because routing can pass through the watertight bulkheads. This results in the knowledge structure size decreasing and the design activity becomes easier to complete. For example, the mean order magnitude decreases by 10^{10} from $p = 0.0$ to $p = 1.0$. The large number of designer actions at $p = 0.0$ is caused by the routing constraints that force the distributed system to route horizontally over the bulkheads.

This creates many routing interactions directly over the damage control deck, which increases the number of knowledge interdependencies that must be integrated by designer actions. Conversely, at $p = 1.0$, routings follow their shortest paths and create less interactions and leads to a lower number designer actions. Extracting this relationship before starting a design activity enables designers to consider early-stage decisions based on the future design difficulty. This information is a potential differentiator for early-stage design decision-making, but the trend does not identify emergent design failures. For example, when the required design effort is high, but consistent, the potential for design failure may be lessened. A design activities could be hard and require considerable effort to complete, but not experience the explosive cost growth and schedule delays associated with design failures.

It has been postulated that design failures occur when the required design effort increases significantly during the design activity. The potential for design failure can be measured within each ensemble. Here, a failure is be described as a design activity that requires more designer actions than expected. For example, a designer may expect that 100 engineering calculations are needed to develop enough information to make a decision about pump size for a chill water distribution system. However, during the design activity, they realize it will require 1,000 calculations. This constitutes a design failure due to increased design effort by one order of magnitude.

In the distributed system design analysis, this type of design failures can be quantified by comparing the number of actions for a design activity to the expected number of actions for that ensemble it came from. The number of action for a single design activity instance is N_A and the expected number of action for an ensemble is $\bar{N_A}$, which is the average number of action across all instances in an ensemble. Considering that the number of actions is an approximation of a design activity and its values are large, design failures can be measured by an order of magnitude difference between the number of actions in a design activity instance and the expected number of actions for its ensemble. To quantify the probability of a potential design failure in an ensemble, a failure threshold is used. For a given failure threshold t, a design failure is said to occur when $N_A/\bar{N}_A \geq 10^t$. Applied to an ensemble, this threshold is used to assess the probability that a design failure on the order of t occurs during a design activity given the ensemble's permeability. Figure 7.7 shows the probability of a design failure at different thresholds for each ensemble.

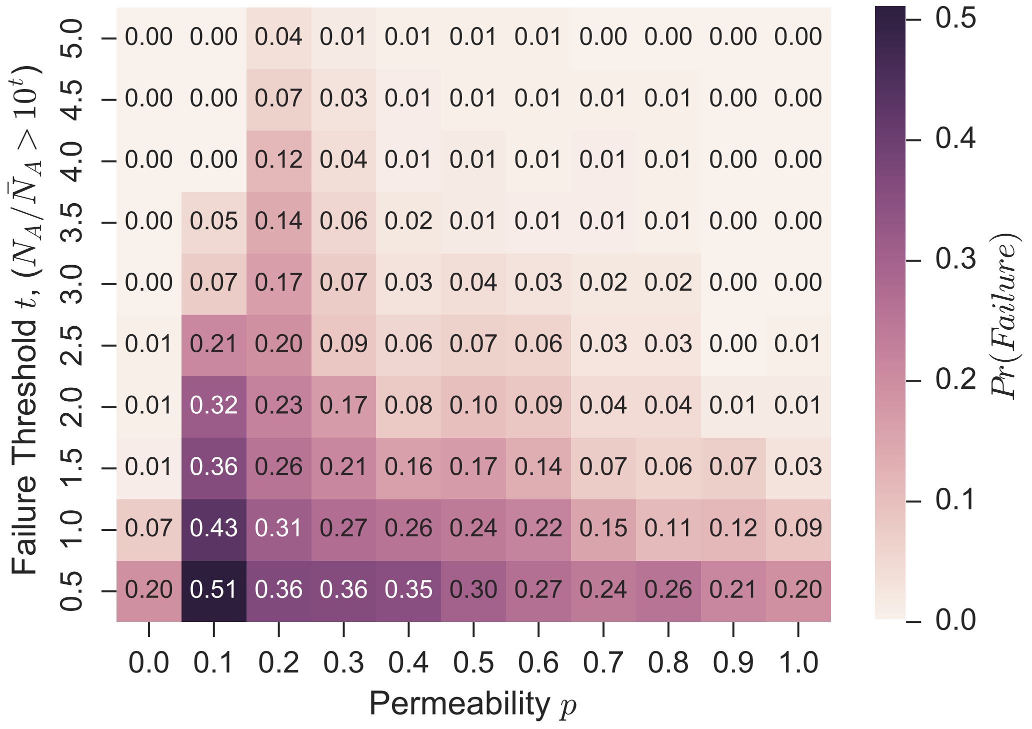


Figure 7.7: Probability of design failure, $10^t\times$ increase from expected number of action $\bar{N}_A$. The x-axis categorizes the ensembles by bulkhead permeability and the y-axis defines the failure threshold. Each cell contains the probability that a design activity from ensemble p results in a design failure on the order of 10^t or greater

Based on Figure 7.7, distributed system design activities can be separated into three segments: safe and hard, safe and easy, and unsafe. The most constrained design activity, $p = 0.0$, has a low probability of experiencing a design failure outside of the lowest threshold, $t = 0.5$, but has the highest expected number of actions.

This ensemble is safe and hard; it is relatively difficult to complete a design activity and the activity is unlikely to experience failure. This is due to the lack of bulkhead permeability, routings are heavily constrained and there is little opportunity for unexpected interdependencies to develop. The result is a decrease in the potential large-scale design failure.

In the mostly unconstrained ensembles, $p > 0.7$, design activities are unlikely to have design failures at thresholds $t \geq 1.0$. This region also has the lowest expected number of design actions, indicating it is safe and easy. In these ensembles, the shortest path routings are mostly uninhibited by bulkheads. This increases the likelihood that routings can follow their unobstructed shortest paths. The result is a decrease in the probability that interdependencies will develop and cause potential design failures.

Design activities in the middle region of ensembles, $0.1 \leq p \leq 0.7$, are unsafe. They have a high probability of potential failure. Compared to $p = 0.0$, this region has lower expected design effort (see Table 7.1), but has the potential to develop failures at all thresholds. There are two distinct failure modes in this region. One at $p = 0.1$, where there is a high probabilities of low-threshold failures (51% for a design activity to require $\approx 3\times$ the expected actions). This failure indicates that there is a high chance the design activity will be harder than expected, but will not have a large-scale failure. The other failure mode occurs in ensembles $0.2 \leq p \leq 0.7$, where there is a non-zero probability of extreme failures. This region has lower overall probability of failure compared to $p = 0.1$, but the failures can be explosive. For example, when $p = 0.2$, there is a 12% chance the design activity requires $10000\times$ more actions than the expected design activity.

In the high failure region, design conditions promote variable knowledge interdependencies. When bulkheads open to allow routings to pass through a small number of places, many horizontal paths between components will use these openings. This introduces the chance that a high number of knowledge interdependences unexpectedly develop. These interdependencies can drastically increase the number of required actions and have greater potential for large-scale design failure.

Regions with low probability of failure develop from the ensemble's consistency. At low permeability, a large number of routings are constrained to pass over the bulkheads when connecting horizontally between components. This increases the knowledge interdependencies associated with certain system environments and drives up the expected number of actions. However, the heavy routing constraints reduce the variability of these interdependencies. This limits the potential for emergent failure. The unconstrained region exhibits similar behavior, but the consistency driven by the shortest routings. When the bulkheads are permeable, the shortest paths between components are unlikely to be blocked by a bulkhead. This limits routing variability which limits the variability in knowledge interdependencies. In both low and high permeability regions, failures are less likely because the design conditions prevent variable knowledge interdependencies.

These results suggest that design activities that are predictably constrained or predictably dominated by an objective (e.g. shortest paths) are less likely to experience design failures. When the design conditions lead to unpredictability, design failures can emerge.

Identifying the conditions that have increased potential to produce emergent design failures is the first step to preventing them. In this case, a small shift in the design activities conditions, from $p = 0.0$ to 0.1, increases the potential for a design activity to require $10\times$ more actions by six-fold. Access to this information gives designers the opportunity to make decisions that reduce the chance they will suffer a design failure. The conditions that lead to emergent failures can be explored in greater detail by identifying the design drivers that cause failures.

7.3 Design Drivers and Design Failures

The distribution of knowledge structure size can elicit what knowledge elements drive the potential for design failures. Described in Chapter VI, the distribution of actions that use each system environment knowledge element can be calculated from the knowledge structure analysis in Equation 6.5. The number of times each element was used by an action that integrates interdependent distributed system knowledge is given by the out-degree k_{ij}, where ij denotes the system environment knowledge element E_{ij}. The distribution of k_{ij} can be examined to determine design drivers that cause design failures. Due to the scaling of k_{ij}, it is more intuitive to consider the mean order of magnitude, $\bar{O_{ij}}$, where $O_{ij} = log_{10}(k_{ij})$. Figure 7.8 shows the distribution of $\bar{O_{ij}}$ in each ensemble using a violin plot.

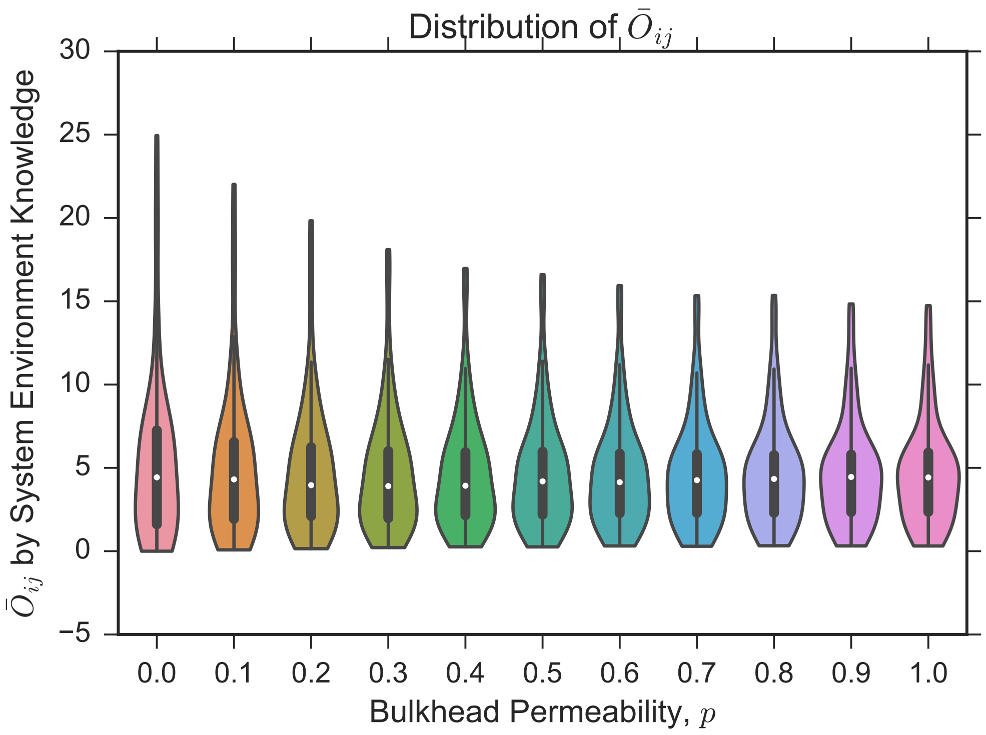


Figure 7.8: Distribution of the order of magnitude of knowledge structure development actions $\bar{O_{ij}}$, where $O_{ij} = log_{10}(k_{ij})$. The distribution of $\bar{O_{ij}}$ is plotted over system environment knowledge element E_{ij}. The plotted shapes illustrate the kernel probability density, i.e. the width of the shaded area represent the proportion of the data located there.

The distribution of actions show that knowledge structure development is driven by a small number of knowledge elements in the upper tails of each distribution. These highly interdependent elements are the expected design drivers that account for the majority of knowledge structure development.

Design failures are caused by an increasing number of actions required to understand the highly interdependent knowledge elements at the upper tail of the distribution. This occurs when unexpected interdependencies with one of these design drivers emerge. Given the exponential scaling in the number of actions that use an element, one additional interdependency with the largest design driver can effectively double knowledge structure size (see Equation 6.4). While this sensitivity can cause emergent failures, it also presents an opportunity to prevent them.

7.4 Design Outcomes and Path Dependencies

Knowledge structure development and the potential for design failure are primarily controlled by a few critical elements. If interdependencies with design drivers can be limited, then potential future design failures can be avoided and design effort can be reduced. Ultimately, this is a question about path dependence. Knowledge interdependencies can be predetermined by the design activity constraints, or they can be created by options in the distributed system configuration. For example, when $p = 0.0$, many interdependencies are guaranteed to exist by constraints that force systems to route horizontally over the damage control deck. In contrast, when $p = 0.5$, the exact bulkhead penetrations and routings will heavily influence the developed interdependencies. Variable interdependencies represent decisions made during the design activity. Accounting for how these decisions affect knowledge structure development models path dependence in the design activity. A similar approach is taken in (*Brown et al.*, 2005) to study path dependence in land usage.

Path dependent elements present opportunities to control knowledge structure development. In particular, path dependent design drivers may be leveraged to prevent design failures and reduce design effort. There are two steps to achieving this. First, the potential for design drivers to be path dependent needs to be evaluated. Path dependent elements are expected to have large variations in their usage depending on the exact configuration. Thus, path dependence appears as a large standard deviation in the distribution of k_{ij} within an ensemble. Due to the scaling of k_{ij}, the standard deviation of the order of magnitude $\sigma_{O_{ij}}$, where $O_{ij} = log_{10}(k_{ij})$, is considered.

For this study, random and path dependent variations in the knowledge structure are differentiated by a standard deviation threshold. The threshold is set at $\sigma_{O_{ij}} = 1$ to demonstrate this analysis as it represents a standard deviation equal to an order of magnitude difference from the expected value. If $\sigma_{O_{ij}} < 1$, the element is considered locked-in by the design activity constraints and the decisions represented by the vessel's physical and logical architectures. In this case, the outcome is the result of *realized* path dependence and variations are treated as random. If $\sigma_{O_{ij}} \geq 1$, the element is treated as having *potential* path dependence because the variability in outcomes suggesting that its influence on the knowledge structure development is not determined by the decisions represented by the physical and logical architectures. Thus future decisions, such as selecting a specific system routing, may significantly shift the knowledge structure outcomes. This differentiation follows from the analysis of realized and potential path dependence proposed in *Bednar and Page* (2017)

Second, knowledge structure elements that are interdependent with design drivers must be identified. If two design drivers are path dependent, but negatively related, shifting interdependencies from one element is likely to increase interdependencies on the other elements. On the other hand, a positive relationship implies that reducing interdependencies for one element is likely to reduce the other's. The relationship between knowledge structure elements can be evaluated by exploiting Equation 6.4,

$$k_{ij} = C_{ij}(f) = 2^{V_{ij}+\sum_l \sum_{s\neq t} \delta_{stl}(i,j)}. \tag{7.1}$$

Taking the log_2 of both sides gives

$$log_2(k_{ij}) = V_{ij} + \sum_l \sum_{s\neq t} \delta_{stl}(i,j), \tag{7.2}$$

where V_{ij} denotes if the physical architecture location has a component, and the summation counts the number of system routings that pass through the location. Equation 7.2 describes the number of knowledge elements that are interdependent with system environment knowledge E_{ij}. These relationships can be quantified by the correlation of $log_2(k_{ij})$ for different elements. The correlation coefficient ρ_{ij}^{kl} denotes the Pearson correlation of Equation 7.2 for knowledge element E_{ij} with element E_{kl}.

Designers have the greatest influence over design outcomes when shifting the most significant design driver. Here, the most significant design driver is defined as the system environment knowledge node with the largest out-degree in the K-A-D network. Out-degree indicates how many times that knowledge element was used in the knowledge structure and thus how many knowledge interdependencies it has. The number of interdependencies increases with the number of systems components and routings that use a space or adjacency in the vessel. Because of this, the primary concern in anticipating knowledge interdependencies is with the knowledge element corresponding to the space or adjacency that has the highest k_{ij}, called $k_{max} = argmax_{i,j\in V} k_{ij}$. Correlations to this element are ρ_{ij}^{max}.

Together $\bar{O_{ij}}$, $\sigma_{O_{ij}}$, and ρ_{ij}^{max} measure the importance of knowledge structure elements, the effect of path dependence, and how a change to the largest design driver is expected influence to the element. Figures 7.9 and 7.10 apply these measures to each ensemble. The results show which knowledge structure elements are path dependent, those over the blue line. The color of each point shows how the importance of that

permeability	slope	intercept	r-squared	p-value	std-error
0.0	0.018	0.390	0.108	0.001	0.005
0.1	0.088	0.671	0.324	0.000	0.013
0.2	0.085	0.799	0.200	0.000	0.017
0.3	0.065	0.878	0.103	0.001	0.019
0.4	0.053	0.899	0.065	0.010	0.020
0.5	0.060	0.822	0.084	0.003	0.020
0.6	0.056	0.784	0.086	0.003	0.018
0.7	0.054	0.729	0.094	0.002	0.017
0.8	0.056	0.651	0.125	0.000	0.015
0.9	0.044	0.608	0.137	0.000	0.011
1.0	0.035	0.475	0.280	0.000	0.006

Table 7.2: Linear regression of $\sigma_{O_{ij}}$ by $\bar{O_{ij}}$ for each permeability.

knowledge element responds to changes of the largest design driver.

The results show that, in all but the edge cases of $p = 0.0$ and 1.0, design drivers are path dependent. In the design activities that had the largest probability to experience design failures $0.1 \leq p \leq 0.2$, design drivers have high standard deviations. The region that is less likely to experience failures, $0.3 \leq p \leq 0.9$, still has path dependent design drivers, but displays a different distribution of points. In this region, design drivers are still path dependent, but have decreased variations. Table 7.2 shows the reduced relationships between $\bar{O_{ij}}$ and $\sigma_{O_{ij}} \geq 1$ as described by linear regression. In the high failure region, the linear regression has a relatively high slope, suggesting that design drivers vary significantly. In the low failure region, the slope decreases, starting with a 25% drop in slope between $p = 0.2$ and $p = 0.3$. This indicates that design drivers tend to have smaller shifts in their contribution to knowledge structure development due to configuration variations.

The linear relationship between p_{ij}^{max}, the correlation to the largest design driver, and $\bar{O_{ij}}$ is shown in Table 7.3 . In the high failure regions, p_{ij}^{max} has a statistically significant positive linear relationship to $\bar{O_{ij}}$. This indicates that positive relationships to design drivers contributes to an increased chance of design failures. This makes intuitive sense. If the unexpected interdependencies that cause design failures

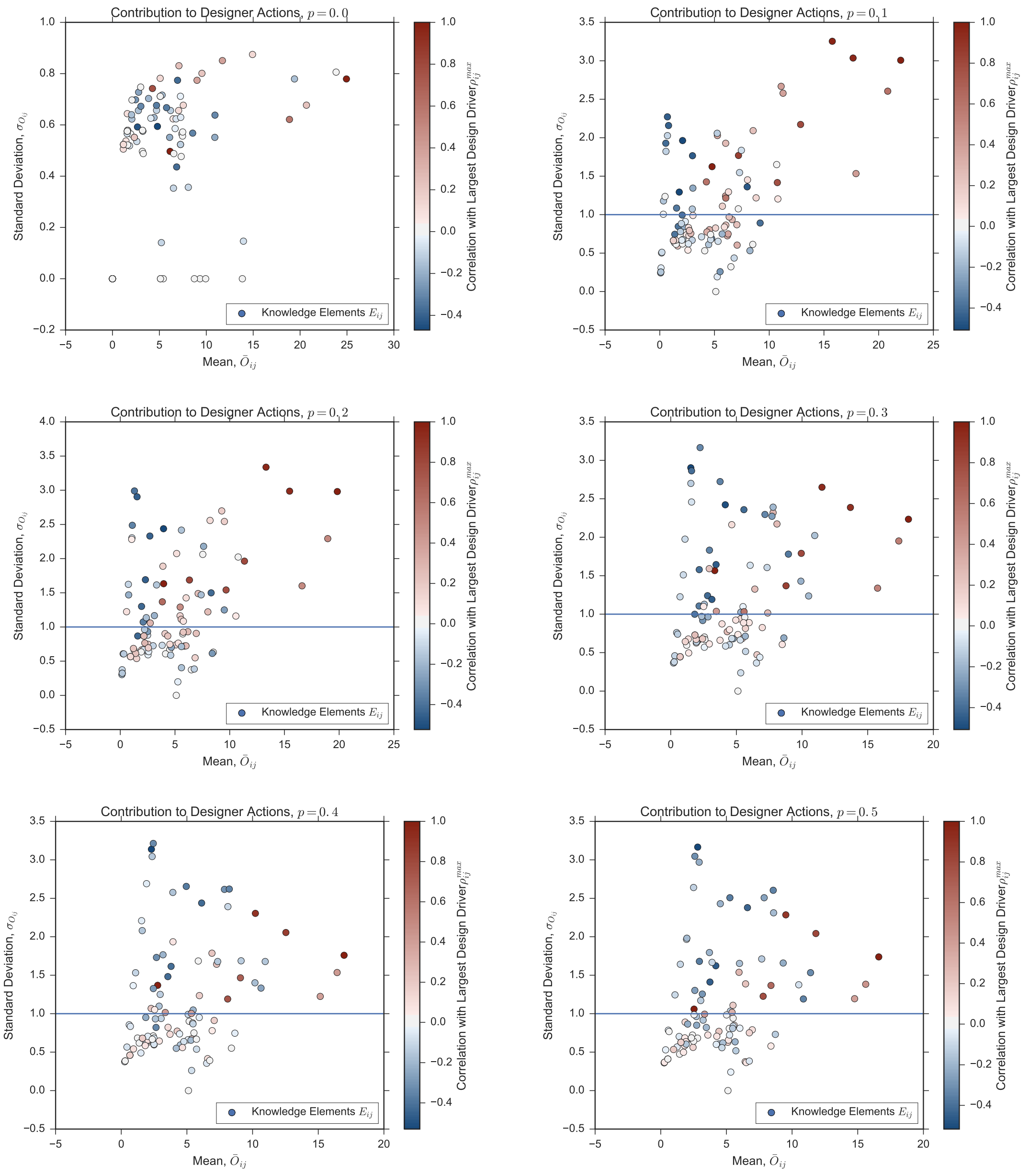


Figure 7.9: Contribution statistics of system environment knowledge E_{ij} to the total number of designer actions N_A for each ensemble, $0.0 \leq p \leq 0.5$. Knowledge structure contribution $\bar{O}_{ij}$ is on the x-axis. Standard deviation of contribution $\sigma_{O_{ij}}$ is on the y-axis. The correlation with largest design driver p_{ij}^{max} is denoted by the color of each point. The threshold of path dependence, $\sigma_{O_{ij}} \geq 1$, is shown by the blue line.

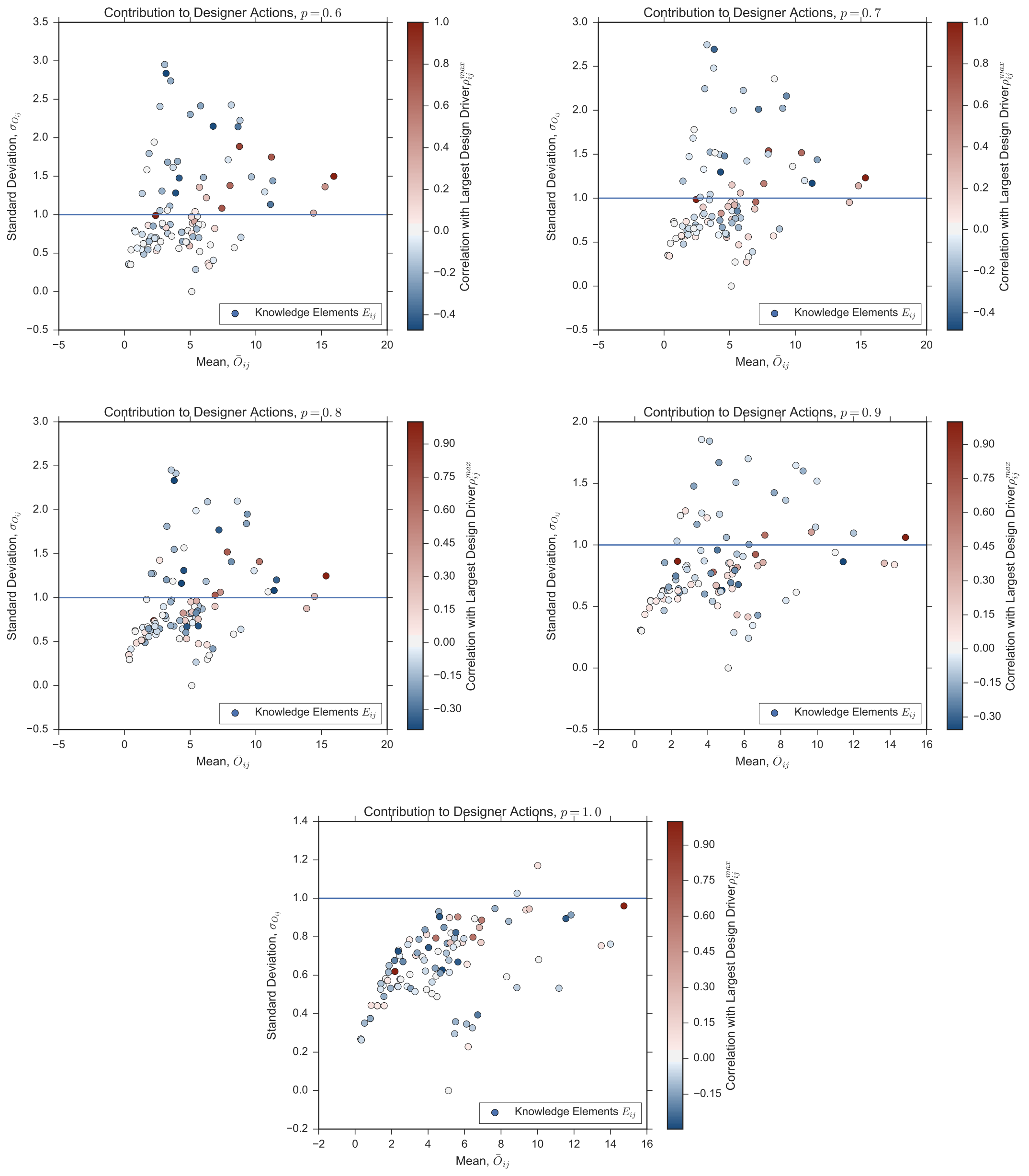


Figure 7.10: Contribution statistics of system environment knowledge E_{ij} to the total number of designer actions N_A for each ensemble, $0.6 \leq p \leq 1.0$. Knowledge structure contribution $\bar{O}_{ij}$ is on the x-axis. Standard deviation of contribution $\sigma_{O_{ij}}$ is on the y-axis. The correlation with largest design driver p_{ij}^{max} is denoted by the color of each point. The threshold of path dependence, $\sigma_{O_{ij}} \geq 1$, is shown by the blue line.

permeability	slope	intercept	r-squared	p-value	std-error
0.0	0.005	-0.035	0.014	0.244	0.004
0.1	0.045	-0.152	0.313	0.000	0.007
0.2	0.039	-0.133	0.211	0.000	0.008
0.3	0.034	-0.138	0.137	0.000	0.008
0.4	0.022	-0.092	0.071	0.007	0.008
0.5	0.017	-0.086	0.041	0.040	0.008
0.6	0.020	-0.103	0.058	0.015	0.008
0.7	0.006	-0.031	0.007	0.410	0.008
0.8	0.004	-0.030	0.003	0.613	0.008
0.9	0.003	-0.016	0.002	0.624	0.007
1.0	0.004	-0.033	0.004	0.527	0.006

Table 7.3: Linear regression of p_{ij}^{max} by $\bar{O_{ij}}$ for each permeability.

impact knowledge structure elements in the same way, small changes can significantly influence design outcomes. In ensembles with higher permeability, there is not a statistically significant linear relationships in the correlation to the largest drivers. Additionally, the correlations in Figure 7.10 show a shift towards uncorrelated or negatively correlated knowledge structure elements. This suggests that the design activities in these ensembles are less likely to experience fluctuations caused by simultaneous shifts in knowledge structure development. Additionally, design drivers have reduced orders of magnitude in the low permeability region. This means that negatively correlated elements have a greater relative influence on the overall knowledge structure size, providing robustness to changes by counter-balancing design driver fluctuations. The reduced design driver variation, reduced correlation, and the balancing effect can be observed in the decreasing standard deviation of O_A in Table 7.1 as well as in the drop in the probability of failure seen in Figure 7.7.

These results suggest that design failures are driven by path dependencies. When design drivers are highly variable and positively correlated, path dependencies can drive massive swings in the design activity outcomes. Stability arises from lower fluctuations and a dampened relationship between design drivers. When designers are cognizant of how knowledge-based interdependencies develop, they may be able to leverage these relationships to systematically reduce the probability of late-stage design failure.

Combined with the previous results in this chapter, this implies that preventing failures can be addressed in two ways:

1. Make design outcomes predictable by limiting variability through design objectives or constraints (e.g. highly constrained routings or shortest paths can be followed).

2. Limit unexpected interdependencies by controlling the knowledge structure around design drivers (e.g. make decisions that to avoid creating interdependencies with design drivers).

The first option may not be applicable for many design activities. Selecting a parameter range that makes design and engineering predictable, may not create feasible products. Further, highly limited designs eliminate innovation and the potential for revolutionary design. The analysis demonstrated here provides a way towards the second, data-driven approach. Considering design drivers with knowledge-based design analysis describes how design outcomes develop. Leveraging patterns in design drivers may help identify ways to reduce design difficulty and prevent emergent design failures.

7.5 Conclusions

This section demonstrates how the K-A-D Framework can be used to identify design failures and prevention measures in naval distributed system design activities. Starting with an early-stage vessel concept, ensembles of approximate late-stage knowledge structures were generated. Analyzing those knowledge structures through the K-A-D network enabled the measurement of design difficulty and the potential of emergent design failures. Investigating design drivers that caused emergent failures identified regions of path dependence in the design activity. In these regions, designer decisions have significant influence on design outcomes. Studying the patterns of design drivers, their level of path dependence, and knowledge interdependencies revealed two ways to reduce the potential for late-stage design failures: limiting the possible design outcomes with constraints or objectives and controlling knowledge structure interdependencies. Using the K-A-D network analysis, design activities that achieve the former can be identified and their expected failure behaviors can be measured. The latter is addressed by calculating design drivers and how assessing how they impact knowledge structure growth. This identifies how design failures emerge and what knowledge interdependencies may need to be controlled to prevent their emergence.

To control the emergence of design failures, the path dependence that influences design drivers need to be harnessed. As decisions about the system configuration are made, they narrow the possible knowledge structure outcomes by locking-in system routing and the knowledge interdependencies they create. Thus, decisions about system routings should be made to reduce the variability of routings through the vessel spaces that represent the path dependent design drivers in the knowledge structure. These decision will narrow the possible outcomes towards overall knowledge structure stability and reduce the potential for design failure. The formal theory behind this process is outside of the scope of this dissertation, but is addressed as future work in Chapter VIII in section "Quantifying and Guiding Design Outcomes".

These methods did not require a detailed product model or expert input to assess the design activity. Instead, characteristics of late-stage design were derived from the early-stage knowledge structure and the knowledge structure development model. This demonstrates the capability of a knowledge-based perspective in preventing the extreme cost-growth and schedule delays caused by emergent design failures.

CHAPTER VIII

Conclusions

Extreme, unanticipated cost growth and schedule delays plague the design of novel naval vessels. Rapid cost increase and long-term delays are caused by design failures that emerge from the integration of designer knowledge and decisions. Preventing these outcomes requires a knowledge-centric perspective of design as opposed to the traditional product-centric perspective. This thesis utilizes complex systems theory to provide the conceptual foundation and mathematical framework to facilitate the knowledge-centric perspective. This final chapter is divided into three parts: the first highlights the major novel contributions of this dissertation; the second part reviews all contributions in detail; and the third part presents areas for future research in knowledge-centric design modeling and analysis

8.1 Major Novel Contributions

Three research questions were presented in Chapter I:

1. Can an objective characterization of the design activities that cause or increase the chance of emergent design failures be developed?
2. Can knowledge-based design complexity be represented, understood, and used?

3. Can complex design activity dynamics provide new insight into emergent naval design failures?

The first major contribution addresses the first question, and was the introduction of a knowledge-centric perspective of naval design activities. This perspective was introduced in Chapter I and expanded in Chapter II. The knowledge-centric perspective of design recognizes that the ideas, concept elements, and evidence used by designers motivate design outcomes, not just the product being designed. From this perspective, many instances of cost overruns and schedule delays in large-scale design activities are knowledge-based design failures. These failures are caused by rapidly increasing design effort that results from the inability to integrate disparate knowledge, excessive rework in the design activity, or other undesirable behaviors that increase the design activity difficulty. Design failures are not inherent to a particular part of a design activity. Instead they emerge from the integration of knowledge and decisions during design. This insight led to a qualitative assessment of the factors of knowledge-based design complexity that create emergent design failures and enabled the quantitative approaches presented later in the dissertation.

The second major contribution ties the first question to the second through the introduction of knowledge structures to understand knowledge-based design complexity. The author argues that designer knowledge structures, the relationships between knowledge that the designer uses to progress the design activity, provide the foundation for considering how knowledge-based complexity influences design outcomes and their success or failure. Chapter III describes knowledge structure elements, how they are related to knowledge-based complexity, and how knowledge structure development can be used to record complex design activity dynamics. Chapter III also argues that path dependence in design activities allows the history of knowledge structure development to be leveraged towards understanding how designer actions and decisions will influence future design activity outcomes.

The third major contribution addresses the second question, and was the definition of the Knowledge-Action-Decision Framework for mapping design knowledge structures. Chapter IV used the qualitative knowledge structure discussions in Chapters II and III to define a mathematical framework for describing knowledge structure development during a design activity. The resulting K-A-D Framework enables design activities to be represented and analyzed as networks in order to identify knowledge structure elements that are important to the design activity's development. These important elements are design drivers that indicate the progression towards design outcomes through path dependent processes. The K-A-D Framework, its network representation, and network analysis were demonstrated on a small example to show how it can be applied to a design activity.

The fourth major contribution addresses the second question, and was the application of the K-A-D Framework to a simulated design activity. This was described in Chapter V. The simulated design activity was formulated to recreate the generation of knowledge for decision-making through time. Modeling the design activity using the K-A-D Framework demonstrated that designer drivers and path dependencies could be identified using network analysis. This showed that knowledge structure development can provide valuable insight into the emergence of design outcomes and that the dynamics behind that emergence can be measured and identified.

The fifth major contribution addresses the third question, and was the application of the knowledge-centric perspective to design failures in naval distributed system design. Chapter VI described an early-stage design technique for generating network-based distributed system configuration models using limited information. Distributed system configuration models were converted into approximate design activity knowledge structures using the K-A-D Framework. These modeling and analysis methods were applied in Chapter VII to a naval vessel concept design. Analysis of the resulting knowledge structures measured the potential for late-stage design failures, identified the knowledge structure elements that were likely to cause design failures, and found opportunities for designers to leverage path dependencies to prevent design failures.

8.2 All Contributions in Detail

1. Can an objective characterization of the design activities that cause or increase the chance of emergent design failures be developed?
    (a) Introduction of a knowledge-centric design perspective and corresponding knowledge-based description of design activities.
    (b) Recognition that the temporal emergence of interdependencies within the knowledge-based system, called knowledge structures, control design outcomes through path dependence.
    (c) Redefined design failure in terms of knowledge structure characteristics and growth.
    (d) Classification of the factors influencing knowledge structure development.
2. Can knowledge-based design complexity be represented, understood, and used?
    (a) Formulated Knowledge-Action-Decision Framework to design activities.
        i. Defined designer knowledge, action, and decisions as the fundamental interdependent elements of knowledge structure development.
        ii. Structured the definitions of knowledge, action, and decisions so that they could be studied temporally to elicit emergent behaviors in the design activity.
    (b) Demonstrated analysis capabilities of the Knowledge-Action-Decision Framework on a simulated design problem.
        i. Defined a network representation of the Knowledge-Action-Decision Framework.

    ii. Applied the network representation to track simulated design activities.
    iii. Introduced interpretations of network analysis methods to measure complex system behaviors in design knowledge structures.
    iv. Demonstrated that network-based knowledge structure analysis can identify emergent designer behavior through time, as modeled in the design simulation.

3. Can complex design activity dynamics provide new insight into emergent naval design failures?

    (a) Re-framed early-stage ship system design as an ensemble analysis of possible late-stage system configuration network representations to identify how early-stage decisions influence late-stage product characteristics.

        i. Expanded network-based models of a vessel physical architecture, logical architecture, and physical system configuration.
        ii. Created methods to generate ensembles of physical system configurations in early-stage ship design.
        iii. Formulated system density analysis to create leading indicators of late-stage design trade-offs between physical system configuration and vessel layout.

    (b) Structured ship system design and integration as a design problem within the Knowledge-Action-Decision Framework.

        i. Identified the knowledge structure characteristics developed during distributed system design using the law of functional complexity.
        ii. Created analysis methods to convert distributed system network representations into knowledge structure characteristics.
        iii. Measured the probabilistic characteristics of system design knowledge

(c) Measured the potential for emergent design failures in distributed system design activities and identified opportunities for designers to prevent them.

   i. Identified the potential for late-stage design failures by applying ensemble knowledge structure analysis to relate early-stage decisions to increased knowledge generation and unpredictable knowledge structures.

   ii. Demonstrated that product-centric analysis cannot elicit the potential for design failures and that knowledge-centric analysis is required.

   iii. Measured the drivers of knowledge structure characteristics to identify opportunities for designers to avoid design failures through their actions.

8.3 Future Topics of Interest

Extending to Acquisition

The knowledge-centric design perspective and K-A-D Framework are envisioned as a way to understand the design outcomes in large-scale design and acquisition programs. This thesis provides the foundation for the broader application of the knowledge-centric perspective, but a full application is outside of the scope of a single thesis. Naval vessel design is unique to the vessel being designed, the stakeholders, and the geo-political landscape of the time. The uniqueness of this activity makes reliance on past experiences an unreliable guide and incurs a greater cost of failure. This makes a rigorous knowledge-centric perspective that can help understand emergent outcomes and potential failures incredibly valuable.

Quantifying and Guiding Design Outcomes

This dissertation provides a new perspective of design outcomes and how they are created. This was enabled, in part, by identifying the role of path dependence in incrementally shifting a design activity towards an outcome. Further formalizing this concept by quantifying how designer actions and decisions change the probability of future outcomes is an important next step. This can be achieved by using the change in entropy over future outcomes to measure how the design activity is narrowing in on a specific outcome. One approach would be to measure the entropy over outcomes by the distribution of centrality over the K-A-D network. When the centrality distribution is highly concentrated, it might indicate a high probability of a certain outcome. This would be reflected by a low entropy. The reverse is also true, a flat centrality distribution might indicate that all outcomes are relatively likely and would have a high entropy. Investigating how potential actions or decisions would shift this entropy measure may help anticipate emergent design failures and design activity convergence. Combined with ensemble analysis, this would harness knowledge-based design complexity to provide new methods to guide and control design outcomes.

Expanded K-A-D Network Analysis

The network analysis of the K-A-D Framework used in this thesis, described in Chapter IV, relied on traditional analysis methods, which were interpreted for their meaning in the multilayer K-A-D network. These methods proved to be sufficiently powerful for the applications in Chapters IV, V and VII. However, there is a growing body of research in multilayer network analysis. Using multilayer-specific analyses may enable different insight into the design activity behaviors and lead to new methods for quantifying design activities.

Renormalization in the K-A-D Framework

Design activities can be described at different levels of detail (*Cash et al.*, 2015). Renormalization is roughly the analysis of how system properties change at different levels of detail. Explorations into the impact of varying levels of detail on the K-A-D Framework may reveal modeling and analysis opportunities. Two areas appear promising. The first is renormalizing the design activity based on Activity Theory, which defines design activities as a series of tasks that are composed of designer actions (*Cash et al.*, 2015). The second promising area is the "information dual," that was developed for street network analysis (*Rosvall et al.*, 2005). The information dual converts a typical street network were road segments are edges and their intersections are nodes in to a representation where streets are nodes and their intersections are edges (*Rigterink*, 2014). This approach has been used for naval system analysis (*Shields et al.*, 2015) and may have applications to naval design activity analysis. Converting series of refined or repetitive knowledge structure elements into "streets" in an information dual may elicit a useful perspective of design activity progression.

Quantifying Design Failure Mitigation

Chapter II qualitatively discussed how Systems Engineering and Set-Based Design mitigate the factors of design complexity to prevent design failures. The author proposed that the respective design activities mitigate failure in different ways based on the knowledge structures that they create. Moving forward, the knowledge structure differences need to be quantified and demonstrated with the K-A-D Framework. This should focus on the knowledge structure differences caused by iterative decision-making (System Engineering), that selects the best alternatives, and convergent decision-making (Set-Based Design), that deselects the worst alternatives. Studying the resulting knowledge structure attributes may reveal how Systems Engineering and Set-Based Design differ from each other and when one should be used over the other.

Decisions in Distributed System Design

Chapter VI modeled naval distributed system design activities by approximating a knowledge structure from possible distributed system configurations. This approach did not incorporate the impact of decisions on the knowledge structure characteristics.

A more realistic model of design failure in naval distributed system design should model designer decisions through the K-A-D Framework. Decisions can be added by “locking-in” parts of the configuration to understand how a decision that commits an element in the product structure will change the resulting knowledge structure characteristics and influence the potential for design failures.

BIBLIOGRAPHY

BIBLIOGRAPHY

ACQuipedia (n.d.), Systems Engineering Process, https://dap.dau.mil/acquipedia, accessed: 2016-02-19.

Akin, Ö., and C. Lin (1995), Design protocol data and novel design decisions, *Design Studies*, *16*(2), 211–236.

Andrews, D. J. (2011), Marine Requirements Elucidation and the Nature of Preliminary Ship Design, *International Journal of Maritime Engineering*, *153 (2011)*.

Andrews, D. J. (2012), Art and science in the design of physically large and complex systems, *Proceedings: Mathematical, Physical and Engineering Sciences*, *468*(2139), 891–912.

Andrews, D. J. (2016), Ship Project Managers Need to be Systems Architects not Systems Engineers, in *Maritime Project Management*, London, United Kingdom, February 2016.

Arena, M. V., I. Blickstein, O. Younossi, and C. A. Grammich (2006), Why Has the Cost of Navy Ships Risen?, *Tech. rep.*, RAND Corporation.

Austin-Breneman, J., B. Y. Yu, and M. C. Yang (2015), Biased Information Passing Between Subsystems Over Time in Complex System Design, *Journal of Mechanical Design*, *138*(1), 11,101.

Axelrod, R. M., and M. D. Cohen (2000), *Harnessing complexity: Organizational implications of a scientific frontier*, Basic Books.

Bahill, A. T., and B. Gissing (1998), Re-evaluating systems engineering concepts using systems thinking, *IEEE Transactions on Systems, Man, and Cybernetics-Part C: Applications and Reviews*, *28*(4), 516–527.

Bar-Yam, Y. (1997), *Dynamics of Complex Systems*, Perseus Books, Cambridge, MA.

Bar-Yam, Y. (2003a), Unifying principles in complex systems, *Converging Technology (NBIC) for Improving Human Performance, MC Roco and WS Bainbridge, Dds., Kluwer*.

Bar-Yam, Y. (2003b), When Systems Engineering Fails - Toward Complex Systems Engineering, *IEEE International Conference on Systems, Man and Cybernetics*, *2*, 2021–2028.

Bedau, M. a. (1997), Weak Emergence, *Philosophical Perspectives, Mind, Causation and World*, *11*(1992), 375–399.

Bednar, J., and S. E. Page (2017), When order affects performance: Behavioral spillovers and institutional path dependence, *forthcoming in American Political Science Reviews.*

Bernstein, B. (1996), *Pedagogy, Symbolic Control & Identity: theory, research & critique*, Rowman & Littlefield.

Bernstein, J. I., and J. Deyst (1998), Design Methods in the Aerospace Industry: Looking for Evidence of Set-Based Practices, Ph.D. thesis, Massachusetts Institute of Technology.

Boccaletti, S., G. Bianconi, R. Criado, C. I. D. Genio, M. R. Jesús Gómez-Gardeñes, I. Sendiña-Nadal, Z. Wang, and M. Zanin (2014), The structure and dynamics of multilayer networks, *Physics Reports*, *544*(1), 1–122.

Braha, D., and Y. Bar-Yam (2007), The Statistical Mechanics of Complex Product Development: Empirical and Analytical Results, *Management Science*, *53*(7), 1127–1145.

Braha, D., and Y. Reich (2003), Topological structures for modeling engineering design processes, *Research in Engineering Design*, *14*(4), 185–199.

Brown, D. G., S. E. Page, R. Riolo, M. Zellner, and W. Rand (2005), Path dependence and the validation of agentbased spatial models of land use, *International Journal of Geographical Information Science*, *19*(2), 153–174.

Buckley, M. E. (2009), Method and system for decision oriented systems engineering (U.S. Patent No. 7,493,298).

Cash, P., and M. Štorga (2015), Multifaceted assessment of ideation: using networks to link ideation and design activity, *Journal of Engineering Design*, *26*(10-12), 391–415.

Cash, P., T. Stankovic, and M. Storga (2014), Using visual information analysis to explore complex patterns in the activity of designers, *Design Studies*, *35*(1), 1–28.

Cash, P., B. Hicks, and S. Culley (2015), Activity Theory as a means for multi-scale analysis of the engineering design process: A protocol study of design in practice, *Design Studies*, *38*, 1–32.

Chi, M. T. (1978), Knowledge structures and memory development, in *Children's thinking: What develops*, chap. 1, Psychology Press.

Cohen, W. M., and D. A. Levinthal (1990), Absorptive Capacity : A New Perspective on Learning and Innovation, *Administrative science quarterly*, pp. 128–152.

Craggs, J., D. Bloor, B. Tanner, and H. Bullen (2004), Naval compensated gross tonnage coefficients and shipyard learning, *Journal of Ship Production*, *20*(2), 107–113.

Cross, N. (1997), Creativity in design: analysing and modelling the creative leap, *Leonardo*, *30*(4), 311 –317.

Cross, N. (2001), Design cognition: results from protocol and other empirical studies of design activity, in *Design knowing and learning: cognition in design education*, pp. 79–103, Elsevier, Oxford, UK.

Cross, N., K. Dorst, H. Christiaans, and (Eds.) (1996), *Analysing design activity*, Wiley.

Dickerson, C., and D. N. Mavris (2016), *Architecture and principles of systems engineering*, CRC Press.

Dinar, M., J. J. Shah, J. Cagan, L. Leifer, J. Linsey, S. M. Smith, and N. V. Hernandez (2015), Empirical Studies of Designer Thinking: Past, Present, and Future, *Journal of Mechanical Design*, *137*(2), 1–13.

Dobson, A. T. (2014), Cost Prediction Via Quantitative Analysis of Complexity in U.S. Navy Shipbuilding, Ph.D. thesis, Massachusetts Institute of Technology.

Doerry, N. (2009), Using the Design Structure Matrix to Plan Complex Design Projects, in *ASNE Intelligent Ships Symposium*, pp. 1–15, Philadelphia, PA, May.

Doerry, N., M. Earnesty, C. Weaver, J. Banko, J. Myers, D. Browne, M. Hopkins, and S. Balestrini (2014), Using Set-Based Design in Concept Exploration, in *SNAME Chesapeake Section Technical Meeting, Army Navy Country Club*, Arlington, VA.

Dong, A. (2016), Functional lock-in and the problem of design transformation, *Research in Engineering Design*, pp. 1–19.

Dorigo, M., V. Maniezzo, and A. Colorni (1996), Ant System : Optimization by a Colony of Cooperating Agents, *IEEE Transactions on Systems, Man, and Cybernetics*, *26*(1), 29–41.

Dorst, K., and N. Cross (2001), Creativity in the design process: co-evolution of problem-solution, *Design Studies*, *22*(5), 425–437.

Dosi, G. (1982), Technological paradigms and technological trajectories. A suggested interpretation of the determinants and directions of technical change, *Research Policy*, *11*(3), 147–162.

First Marine International (2005), Findings for the Global Shipbuilding Industrial Base Benchmarking Study Part 1 : Major Shipyards, *Tech. rep.*, First Marine International.

Galunic, D. C., and S. Rodan (1998), Resource Recombinations in the Firm : Knowledge Structures and the Potential for Schumpeterian Innovation, *Strategic Management Journal*, pp. 1193–1201.

Gaspar, H. M., D. H. Rhodes, A. M. Ross, and S. O. Erikstad (2012), Addressing Complexity Aspects in Conceptual Ship Design : A Systems Engineering Approach, *Journal of Ship Production and Design*, *28*(4), 145–159.

Gero, J. (1990), Design Prototypes: A Knowledge Representation Schema for Design, *AI Magazine*, *11*(4), 26.

Gero, J. S. (1996), Creativity, emergence and evolution in Design, *Knowledge-Based Systems*, *9*(1996), 435–448.

Gillespie, J. W. (2012), A Network Science Approach to Understanding and Generating Ship Arrangements in Early-Stage Design, Ph.D. thesis, University of Michigan.

Gillespie, J. W., A. S. Daniels, and D. J. Singer (2013), Generating functional complex-based ship arrangements using network partitioning and community preferences, *Ocean Engineering*, *72*, 107–115.

Goldschmidt, G. (1990), Linkography: assessing design productivity, in *Cyberbetics and System'90, Proceedings of the Tenth European Meeting on Cybernetics and Systems Research*, World Scientific.

Goldschmidt, G. (1995), The designer as a team of one, *Design Studies*, *16*(2), 189–209.

Goldschmidt, G. (2014), *Linkography: unfolding the design process*, MIT Press.

Goldschmidt, G., and M. Weil (1998), Contents and Structure in Design Reasoning, *Design Issues*, *14*(3), 85–100.

Government Accountability Office (2007), Realistic Business Cases Needed to Execute Navy Shipbuilding Programs, *Tech. rep.*, United States Government Accountability Office.

Government Accountability Office (GAO) (2009), High Levels of Knowledge at Key Points Differentiate Commercial Shipbuilding from Navy Shipbuilding, *Tech. Rep. GAO-09-322*, Government Accountability Office.

Grassberger, P. (1983), On the critical behavior of the general epidemic process and dynamical percolation.

Grim, P., D. J. Singer, S. Fisher, A. Bramson, W. J. Berger, C. Reade, C. Flocken, and A. Sales (2013), Scientific Networks on Data Landscapes: Question Difficulty, Epistemic Success, and Convergence, *Episteme-a Journal of Individual and Social Epistemology*, *10*(4), 441–464.

Hansen, C. T., and M. M. Andreasen (2004), A Mapping of Design Decision-Making, in *DS 32: Proceedings of DESIGN 2004, the 8th International Design Conference*, Dubrovnik, Croatia.

Hatchuel, A., and B. Weil (2003), A New Approach of Innovative Design: an Introduction To C-K Theory, in *DS 31: Proceedings of ICED 03, the 14th International Conference on Engineering Design*, pp. 1–15, Stockholm, Sweden.

Hatchuel, A., and B. Weil (2009), C-K design theory: An advanced formulation, *Research in Engineering Design*, *19*(4), 181–192.

Hazelrigg, G. A. (1996), *Systems engineering: an approach to information-based design*, Pearson College Division.

INCOSE (n.d.), What is Systems Engineering, http://www.incose.org/AboutSE/WhatIsSE, accessed: 2017-04-10.

Kan, J. W. T., and J. S. Gero (2008), Acquiring information from linkography in protocol studies of designing, *Design Studies*, *29*(4), 315–337.

Kana, A. A., C. P. F. Shields, and D. J. Singer (2016), Why is Naval Design Decision-Making so Difficult?, in *Warships 2016: Advanced Technologies in Naval Design, Construction, & Operation*, Bath, United Kingdom, June.

Keane, R. G. (2011), Reducing Total Ownership Cost: Designing Inside-Out of the Hull, in *ASNE Day 2011*, Arlington, VA, February.

Keane, R. G., L. Deschamps, and S. Maguire (2015), Reducing Detail Design and Construction Work Content by Cost-Effective Decisions in Early-Stage Naval Ship Design, *Journal of Ship Production and Design*, *31*(3), 1–14.

Keane, R. G., T. S. Mierzwicki, and G. R. Grogan (2016), Designing out complexity early: A path to flexible warships, in *ASNE Day 2016*, Arlington, VA, February.

Keane, R. G., T. S. Mierzwicki, and G. R. Grogan (2017), Designing Out Complexity Early: A Path to Flexible Warships, *Naval Engineers Journal*, *129*(1), 25–40.

Kim, E. J., and K. M. Kim (2015), Cognitive styles in design problem solving: Insights from network-based cognitive maps, *Design Studies*, *40*, 1–38.

Kivelä, M., U. R. I. Virgili, and J. P. Gleeson (2014), Multilayer Networks, *Journal of complex networks*, *2*(3), 203–271.

Kleinberg, J. M. (1999), Authoritative Sources in a Hyperlinked Environment, *Journal of the ACM*, *46*(5), 604–632.

Kossiakoff, A. (2011), *Systems Engineering - Principles and Practice*, Wiley.

Kotha, S., and K. Srikanth (2013), Managing A Global Partnership Model: Lessons from the Boeing 787 Dreamliner' Program, *Global Strategy Journal*, *3*, 41–66.

Kroll, E., P. Le Masson, and B. Weil (2014), Steepest-first exploration with learning-based path evaluation: uncovering the design strategy of parameter analysis with CK theory, *Research in Engineering Design*, *25*(4), 351–373, doi:10.1007/s00163-014-0182-8.

Kusiak, A. (1993), *Concurrent Engineering: Automation, Tools and Technique*, John Wiley & Sons.

Lamberson, P., and S. E. Page (2012), Tipping points, *Quarterly Journal of Political Science*, *7*(2), 175–208.

Lawson, B. (1979), Cognitive strategies in architectural design, *Ergonomics*, *22*(1), 59–68.

Laxton, M. (1969), Design education in practice, *Attitudes in Design Education.*

Levinthal, D. A., and M. Warglien (1999), Landscape Design: Designing for Local Action in Complex Worlds, *Organization Science*, *10*(3), 342–357.

Lyles, M. A., and C. R. Schwenk (1992), Top management, strategy and organizational knowledge structure, *Journal of Management Studies*, *29*(2), 155–174.

Maher, M. L., J. Poon, and S. Boulanger (1996), Formalising Design Exploration as Co-evolution, in *Advances in Formal Design Methods for CAD*, pp. 3–30, Springer US.

Mahoney, J. (2000), Path Dependence in Historical Sociology, *Theory and Society*, *29*(4), 507–548.

Markus, H., M. Crane, S. Bernstein, and M. Siladi (1982), Self-Schemas and Gender, *Journal of Personality and Social Psychology*, *42*(1), 38–50.

Maton, K. (2007), Knowledge-knower structures in intellectual and educational fields, in *Language, Knowledge and Pedagogy: Functional linguistic and sociological perspectives*, pp. 87–108, Bloomsbury Publishing.

Mavris, D., and D. DeLaurentis (2000), Methodology for Examining the Simultaneous Impact of Requirements, Vehicle Characteristics, and Technologies on Military Aircraft Design, in *22nd Congress of the International Council on the Aeronautical Sciences (ICAS)*, Harrogate, England, August 2000.

McKenney, T. A. (2013), An Early-Stage Set-Based Design Reduction Decision Support Framework Utilizing Design Space Mapping and a Graph Theoretic Markov Decision Process Formulation, Ph.D. thesis, University of Michigan.

McKenney, T. A., and D. J. Singer (2014), Set-Based Design, *Marine Technology.*

Mihm, J., and C. H. Loch (2006), Spiraling out of control: Problem-solving dynamics in complex distributed engineering projects, in *Complex Engineered Systems*, pp. 141–157, Springer Berlin Heidelberg.

Miles, R. E., C. C. Snow, A. D. Meyer, and H. J. Coleman (1978), Organizational Strategy, Structure, and Process, *Academy of management review*, *3*(3), 546–562.

Miller, J. H., and S. E. Page (2008), *Complex adaptive systems: an introduction to computational models of social life*, Princeton University Press.

Miroyannis, A. (2006), Estimation of Ship Construction Costs, Ph.D. thesis, Massachusetts Institute of Technology.

Mitchell, M. (2009), *Complexity: A Guided Tour*, Oxford University Press, New York, NY.

NAVSEA (2012), Ship Design Manager (SDM) and Systems Integration Manager (SIM) Manual.

Newman, M. E. J. (2003), The structure and function of complex networks, *SIAM Review*, *45*(2), 167–256.

Newman, M. E. J. (2010), *Networks: an introduction*, 720 pp., Oxford University Press, Oxford.

Newman, M. E. J., and R. M. Ziff (2001), A fast Monte Carlo algorithm for site or bond percolation, *Physical Review E*, *64*(1), 016,706.

O'Rourke, R. (2015), Navy Ford (CVN-78) Class Aircraft Carrier Program : Background and Issues for Congress, *Tech. rep.*, Congressional Research Service.

O'Rourke, R. (2017a), Navy Ford (CVN-78) Class Aircraft Carrier Program: Background and Issues for Congress, *Tech. rep.*, Congressional Research Service.

O'Rourke, R. (2017b), Navy Littoral Combat Ship (LCS)/ Frigate Program: Background and Issues for Congress, *Tech. rep.*, Congressional Research Service.

O'Rourke, R. (2017c), Force Structure and Shipbuilding Plans: Background and Issues for Congress, *Tech. rep.*, Congressional Research Service.

Oxman, R. E. (1990), Prior knolwedge in design: a dynamic knowledge-based model of design and creativity, *Design Studies*, *11*(1), 17–28.

Page, S. E. (2006), Path Dependence, *Quarterly Journal of Political Science*, *1*, 87–115.

Page, S. E. (2008), Uncertainty, Difficulty, and Complexity, *Journal of Theoretical Politics*, *20*(2), 115–149.

Page, S. E. (2010), *Diversity and complexity*, Princeton University Press.

Parker, M. C. (2014), A Contextual Multipartite Network Approach to Comprehending the Structure of Naval Design, Ph.D. thesis, University of Michigan.

Parker, M. C., and D. J. Singer (2015), Analyzing the dynamic behavior of marine design tools using network theory, *Ocean Engineering*, *106*, 227–237.

Pierson, P. (2000), Increasing Returns, Path Dependence, and the Study of Politics, *American political science review*, *94*(2), 251–267.

Pierson, P. (2004), *Politics in time: History, institutions, and social analysis*, Princeton University Press.

Radicchi, F., and A. Arenas (2013), Abrupt transition in the structural formation of interconnected networks, *Nature Physics*, *9*(10), 717–720, doi:10.1038/nphys2761.

Reich, Y. (1995), Measuring the value of knowledge, *Int. J. Human-Computer Studies*, *42*(1), 3–30.

Rigterink, D. T. (2014), Methods for Analyzing Early Stage Naval Distributed Systems Designs, Employing Simplex, Multislice, and Multiplex Networks, Ph.D. thesis, University of Michigan.

Rittel, H. W., and W. M. Melvin (1973), Dilemmas in a General Theory of Planning, *Policy sciences*, *4*(2), 155–169.

Rosvall, M., A. Trusina, P. Minnhagen, and K. Sneppen (2005), Networks and cities: An information perspective, *Physical Review Letters*, *94*(2), 028,701.

Rycroft, R. W., and D. E. Kash (2002), Path dependence in the innovation of complex technologies, *Technology Analysis & Strategic Management*, *14*(1), 21–35.

Schank, J. F., C. G. Pernin, M. V. Arena, C. C. Price, and S. K. Woodward (2009), Controlling the Cost of C4I Upgrades on Naval Ships, *Tech. rep.*, RAND NATIONAL DEFENSE RESEARCH INST, Santa Monica, CA.

Schank, R. C., and R. P. Abelson (2013), *Scripts, plans, goals, and understanding: An inquiry into human knowledge structures*, Psychology Press.

Shields, C. P. F., D. T. Rigterink, and D. J. Singer (2015), The Information Dual Network and its Applications to Naval Architecture, in *12th International Marine Design Conference*, vol. 2, pp. 1–8, Tokyo, Japan, May.

Shields, C. P. F., J. T. Knight, and D. J. Singer (2016a), The Design Process as a Complex System, in *ASNE Day 2016*, Arlington, Virginia, February.

Shields, C. P. F., M. J. Sypniewski, and D. J. Singer (2016b), Understanding the relationship between naval product complexity and on-board system survivability using network routing and design ensemble analysis, in *International Symposium on Practical Design of Ships and Other Floating Structures 2016*, pp. 219–225, Copenhagen, Denmark, September.

Shields, C. P. F., D. T. Rigterink, and D. J. Singer (2017), Investigating physical solutions in the architectural design of distributed ship service systems, *Ocean Engineering*, *135*(January), 236–245.

Sim, S. K., and A. H. B. Duffy (2004), Evolving a model of learning in design, *Research in Engineering Design*, *15*(1), 40–61.

Simon, H. A. (1969), *The sciences of the artificial*, MIT Press.

Singer, D. J., N. Doerry, and M. E. Buckley (2009), What Is Set-Based Design?, *Naval Engineers Journal*, *121*(4), 31–43.

Smith, R. P., and S. D. Eppinger (1997), Identifying Controlling Features of Engineering Design Iteration, *Management Science*, *43*(3), 276–293.

Sobieszczanski-Sobieski, J., and R. T. Haftka (1997), Multidisciplinary Aerospace Design Optimization: Survey of Recent Developments, *Structural Optimization*, *14*(1), 1–23.

Sparrow, P. (1999), Strategy and Cognition: Understanding the Role of Management Knowledge Structures, Organizational Memory and Information Overload, *Creativity and Innovation Management*, *8*(2), 140.

Under Secretary of Defense Acquisition Technology and Logistics (US[AT&L]) (2013), Performance of the Defense Aquisition System, 2013 Annual Report, *Tech. rep.*, Under Secretary of Defense Acquisition Technology and Logistics (USD[AT&L]), Washington, D.C.

Under Secretary of Defense Acquisition Technology and Logistics (US[AT&L]) (2014), Performance of the Defense Acquisition System, 2014 Annual Report, *Tech. rep.*, Under Secretary of Defense Acquisition Technology and Logistics (USD[AT&L]), Washington, D.C.

United States Government Accountability Office (2015), LITTORAL COMBAT SHIP: Knowledge of Survivability and Lethality Capabilities Needed Prior to Making Major Funding Decisions, *Tech. rep.*, United States Government Accountability Office.

United States Navy (2003), Preliminary Design Interim Requirements Document for Littoral Combat Ship (LCS) Flight 0, *Tech. rep.*, United States Navy.

van der Lugt, R. (2000), Developing a graphic tool for creative problem solving in design groups, *Design Studies*, *21*(5), 505–522.

van der Lugt, R. (2001), *Sketching in design idea generation meetings*, TU Delft, Delft University of Technology.

Visser, W. (1995), Use of episodic knowledge and information in design problem solving, *Design Studies*, *16*(2), 171–187.

Visser, W., and B. Trousse (1993), Reuse of designs: an interdisciplinary cognitive approach, in *Thirteenth international joint conference on artificial intelligence workshop.*

Ward, A. C., J. K. Liker, J. J. Christiano, and D. K. Sobek (1995), The Second Toyota Paradox: How Delaying Decisions Can Make Better Cars Faster, *Sloan Management Review.*

Ward, T. B., S. M. Smith, and J. E. Vaid (1997), *Creative thought: An investigation of conceptual structures and processes*, American Psychological Association.

Wheelwright, S. C., and K. B. Clark (1992), *Revolutionizing Product Development: Quantum Leaps in Speed, Efficiency, and Quality*, New York: The Free Press.

Work, R. O. (2014), *The Littoral Combat Ship: How We Got Here, and Why*, OFFICE OF THE UNDER SECRETARY OF THE NAVY PENTAGON WASHINGTON DC.