3-2023

Illuminating the Unknown: A Mixed Methods Exploration of the DoD Software Factory Ecosystem

Zachary O. Ryan

ILLUMINATING THE UNKNOWN: A MIXED METHODS EXPLORATION OF THE DOD SOFTWARE FACTORY ECOSYSTEM

THESIS

Zachary O. Ryan, Capt, USAF

AFIT-ENV-MS-23-M-229

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-23-M-229

ILLUMINATING THE UNKNOWN: A MIXED METHODS EXPLORATION OF THE DOD SOFTWARE FACTORY ECOSYSTEM

THESIS

Presented to the Faculty

Department of Systems Engineering and Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Acquisitions and Program Management

Zachary O. Ryan, BS

Capt, USAF

March 2023

AFIT-ENV-MS-23-M-229

ILLUMINATING THE UNKNOWN: A MIXED METHODS EXPLORATION OF THE DOD SOFTWARE FACTORY ECOSYSTEM

Zachary O. Ryan, BS

Capt, USAF

Committee Membership:

Dr. Mark Reith
Chair

Lt Col Paul Beach, PhD
Member

Lt Col Clay Koschnick, PhD
Member

Abstract

The Department of Defense's (DoD) software factories are a collection of modern software acquisition programs that commonly employ the agile, network-based business strategies often found within commercial industries. Having been formally recognized by senior leaders for their revolutionary software development approaches, the software factories highlight a cultural shift within the DoD away from traditional organizational practices. As a result of the factories demonstrated successes, the number of programs employing non-traditional strategies is expanding. While this is notable, it also presents a challenge because a comprehensive understanding of the characteristics, structures, and behaviors of the DoD's software factories does not currently exist.

This thesis addresses this knowledge gap by employing a sequential mixed methods methodology to explore the organizational characteristics and structures of the DoD's software factories using a three-phased research approach designed to facilitate active community engagement and feedback. Primary research data was collected from the software factory community through personnel interviews, participant observation, and a case study. Results from this research include a software factory characterization framework, a structural definition of software factories, and a new programmatic assessment process designed to help acquisition practitioners understand the organizational behaviors of non-traditional programs. The composition of these efforts provides senior leaders and acquisition professionals with new insights into the fundamental organizational components of the DoD's software factory ecosystem.

Acknowledgments

I would like to express my sincere gratitude to my family for their unwavering support and encouragement throughout my academic journey. Without their love and belief in me, I would not have been able to achieve the success that I have today.

I would also like to thank my advisors, Dr. Mark Reith, Lt Col Paul Beach, and Lt Col Clay Koschnick for their guidance, mentorship, and patience. Their expertise and insights have been invaluable and I am deeply grateful for the time and energy that they dedicated to my academic and personal growth.

Finally, I would like to thank the senior leaders within the software factory community who took time out of their busy schedules to provide me with their personal insights and to those who provided me with the opportunities and resources necessary to complete this thesis. I have learned so much from this community and I am deeply grateful for the support and encouragement they provided along the way.

Zachary O. Ryan

Table of Contents

Page

Page

List of Figures

Page

List of Tables

Page

ILLUMINATING THE UNKNOWN: A MIXED METHODS EXPLORATION OF THE DOD SOFTWARE FACTORY ECOSYSTEM

I. Introduction

General Issue

The Department of Defense (DoD) acquisitions community, the Congress, and the military services do not currently have a shared understanding of the characteristics, structures, or functions of the non-traditional software acquisitions programs called software factories. This knowledge gap is consistent with a longstanding trend as the DoD has historically struggled to manage its software acquisitions programs; a challenge it has acknowledged as a department wide issue dating back to the 1970s. Recent official guidance reiterates these difficulties with the most recent Defense Innovation Board Report titled *Software is Never Done*, stating "Countless past studies have recognized the deficiencies in software acquisition and practices within DoD, but little seems to be changing" (McQuade et al., 2019).[1] Evidence illustrating the department's lack of shared knowledge into the software factories also extends beyond historical and recent reports; it is evident in the recently observed actions and statements of the Congress, the DAF CSO, and the greater defense acquisition community. A recent congressional inquiry into the structures of the software factory ecosystem, and the cascade of actions it spurred within

[1] A historical overview of the Department of Defense's software acquisition challenges from 1970-2022 is included in Appendix A. Appendix B includes an in-depth overview of the main lines of effort highlighted in the 2019 Defense Innovation Board study.

the defense community, illustrates the fractured and inconsistent nature of the departments collective knowledge and understanding of these new, non-traditional software organizations.

On July 27, 2022, the Congress issued a tasking to the Secretary of the Air Force formally recognized the "significant contributions" that the software factories had made to the Department of Defense's software modernization efforts. In accompaniment to this recognition, the Congress directed the Department of the Air Force (DAF) to provide a formal briefing to the House Armed Services Committee (HASC) outlining its plan for the structure of the software factory ecosystem by January 1, 2023. In response to this congressional tasker, the DAF Chief Software Office (CSO) coordinated with Air Force Materiel Command (AFMC) to lead the establishment of a series of working groups tasked with understanding the core components of the greater software ecosystem.

Acknowledging that a comprehensive and universally accepted strategic overview of the software factory ecosystem did not exist, seven working groups were established to study specific areas of the ecosystem and to develop a comprehensive roadmap for the software factories. Those working groups were: strategic capability, procurement management, talent management, cost management, operations and deployment, software resiliency, and software architecture (AFMC, 2022). The strategic capability working group, established as the AFMC lead, specifically focused on formalizing the relationships, capabilities, processes, and structures of these organizations. This thesis feeds and informs the ongoing efforts of the strategic capability working group and the DAF CSO by providing an outside, research-centric perspective into the fundamental building blocks that compose the software factory ecosystem.

Problem Statement

There is a pressing need to *understand, define,* and *describe* the Department of Defense's (DoD) software factories. Beginning with Kessel Run (Perkins & Long, 2020), the software factories have existed in some form for almost five years yet the debate about how to conceptualize, define, and subsequently manage these organizations has remained largely unsettled in both the defense acquisitions and software communities. This lack of a clear and shared understanding has led to a fragmented factory ecosystem and it has caused decision-makers to struggle with reconciling the often unspoken and socially defined constructs used within the factory community with the formal organizational definitions required for executive oversight.

As a result of the widespread interpretations of what defines the DoD's software factories, misconceptions and misunderstandings are common because a shared language that illustrates the values, goals, and visions of the two communities does not exist. Addressing this divide is paramount to the continued success of the greater software ecosystem. Recognizing this, the overarching premise of this thesis is to develop a clearer and more comprehensive understanding of the software factories, and in doing so, establish a common theoretical foundation that can be used by both the software development and acquisitions communities to enable more effective cross-domain communication and collaboration.

Research Objectives/Questions

The overarching theme of this thesis is one of initial exploration and foundational understanding. Existing literature on the organizational behaviors, characteristics, and structures of software factories is largely non-existent and the community is

geographically and culturally diverse (Assistant Secretary of Acquisition & Air Force Chief Software Office, 2021). Given this pretext, and understanding that the software factories are largely unstudied from an organizational standpoint, this thesis employs an iterative, three phased research approach designed to methodically deconstruct the software factory ecosystem despite its immaturity. Within each phase of research, this thesis employs a variety of methods to include interviews, literature reviews, observational analysis, network analysis, and a case study analysis in order to make sense of the fractured and disparate data sources that compose the collective understanding of defense acquisitions knowledge on software factories.

This thesis follows a scholarly format and it is comprised of three separate manuscripts that align with the objectives: *Understand*, *Define*, and *Describe*. This iterative approach addresses the overarching theme of this thesis, to explore and understand the DoD's software factories, by breaking it into three incrementally consumable objectives. These objectives are:

Understand the characteristics of software factories.

The first objective is to gain a baseline understanding of software factories by identifying and defining their key characteristics from an organizational perspective. This will involve analyzing the common traits and features of these organizations and using this information to develop a more comprehensive picture of their underlying structures and functions.

Primary Research Question: What are the defining characteristics of software factories?

Define the software factories.

The second objective of this thesis is to develop a formal definition for software factories that is accessible to both software factory professionals and career acquisitions practitioners. This objective is intended to build upon the first by comparing the characteristics of software factories to existing organizational definitions with the intent of identifying commonalities within their organizational structures to inform a unifying definition. Establishing this definition will help to remove the underlying barriers to communication that currently exist between the software factory and acquisitions communities. Additionally, this will help reduce inter-organizational conflict by establishing a shared perspective that both communities can reference.

Primary Research Question: What are the organizational structures of the software factories?

Describe the software factories.

The third objective of this thesis is to expand upon the definition of the software factory by describing it using a repeatable methodology. In addition to this description, the development of a methodology will inform, educate, and equip the defense community with new tools and techniques necessary to understand, define, and describe other non-traditional organizations like software factories in a way that is repeatable and accessible to the general acquisition practitioner.

Primary Research Question: How can acquisition practitioners analyze non-traditional organizations like software factories?

Despite the increasing importance of the software factories within the Department of Defense, there exists a lack of understanding about what defines and differentiates them from other acquisitions organizations. This lack of understanding limits the defense community's ability to predict and explain the behaviors of these organizations as well as hinders the development of strategic policy. If decision makers are going to effectively manage the software factories at the departmental level, they must first understand what they are and how they fit within the greater defense acquisitions architecture. This thesis acknowledges those shortfalls while simultaneously employing an approach designed to facilitate active engagement with the software factory community to inform thesis progression while staying abreast of new information and data sources as they become available.

Methodology

This thesis employs a sequential research design following the mixed methods methodology. The mixed methods methodology emphasizes a research approach that employs different types of qualitative and quantitative methodological techniques based on the unique demands of the research topic, availability of data, and existing knowledge on the subject (Tashakkori & Creswell, 2007; Tashakkori & Teddlie, 2003, 2009). Mixed methods approaches are also said to be "pragmatic" because the selection of method within the research cycle is driven by the immediate needs and values of the researcher and their interpretation of the area of study (Tashakkori & Teddlie, 2003). While mixed methods research designs often employ a combination of qualitative and quantitative approaches, they can also employ different types of qualitative or quantitative approaches depending on the needs of the researcher. Since this thesis

focuses on a topic area that remains largely unstudied by academics, it primarily employs qualitative methods designed to establish theory and identify organizational phenomenon.

The mixed methods methodology employs a cyclical pattern of research called the "inductive-deductive research cycle" (Figure 2). This cycle captures both inductive and deductive reasoning processes by illustrating how theory and hypothesis are informed by and inform the collection and understanding of observations, facts, and evidence. Since software factory structures have not been previously studied, this thesis begins the cycle with inductive reasoning, beginning with the observational stage. Inductive reasoning is then used to develop theory from existing data, a hypothesis is developed, and a method of analysis is selected and conducted to test the hypothesis. Results from the analysis are then integrated with existing data and the research cycle is reiterated.

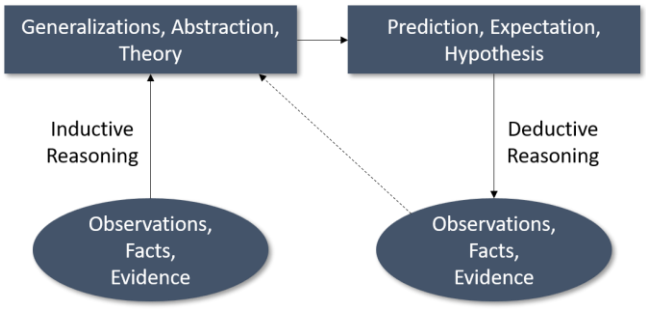

Figure 1: The inductive-deductive research cycle **(Tashakkori & Teddlie, 2009)**.

Mixed methods research designs vary depending on both the primary orientation of the research, (qualitative or quantitative) and the temporal nature of the research stages, (parallel or sequential) (Tashakkori & Teddlie, 2009). Considering these major components, this thesis utilizes a sequential design (meaning the research is broken up into sequential phases or threads), with the results of each phase informing and shaping its successors. The three primary thread designs used in this thesis are narrative research, participant observation, and a case study.

Supporting the exploratory theme of this thesis, the three threads were selected based on their suitability for assisting researchers in understanding undefined phenomenon and problems. All of the selected research designs are primarily qualitative, however, some of the data collected from the third phase is quantitative and it has been used to assist in the triangulation of the qualitative narratives through the assessment of network data. The characteristics of the approaches selected for this research are also complementary; the data collection forms, type of problems, analysis strategies, and structure are similar and well suited for understanding and informing poorly defined problems through interviews and observations (Creswell et al., 2007; Jorgensen, 2015).

In summary, this thesis studies the DoD's software factories using the mixed methods methodology. It sequentially iterates through three phases of research primarily utilizing the narrative research, participant observation, and case study approaches. The phases of research are sequential and the results of each sequence influences the next cycle. The mixed methods research cycle begins at the observation stage and inductive reasoning is used to develop general inferences about the software factory. This decision (to begin with observation and inductive reasoning) is necessary due to the limited availability of existing research on the software factory ecosystem. A summary of this thesis, with each stage and its resultant method, output, and conclusion is included in Figure 2.

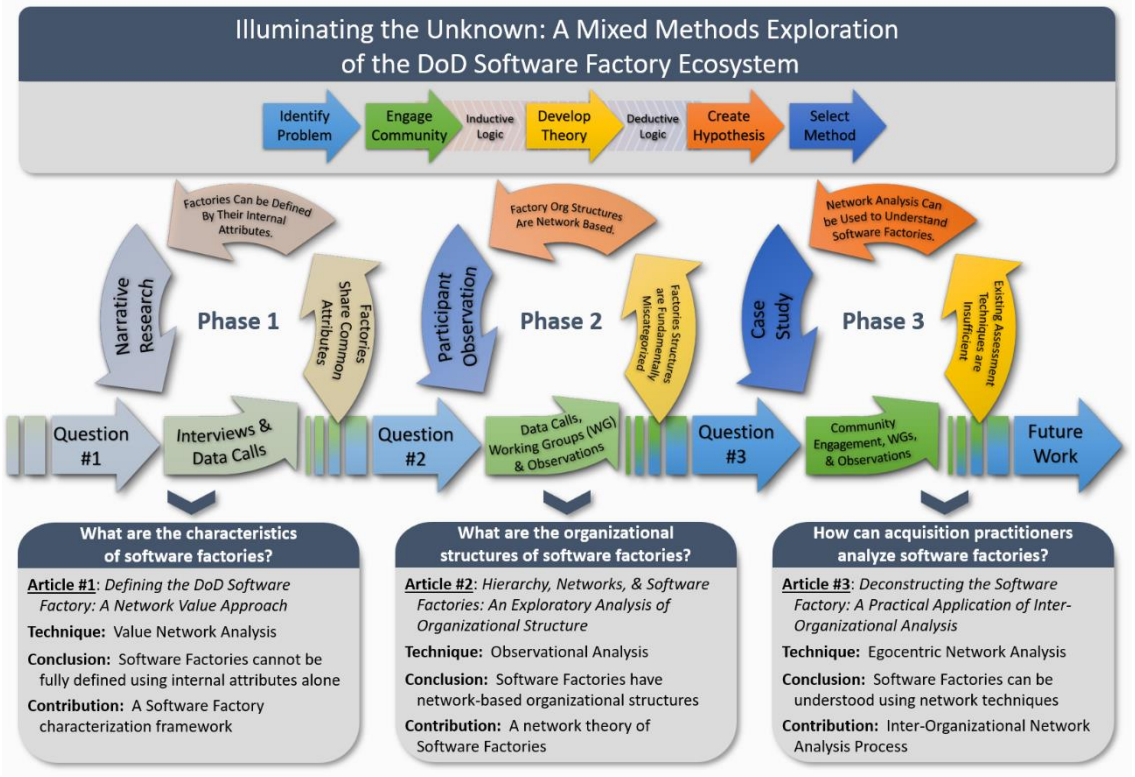

Figure 1: Thesis Methodology Overview. This thesis employs a sequential research design following the mixed methods methodology (Tashakkori & Teddlie, 2009, pp. 32–33).

Assumptions/Limitations

The DoD's software factories are immature and fragmented, and most are not formally established programs. This limits access to consistent data sources since traditional programmatic metrics are aggregated and collected at the program level. Basic organizational definitions and data that accurately depicts the characteristics of software factories does not currently exist. Understanding this constraint and acknowledging that quantitative data is largely unavailable, a qualitative approach to data collection is appropriate. This will inherently limit the type of analysis that can be conducted.

Organizations are constantly evolving and forming new relationships as they grow in size and scope. Since a focus of this research is understanding the nuanced behaviors of the software factories, some of the more specific results presented within the manuscripts will be temporal. For example, some of the case-history examples contained in Phase 3 must be considered as snapshots in time because they contain specific recommendations tailored towards the immediate needs of the organization. Additionally, while this research does make use of a variety of data sources and assessment methodologies, it is unable to capture specific observations of each software factory due to the size of the existing ecosystem and resource constraints. Instead, it relies on methodological triangulation to support the credibility, quality, and transferability of the inferences made within this thesis.

This research assumes that while the nuanced and specific individual characteristics, structures, and behaviors of different software factories will undoubtedly differ, the general structures and characteristics will be consistent amongst other similar software factory organizations. Additionally, because this research prioritizes community engagement and observation as well as the public presentation (formally and informally) of qualitatively generated inferences to the community being studied, its credibility is supported through its use of persistent observation and prolonged community engagement.

Since this thesis uses a qualitative approach to develop theory and interact with the community of interest, the results and points of view presented are also influenced by the researcher's values (Graff, 2013). Additionally, since social constructs like power and organizational dynamics are explored primarily through observations and interviews,

the resultant conclusions and theories presented within must be considered in the context of the community of interest and its value system. The mixed methods methodology acknowledges the existence of these values structures in the researcher and the community of study and thus the validity and credibility of this thesis is best considered in context with the software factory community. For this reason, community engagement and discussion are continually emphasized throughout the report.

Implications or Expected Contributions

This report will begin to systematically demystify the organizational phenomenon that is the software factory through the application of proven research practices. First, it will identify and capture the characteristics of software factories to expand the acquisition community's understanding of individual organizations and provide an academic basis for future research. Second, it will then attempt to define the software factories by observing the community and capturing the exchanges that occur between the organizations with which the software factories most commonly interact. By assessing the interactions of the organizations within their greater environments instead of analyzing internal organizational behaviors, this thesis can more objectively identify the commonalities in the structures of these organizations and as our understanding of the current state of software acquisitions matures, make inferences on the characteristics of the ecosystem as a whole. Finally, this thesis will conclude by providing acquisition practitioners with the ability to assess their own non-traditional organizations. The approach and perspectives used within this thesis will mark it as the first to define the DoD software factories beyond traditional cost-schedule-performance measures and instead define the software factories based on their organizational behaviors and

attributes whose interactions and exchanges comprise the wider department-wide ecosystem.

II. Defining the Department of Defense (DoD) Software Factory: A Network Value Approach

Chapter Overview

This chapter contains the first of three articles written for this thesis. The included manuscript, which was published in *Crosstalk: The Journal of Defense Software Engineering*, by Ryan, Reith, and Beach (2022) marks the initial, exploratory phase of this thesis's immersion into the greater software factory ecosystem. Through a series of interviews with senior leaders, software factory personnel, and a review of internal Department of the Air Force software factory documentation, the article begins with an assessment of the acquisition communities understanding of the software factories. After discovering that the software factory is not formally defined within the DoD, the article begins to frame the software factory by identifying the tangible and intangible value exchanges of Space CAMP, an existing DoD software factory, through the application of a value network analysis methodology. Combining the results from the interviews, document review, and value analysis the article concludes by presenting a characterization framework that outlines the most common characteristics of software factories.

Publication Details

This article was published in the July 2022 issue of *Crosstalk: The Journal of Defense Software Engineering.* It has been stylistically modified from its original published format to align with the prescribed format of this thesis.

Abstract

Over the past five years, the Department of Defense (DoD) has placed renewed emphasis on improving the DoD Acquisition System's ability to develop and manage its increasingly complex arsenal of software-centric materiel solutions. This sharpened focus on software development has led to the widespread implementation of both legislative and organizational changes that have directly impacted how Department of the Air Force (DAF) programs develop software capabilities. With the evolving acquisitions landscape, organizations have been exploring alternative business strategies that take advantage of modern development practices in order to improve organic development capabilities and more rapidly deliver secure and reliable code to end-users. A new business model, the "Software Factory", has become increasingly popular and has proven itself as a key capability enabler for many organizations. While the intent and tactical direction of the software factories are generally well understood by their parent units, exactly how these organizations fit within the larger acquisition's lifecycle and the manner by which they deliver long-term value is less clear. This article seeks to shed light on the underpinnings of the DAF's software factory ecosystem by first modeling the relationships between a single software factory and its stakeholder organizations in order to identify how value is unlocked, and then proposing a set of generalizable criteria with which to define and evaluate the greater software factory ecosystem.

Introduction

Software development within the DAF has historically lagged behind industry. In 2018, Congress directed a formal assessment of the DoD's software acquisitions policies and practices (McQuade et al., 2019). The results of this study recommended key areas

of improvement that were proposed to bring the DoD in line with industry software development standards. Since the report was published, many high-level programmatic recommendations have been implemented at both the services and congressional levels (Lord, 2020; OSD, 2020). Paralleling the acquisition pathway and funding strategic efforts, individual units have been quick to capitalize on the heightened interest and a new type of organization, the "Software Factory", has developed within the DoD software ecosystem. While strategic efforts have been in lock-step with the study's recommendations, the changes do not necessarily align with these new units' strategies. An apparent disconnect has developed as formally documented processes and policies fail to accurately reflect the actual methods, structures, and strategies that are being employed at the operational level.

To understand the disconnect between perception and reality, one may observe the DoD's own use and definition of the term "Software Factory". A cursory investigation of the Air Force Chief Software Office's (AF CSO) website seems to indicate that Software Factories are a type of organization (Assistant Secretary of Acquisition & Air Force Chief Software Office, 2021). This initial perception may appear at first glance to be correct, however, upon deeper inspection, it is clear that the listed programs are incredibly dissimilar—varying drastically in complexity, maturity, and mission set. The AF CSO's Software Factory listing is, in reality, simply a collection of organizations that have self-declared themselves as software factories and were subsequently endorsed by the AF CSO. While this disconnect would not normally be an issue, the use of the term "Software Factory" has become widespread within the DoD's

collective lexicon and is now being used as a cornerstone reference in strategic force modernization guidance (Hicks, 2022).

The DoD's pragmatic usage of the term is dichotomous to the units' that have chosen to carry it as a label. While the DoD often uses Software Factories to describe an organizational construct, units instead use it in a less clearly defined manner. At the unit level, "Software Factory" has no easily communicated definition. Instead, it is used as a signal to the broader defense community that the organization carrying its name does more than simply develop software. The Software Factory label, in this sense, relays that the unit views itself as part of a greater movement within the DoD, not as simply an organization that delivers software-based materiel solutions.

Central to the issue is that the use of the Software Factory label has become so widespread that it no longer accurately describes any one type of organization, yet the DoD still treats it as such. "Software Factories" now applies to an entire software ecosystem within the DoD that is largely composed of organically funded, self-directed units functioning as application developers, trainers, developmental services producers, hosting service providers, and even fee-for-service consultants. If the DoD is going to effectively manage, direct, and fund these organizations, it needs to stop thinking of its Software Factories as a singular, easily-defined entity, and instead treat what it has as a composition of demand driven organizations that have organically developed into a complex ecosystem.

This article attempts to inform strategic decision making by exploring and identifying the underpinning characteristics of one of the DAF's software factories through a preliminary data collection effort and through the application of a network

analysis technique. It is our hope that the characteristics identified within this analysis can help establish the foundations of a larger framework or organizational model that can be applied at scale to improve the understanding of the DoD's organic software development capabilities and to inform future direction.

The Software Factory Ecosystem

Software ecosystems, from a practical aspect, have been well studied within academia and defining them is thus not the focus of this article. It is important, however, to briefly review the formal definition as its composition provides insight into the reasoning behind our proposed network-based analysis approach. This study, like many other recent bodies of research, utilizes the definition of software ecosystems as defined by Jansen, Finkelstein, and Brinkemper (2009) and reiterated in Jansen, Brinkemper, and Cusumano's book (2013) on Software Ecosystems as:

"…a set of actors functioning as a unit and interacting with a shared market for software and services, together with relationships among them. These relationships are frequently underpinned by a common technological platform or market and operate through the exchange of information, resources, and artifacts."

The DoD has, over the past two years, placed heavy emphasis on the technical components within this definition, focusing strategic efforts on attempting to define and develop common platforms through the implementation of efforts such as the DevSecOps fundamentals guide (Department of Defense, 2021) and the stand-up of Platform One. These efforts are not surprising as the technical underpinnings of platform definition are

fairly straightforward to capture and communicate; at least for specific applications and architectures, which is evidenced by the AF CSO's endorsement of a common technical framework that outlines specific development requirements based on a containerized, Kubernetes managed solution (Chaillan, 2019). As units have matured, they often struggle to describe themselves beyond their technical commonalities which hampers external messaging.

On the other hand, the DoD has spent less time communicating how it intends to address business aspects of the definition which focus on the economics of software ecosystems. While a business-based definition may initially seem to be of questionable applicability as the DoD is neither a traditional business nor a market, our observations suggest that the existing organization of units and the manner in which they produce value closely resembles that of entrepreneurial business networks. When one steps back and views the DoD Software Factories through this lens, using an organizational and interaction-based focus, it becomes clear that understanding how these organizations interact and exchange value is critical to understanding and capturing the efficiency and effectiveness of the larger software ecosystem.

Analysis Methodology

Network Value Analysis

In order to characterize the larger ecosystem, we must first begin to identify characteristics inherent to the DAFs software factory organizations. This study begins this analysis by applying a network analysis technique to analyze the strategic relationships of an existing Air Force Software Factory, Space CAMP. Network analysis is an area of study and analysis that focuses on depicting relationships between actors in

order to analyze the structures that emerge from the recurrence of these relations (Chiesi, 2015). Network analysis has seen widespread application with uses in geosciences, social and behavioral sciences, economics, and many other disciplines (Chiesi, 2015; Curtin, 2017; Kronenfeld, 2004) including the DoD (Enos & Nilchiani, 2018). Our proposed analysis of the DoD's software factory ecosystem looks to apply network analysis based on the underlying assumptions that unique characteristics will become more evident and can be better captured when broken down into key components.

The area of network analysis as it relates to software ecosystems encompasses a wide span of topic areas, with differing intents and focuses. For example, a product-centric approach of software ecosystem modeling includes Boucharas et al.'s (2009) work which captured direct exchanges of products and services for resources surrounding a real estate software product. Additional types of application include the fine-grained modeling of internal software ecosystems (Boldi, 2020) and user/contributor centric ecosystem modeling (Guércio et al., 2018).

Given the immaturity of our target ecosystem, the diversity of software solutions and architectures, and the variety of organizations and customers within the DoD's software ecosystem, we determined that a value-based assessment would provide a unique and new way of characterizing capability development for strategic decision-makers. Value-based network analysis, which is rooted primarily in business theory, Software Design and Architecture focuses on identifying areas of human-centric value generation within business ecosystems (Allee, 2009).

We have chosen to analyze our ecosystem using the V2 model notation (Vorraber et al., 2011) which was built off of work by Biem and Caswell (Biem & Caswell, 2008)

and then successfully applied to an open-source software project by Vorraber et al. (Vorraber et al., 2019) yielding a detailed relationship model of the key stakeholders involved in the development of a free, open-source software project. While Vorraber et al.'s study focused on individual actors within the software ecosystem, our study abstracts the actors to an organizational level in order to map value from a macro-view. This approach allows us to garner insight into organizational relationships and characteristics while still allowing for scaling, which has been a challenge with some other types of software ecosystem modeling approaches that attempted to communicate with both technical and non-technical audiences (Boucharas et al., 2009).

Data Collection

Data collection efforts for this study were undertaken with two main objectives. First, we wanted to begin establishing some common characteristics that appeared to be prevalent within the software factory ecosystem. Second, we needed to determine which sources of information were most relevant and appropriate to guide our more in-depth network analysis of a single DoD software organization. In order to meet these two goals, our data collection strategy began with a broad general data collection effort and narrowed in on specific organizations of interest to better understand the overall relevancy and accuracy of different sources as well as establish a baseline understanding of the overall software factory ecosystem.

We began our assessment with a review of the public-facing websites of the Software Factory organizations as listed on the DoD CSO's software.mil website. This listing contains 19 organizations, 14 of which have short descriptions outlining their missions, and 6 of them maintain public-facing websites with relevant information. The

corresponding websites varied in maturity, level of information, and relevance with a broad range of communications ranging from information sharing to customer seeking strategies. While the information contained on these public facing websites was useful to develop a base level understanding of each organization, they do not convey the detailed inter-organizational information that is required for us to perform an analysis.

In order to collect more tailored and relevant information, we expanded our efforts and reached out directly to individual organizations with varying degrees of success. Given the organizations are military units, identifying up-to-date contact information from publicly available information proved challenging. In the end, a request for internal program documentation was sent via email to 12 organizations; from these, 4 responded and were willing to share additional program documentation. Documentation ranged from highly detailed programmatic handbooks to basic executive slide-show overviews. In total, 138 pages of documentation were collected and reviewed for relevancy.

To further understand the characteristics of these organizations and develop a foundational understanding from which to base our case study analysis, a series of interviews were conducted with SMEs from the most mature and established software factory organizations. Interviews focused on developing an understanding of how the respective organizations operate, their inter-organizational relationships, and their experiences operating within the DoD's software development space.

In summation, our overall data collection efforts generated a large amount of information from a wide variety of primary and secondary sources. This broad initial approach supported both of this pilot study's main data collection objectives—establish a

baseline understanding of the software factory ecosystem and identify the most relevant data sources for future analysis. Finally, the large amount of data collected allowed us to perform an initial analysis on an individual pilot organization in order to test the feasibility of our proposed network-based approach.

Discussion and Results

The network model in Figure 3 was achieved by first performing an extensive review of both public-facing and internal developer documentation for the target organization. After forming a baseline understanding of organizational relationships, outputs, and exchanges, an interview was conducted with an acquisitions SME who was familiar with the overall mission and relationships of the organization. Questions were developed to focus the conversation on customer interactions and were provided to the interviewee prior to the interview. The primary questions used to guide the discussion are as follows:

1. *Can you tell me about how your organization interacts with operational units? Other program offices?*
2. *Who do you consider your customers? Sponsors? Partners?*
3. *Can you walk me through the program lifecycle of an application from initial request to handoff?*
4. *Does your organization have a formal strategy? Can you tell me about it?*

After completing the interview, the analysis was conducted, the results of the analysis were presented to the interviewee, and feedback was solicited to refine the model. The results of the network analysis, in Figure 3, identified the primary organizations and their value exchanges. In addition to developing a proposed list of

organization characteristics, we also identified a few unique areas of interest that appear to be inherent to these types of organizations.

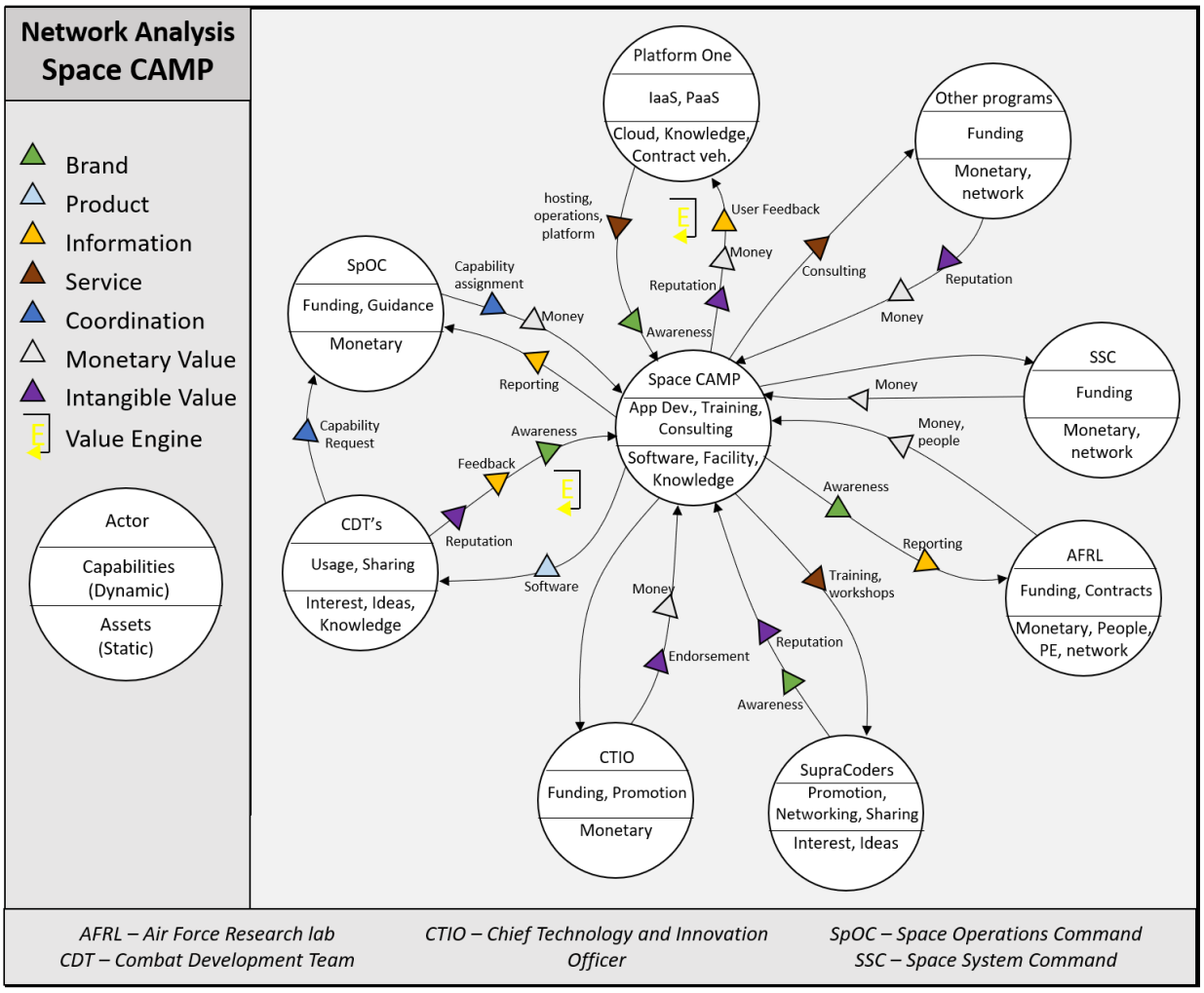

Figure 2:Space Camp value network analysis.

Value Diversity and Generation

A surface-level analysis of software factories such as Space CAMP may lead many to focus only on the primary tangible output, namely software applications for end-users. While software development and delivery is the primary mission of Space CAMP (*Space CAMP*, 2021a), it is but one source of value that the organization creates within the ecosystem. In addition to tangible services offered by the organization such as

consulting and training, non-tangible value by way of brand awareness and reputation exchange is also noted. Space CAMP has hosted over 500 visitors since its 2018 inception (Space CAMP, 2021b), and search engine results for "Space CAMP Air Force" generate multiple magazine articles, websites, and videos. While it is difficult to quantify non-tangible value associated with recruiting, inspiring, and informing the organization, the exchange of monetary capital for awareness and association is clearly visible within the network graph.

Fiscal Relationships

The initial network analysis provides interesting insight into the fiscal relationships and associated value exchanges of this particular organization. While one would initially expect a DoD program to follow a formal top-down directed fiscal hierarchy, our network graph clearly shows that Space CAMP does not adhere to this model. Instead, it maintains multiple fiscal relationships outside of its parent organization—in this case, AFRL/RV— instead of relying on a singular primary source of capital. This particular organizational model differs significantly from the traditional federal acquisitions lifecycle model which would designate funds based on a five-year planning cycle. Since Space CAMP is not a formally funded program via the Future Years Defense Program (FYDP), it's not bound to traditional cycles making it more susceptible to EOY funding chaos and its wide variety of fiscal relationships reflect that reality.

Value Engines

Two primary value engines were identified in this analysis. Value engines are "reinforcing loops of value creation and exchange between actors" (Vorraber et al.,

2011). The two most notable exchanges of value were between Platform One and Space CAMP and the combat development teams (CDTs) and Space CAMP. These two exchanges, as opposed to the other exchanges, directly impacted the value generation of the partner organization. Furthermore, by way of information and services exchange, they generated value in excess of what was initially exchanged by each organization. While quantifying the value generation of the overall ecosystem was not a goal of this study, it could provide a framework for future analysis.

Future Work and Conclusion

The analysis of Space CAMP's value network map helps shed light on the complex inter-organizational structures that have organically formed within the Software Factory ecosystem. A significant amount of the DAF's newest Software Factories do not necessarily follow the existing acquisitions pathways, including the newly created software pathway, that was established through congressional legislation. Instead, Space CAMP and many other Software Factories currently exist and operate as what appear to be informal programs, delivering software capability early in what could be loosely defined as the pre-acquisition pathway, having yet to achieve the official MDA designation that one would normally expect prior to active capability delivery. Since this organizational space falls outside of traditional acquisitions processes, it is not well understood nor formally captured.

This disconnect causes significant challenges as capturing and quantifying the value produced by these organizations becomes exceedingly difficult as funding profiles, capability delivery, and inter-organizational dependencies are not well tracked at a strategic level. It is our belief that if the DAF is going to successfully leverage its

software factories as a strategic asset it must first strive to define and understand the ecosystem that has developed organically.

Our research suggests that formally defining and categorizing the DAF's software factory ecosystem could be accomplished through the development and application of an organizational framework or model designed to capture specific characteristics that are inherent to these programs. Some potential defining characteristics and potential spectrums of measure that were revealed during this case study and our larger data review of other software factory organizations are proposed in Table 1. Focusing on the unique characteristics of these organizations will enable decision-makers to strategically assess the DAF's software development capabilities, identify organizational gaps or redundancies, and duplicate proven success models when establishing new software-centric units.

This study was undertaken in order to begin establishing characteristics of DAF software factories through the review and assessment of a singular organization. This was accomplished utilizing network value mapping techniques to identify key organizations involved in the support and value creation of Space CAMP. Our approach, which focused on identifying inter-organizational value exchanges extended beyond the traditional acquisition-centric approaches and provided new and interesting insights into the inner workings of the Department of the Air Force's newest type of software organization. Additionally, our review of software factory organizations revealed an initial set of characteristics that could be used to define the DoD's software factories more appropriately than existing dictionary definitions.

Finally, it is our belief that a wider organizational assessment that expands the scope of data collection and analysis beyond a single organization should be investigated as a means to characterize DAF software organizations and develop an organizational framework that can be used to guide software factories through a proven lifecycle and improve the DAF's understanding at a strategic level.

Table 1: Characteristics framework for defining the DoD software ecosystem. Proposed software factory framework characteristics for defining the larger DoD software ecosystem.

Characteristic	Measurement Spectrum
Funding Strategy	Fee-for-service – Centralized
Organizational Leadership	Personality driven – Institutionalized
Requirements Sourcing	Opportunistic – Deliberate
Government Manning Profile	Volunteer – Formalized
Technical Solution Scope	Specialized – Broad
Product Scope	Application – System of Systems
Contracting Strategy	Fragmented – Centralized
Service Offerings	None – Multiple
Value Engines	None – Multiple
External Signaling and Outreach	Low – High
Organizational Size	FY Executed Funds ($)

References

Allee, V. (2009). Value-creating networks: Organizational issues and challenges. *Learning Organization, 16*(6), 427–442. https://doi.org/10.1108/09696470910993918

Assistant Secretary of Acquisition, & Air Force Chief Software Office. (2021). *Software Factories*. https://software.af.mil/software-factories/

Biem, A., & Caswell, N. (2008). A Value Network Model for Strategic Analysis. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*. https://doi.org/10.1109/HICSS.2008.43

Boldi, P. (2020). Fine-Grained Network Analysis for Modern Software Ecosystems. *ACM Trans. Internet Technol*, *21*(1). https://doi.org/10.1145/3418209

Boucharas, V., Jansen, S., & Brinkkemper, S. (2009). *Formalizing Software Ecosystem Modeling*. http://softwareecosystems.com

Chaillan, N. (2019). *DoD Enterprise DevSecOps Reference Design*.

Chiesi, A. M. (2015). Network Analysis. *International Encyclopedia of the Social & Behavioral Sciences: Second Edition*, 518–523. https://doi.org/10.1016/B978-0-08-097086-8.73055-8

Curtin, K. M. (2017). Network Analysis. *Comprehensive Geographic Information Systems*, *3*, 153–161. https://doi.org/10.1016/B978-0-12-409548-9.09599-3

Department of Defense. (2021). *DevSecOps Fundamentals Playbook*. https://software.af.mil/wp-content/uploads/2021/05/DoD-Enterprise-DevSecOps-2.0-Playbook.pdf

Enos, J. R., & Nilchiani, R. (2018). *Using Social Network Analysis to Identify Systems of Systems in a Network of Systems*. https://doi.org/10.1109/SYSOSE.2018.8428791

Guércio, H., Ströele, V., David, M. N., Braga, R., & Campos, F. (2018). Complex Network Analysis in a Software Ecosystem: Studying the Eclipse Community. In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*. https://doi.org/10.1109/CSCWD.2018.8465170

Hicks, K. (2022). *Department of Defense Software Modernization Strategy*.

Jansen, S., Brinkkemper, S., & Cusumano, M. (2013). *Software Ecosystems*. Edward Elgar Publishing. https://doi.org/10.4337/9781781955635

Jansen, S., Finkelstein, A., & Brinkkemper, S. (2009). *A sense of community: A research agenda for software ecosystems.* https://doi.org/10.1109/ICSE-COMPANION.2009.5070978

Kronenfeld, D. B. (2004). Cognitive Research Methods. *Encyclopedia of Social Measurement*, 361–374. https://doi.org/10.1016/B0-12-369398-5/00325-X

Lord, E. M. (2020). *DOD INSTRUCTION 5000.87 OPERATION OF THE SOFTWARE ACQUISITION PATHWAY*. https://www.esd.whs.mil/DD/.

McQuade, M. J., Murray, R. M., Louie, G., Medin, M., Pahlka, J., & Stephens, T. (2019). *Software Is Never Done: Refactoring the Acquisition Code for Competitive Advantage*.

OSD. (2020). *Budget Activity (BA) "BA-08": Software and Digital Technology Pilot Program*. https://discover.dtic.mil/section-809-panel/,

Space CAMP. (2021a). https://spacecamp.il2.dso.mil/#/home

Space CAMP. (2021b). *Space CAMP Survival Guide 2.0*.

Vorraber, W., Mueller, M., Voessner, S., & Slany, W. (2019). Analyzing and managing complex software ecosystems: A framework to understand value in information systems. *IEEE Software*, *36*(3), 55–60. https://doi.org/10.1109/MS.2018.290100810

Vorraber, W., Voessner, S., Ssner, S. V., & Vössner, S. (2011). *Modeling endogenous motivation and exogenous influences in value networks of information service systems*, https://doi.org/10.4156/jcit.vol6.issue8.43

Summary

This chapter contained the first of three manuscripts written for this thesis. While this introductory article did not formally define the software factory, it did establish an initial foundation upon which to base future analysis by identifying common organizational characteristics and capturing those results in a software factory characterization framework. In addition to its publication in a peer reviewed journal, the article was circulated throughout the software factory community and it was presented to the 4th DoD Software Factory Working Group and the DoD DevSecOps Community of Practice, reaching a large joint audience. This community interaction and the circulation of the characterization framework developed within this article spurred an ongoing conversation within the community around what exactly defined the software factory. As interest in the topic continued to expand, additional software factories began providing feedback and internal organizational data which expanded the data set available for this thesis.

While the published article showcased the value network analysis technique as the primary method of data collection, it was not the only approach employed during the data collection process. Prior to conducting the network analysis, a narrative research process was employed to generate an initial data set and to establish a foundational understanding of the software factory ecosystem. This approach, which focused on interviews with senior leaders and software factory founders, was necessary because data on the organizational characteristics of the software factories did not exist. The data set created during the narrative process was continually revisited and reanalyzed throughout the mixed methods process.

III. Hierarchy, Networks, and Software Factories: An Exploratory Analysis of Organizational Structure

Chapter Overview

This chapter contains the second of three manuscripts written for this thesis. Building upon the first article, the included *Hierarchy, Networks and Software Factories: An Exploratory Analysis of Organizational Structure,* written by Ryan, Reith, and Beach, seeks to define the software factory at the structural level by highlighting its fundamental economic and social characteristics. The article begins with a review of past influential Department of Defense software strategies and reports in an attempt to identify parallels between past DoD software programs and the software factories[2]. After identifying what appeared to be a historical trend of strategic management failures within DoD software acquisitions, the article theorizes that an unspecified disconnect must exist between DoD management policies and the natural organizational structures of software organizations. The article builds upon this theory by comparing the characteristics of software factories to the characteristics of the three primary economic forms of organization: hierarchy, networks, and markets. The article concludes by identifying and characterizing the software factory as an existing and formally defined economic form of organization— the network.

[2] A detailed historical assessment of the most influential DoD software reports conducted by the Defense Innovation Board, Defense Science Board, and Government Accountability Office spanning 1970-2022 is included as an additional supplement in Appendix A.

Publication Details

This article was written for submission and consideration to *Joint Forces Quarterly.*

Abstract

The Acquisitions community has historically struggled to understand, measure, and manage non-traditional programs. The Department of Defense's (DoD) newest batch of software organizations, commonly known as *Software Factories*, highlights these shortfalls and illustrates the need for alternative strategic management approaches tailored toward unfamiliar economic architectures. This paper explores the characteristics of software factories and classifies them as a formally defined economic structure uncommon within the DoD called the *network*. We then expand upon our assessment by exploring the social and economic behaviors of software factories by analyzing them in relation to established socio-economic concepts. Finally, we close by providing a set of recommendations, supported by academic literature, that can be used to tailor the strategic development of the software factory ecosystem while simultaneously leveraging the unique characteristics of networks.

Two-sentence summary

The Software Factory is a new type of software development organization that is gaining popularity within the Department of Defense. This article explores their structures in relation to their social and economic environments and suggests they are best classified as networks.

Keywords

Inter-Organizational, Relationships, Management, Non-Traditional, Socio-economics

Introduction

In 2018 an unusual organizational phenomenon emerged within the Department of Defense (DoD) acquisitions community. Advocates of agile software development began self-organizing under the term *Software Factory* to distinguish their approach from traditional software development organizations. Built upon a set of common foundational principles, the factories organically emerged and established themselves around common goals, strategies, and technical vision. Acting outside of formally established programmatic frameworks like those defined within the Defense Acquisitions System (DAS), these organizations exhibited non-traditional characteristics unfamiliar to many acquisition practitioners. As individual entities, the factories appeared to act without formal structure yet when viewed holistically they shared a unified goal – to develop better software. The software factories represented a cultural shift within the military and they promised to deliver a faster, more effective method of software acquisition through the application of alternative approaches, techniques, and structures.

From a historical perspective, many of the innovative principles and practices espoused by the software factories such as iterative development practices, end-user requirements flexibility, and a focus on software quality are not new. In fact, strategic guidance illustrates the opposite. In a review of the most influential DoD software strategy documents from the past 50 years, we determined that when one looks past the constantly evolving methodologies, technologies, and architectures the roots of the DoD's software struggle have remained fundamentally unchanged. The Acquisition

System is inflexible, the DoD fails to implement development best practices, and software metrics are non-existent or ineffective. The Defense Innovation Board (DIB) study *Defense Software* (2000), which also reviewed and aggregated strategic studies arrived at a similar conclusion stating:

"Most all of the recommendations remain valid today and many could significantly and positively impact DoD software development capability. The DoD's failure to implement these recommendations is most disturbing and is perhaps the most relevant finding of the Task Force. Clearly, there are inhibitors within the DoD to adopting the recommended changes "(Hansen & Nesbit, 2000, p. ES1).

The harsh language used in the 2000 DIB study is not unique. The historical reports and strategic documents listed in Figure 4 illustrate through their consistency that there exists an unidentified problem with the mechanisms that the DoD uses to strategically manage its software enterprise. While many of the reports acknowledge the existence of this problem, they simply attribute it to poor execution instead of seeking to understand *why* the problem exists. In doing so, they ignore the heart of the issue choosing instead to reiterate known organizational challenges and repeat the same general set of recommendations. We believe that it is time to stop re-explaining the *known* and instead assess the *unknown* by exploring the underlying economic and social environments that influence the DoD's software programs. To quote the late Colin Powell, "There are no secrets to success. It is the result of preparation, hard work, and learning from failure." If the DoD is going to be bold and dominate the modern battlespace by enabling Joint All Domain Command and Control (JADC2), it must first learn

from its past failures. To do so we must be prepared to challenge our existing preconceptions and established organizational norms to understand why our past software acquisition efforts have continuously fallen short. The current software factory movement provides us with this opportunity.

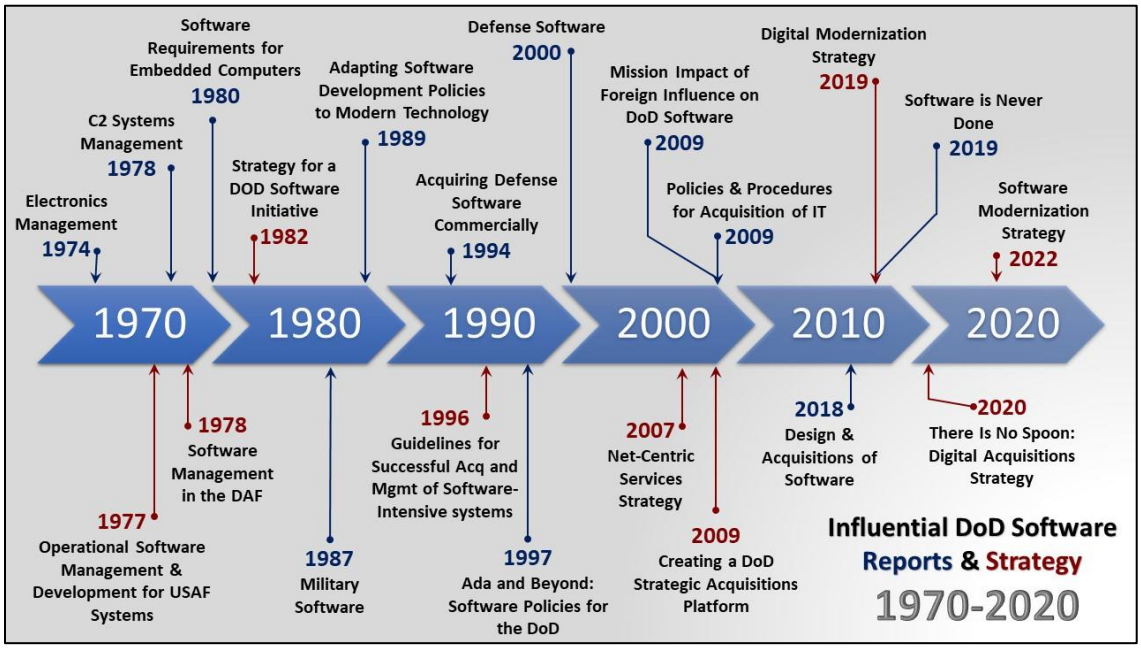

Figure 3: Influential DoD Software Reports & Strategy.

Research into software factories has primarily focused on the technical and financial underpinnings of factory operations; however, recent academic works have begun to explore their organizational structures. In a recent *Crosstalk* article, Ryan, Reith, and Beach (2022) published an analysis of the software factory ecosystem which described a framework outlining observed software factory characteristics. While the writers did not provide a written definition of the software factory, they did identify and catalog previously undocumented organizational characteristics. In reviewing their framework, we discovered that the characteristics they captured appeared to align closely

with the features of a previously defined and well-understood economic form called the *network*. Building upon this observation, we hypothesize that the organizational structures of software factories are best represented, understood, and managed as networks.

This article begins by introducing the network economic form by discussing fundamental socio-economic concepts that may be unfamiliar to the reader. We then characterize the software factory as a network through a series of contextual examples designed to illustrate how software factories fit within the hierarchy-network-market continuum. Additionally, we also expand upon the social and economic behaviors of software factories that are not typically found within traditional military hierarchies. Finally, we close with a series of recommendations that we believe can be used to guide the strategic development of the software factory ecosystem by drawing on established network management principles.

The Network, Resources, and Exchange

The network as an economic form was first defined in 1990 when Walter Powell published a seminal piece of research titled *Neither Market nor Hierarchy: Network Forms of Organization.* Powell's work introduced the network as a formal form of organization that existed between the previously well-defined and explored economic forms of market and hierarchy. He argued that the network form had unique attributes that could be identified by observing the patterns of both economic and social behaviors between interdependent organizations. Powell's work drew upon exchange-based theories to describe the network as well as socio-economic theories of social structure. His consideration of social concepts such as network embeddedness (Granovetter, 1985),

Table 2: Powell's stylized comparison of forms of economic organization (1990).

	Forms		
Key Features	*Market*	*Hierarchy*	*Network*
Normative Basis	Contract-Property Rights	Employment Relationship	Complementary Strengths
Means of Communication	Prices	Routines	Relational
Methods of Conflict Resolution	Haggling – Resort to courts for enforcement	Administrative fiat – Supervision	Norm of reciprocity – Reputational concerns
Degree of Flexibility	High	Low	Medium
Amount of Commitment Among the Parties	Low	Medium to High	Medium to High
Tone or Climate	Precision and/or Suspicion	Formal, bureaucratic	Open-ended, mutual benefits
Actor Preferences or Choices	Independent	Dependent	Interdependent
Mixing of Forms	Repeat Transactions (Geertz, 1978)	Informal organization (Dalton, 1957)	Status Hierarchies
	Contracts as hierarchical documents (Stinchcombe, 1985)	Market-like features: profit centers, transfer pricing (Eccles, 1985)	Multiple Partners Formal Rules

trust (Arrow, 1975), and reciprocity (Gouldner, 1960; Keohane, 1986) and their impacts on information exchange allowed Powell to identify the key features of economic networks. Powell chose to define this new economic form by creating a framework, illustrated in Table 2, that highlighted the network's key features in comparison to markets and hierarchies.

Before progressing further, we must first take a moment to address the definition of the term *network.* This article primarily discusses networks in a strategic context, referring to the formal economic form of organization. In practical application, however, the term network is ubiquitous and its meaning is often derived contextually which makes it difficult to define. This presents a challenge as unstated ambiguity often leads to confusion. For example, when we speak of managing networks to what do we refer? From where are we deriving our perspective? Are we speaking broadly, referring simply to groupings of similar entities, or are we attempting to communicate something more nuanced such as the existence of an implicit social structure? Perhaps we are inferring the existence of specific characteristics or maybe we are simply implying casual connection. To what do we speak when we reference inter-organizational networks?

Software Factory networks? We cannot answer these questions without first defining and understanding the pre-existing or implied context.

To maintain a strategic management focus, we supplement Powell's definition of the network form by drawing upon network research as it relates to inter-organizational relationships. Within this context, we define networks as the continuous exchange of resources between interdependent organizations acting within shared social and economic environments. This definition draws upon the resource dependence theory of inter-organizational relationships (Franke, 2017), which is built upon the underlying assumption that an organization's success in achieving its stated goals depends on its ability to access limited *resources* in relation to its greater environment. By extension this introduces the concept of the *exchange* – the mechanism organizations use to gain access to these limited resources (Levine & White, 1961). By applying the basic concepts of resources and exchange, we can employ a simple but useful medium by which to observe inter-organizational behaviors. Put simply, by observing the exchange of resources between organizations we can build our understanding of networks.

Research into networks is incredibly broad with existing literature having expanded upon Powell's original work. Network studies now encompass many diverse fields including public policy and management, innovation, and organizational dynamics. Given this diverse and ever-growing body of knowledge on networks, we cannot realistically provide an all-encompassing overview of the concept. Instead, we will take a tailored approach by focusing on the exchange behaviors of software factories to illustrate their organizational alignment as networks. The following section discusses the

software factories as networks by analyzing patterns of behavior that we observed within their social and economic environments.

Software Factories as Networks

In order to utilize Powell's definition of the network as a mechanism to help describe the software factory, we must first review and loosely characterize existing Department of Defense organizational and programmatic structures. Traditionally, the DoD and the military services follow the hierarchical form of economic organization. Communication and directives flow downward via formal channels, passing through various levels of the organization before arriving at subordinate units. Communication, or information exchange, is routine and the overall climate is bureaucratic and formal. The traditional

Table 3: Software factory characteristics and the network form. The observed characteristics of software factories align with the features of economic networks. Because software factories often lack formal programmatic designations, they must rely heavily on social relationships and trust-based networks to secure the resources (i.e. money & personnel) that are necessary to accomplish organizational objectives. Software Factories strategically build their social capital and reputations by leveraging non-traditional mechanisms such as LinkedIn, websites, media outreach, and trade shows to drive organizational growth. Traditional programs do not exhibit this social reliance because they receive resources through the formally established hierarchical mechanisms defined within the Defense Acquisitions System.

	Form	*Software Factories*
Key Features	*Network*	***Observed Characteristics***
Normative Basis	Complementary Strengths	• **Broad Partner Networks (Industry, Academia, Inter-service)** • **Volunteer Manning Profiles** • **Personality Driven Leadership**
Means of Communication	Relational	• **Reliance on Social Networking** • **Customer Seeking**
Methods of Conflict Resolution	Norm of reciprocity – Reputational concerns	• **Active Social Community (LinkedIn, Websites, Working Groups)** • **High External Signaling (Websites, magazines, PA pieces)**
Degree of Flexibility	Medium	• **Opportunistic Requirements Sourcing** • **Product & Service Diversity** • **Flexible Funding Models** • **Organizational Autonomy**
Amount of Commitment Among the Parties	Medium to High	• **Active employee involvement** • **Tight-knit community** • **Inter-organizational problem solving**
Tone or Climate	Open-ended, mutual benefits	• **Non-Traditional Workplaces** • **Casual Dress Codes** • **"Start-up" cultures**
Actor Preferences or Choices	Interdependent	• **Customer Dependent Funding Models** • **Industry Partner Interdependence** • **Shared acquisitions functions**
Mixing of Forms	Status Hierarchies	
	Multiple Partners Formal Rules	

organizational chart which defines administrative and operational authorities is a familiar visualization of formal hierarchy at work. The Defense Acquisitions System (DAS) exhibits similar hierarchical patterns. Control of programmatic organizations is maintained through the application of formal financial controls, standardized directives, and careful structural definition of programs outlined within DoD 5000.01, *The Defense Acquisitions System.* These traditional frameworks break down when used to describe software factories. Acknowledging this shortfall, we turn towards the network.

Utilizing Powell's framework to expand Ryan et al.'s work, we engaged with and observed the software factory community through working groups, communities of practice, and senior leader interactions. Additionally, we also embedded ourselves within a Department of the Air Force software factory for three months in order to observe its social and economic exchanges. Our observations of software factory behaviors closely aligned with Ryan et al.'s findings as well as Powell's network characteristics supporting our proposed hypothesis. Our observed characteristics of software factories are presented alongside Powell's key features of networks in Table 3. In order to further illustrate the networked nature of software factories and provide the reader with a foundational understanding of network fundamentals, the following sections expand upon the social and economic environments of software factories using basic explanatory examples.

Social Environment

Software factories are socially active organizations. Our observations revealed that software factories are active participants within their social environments with many software factories maintaining public-facing websites, active social media profiles, and well-defined public messaging strategies. Additionally, within the software factory social

sphere, there are various working groups, conferences, and trade shows through which these organizations regularly participate. While these social behaviors may at first appear as common to the casual observer since most are accustomed to the commercial websites and social media presence of companies, they are in actuality less common internally to the DoD. This raises the question: why are software factories so active and why are they expending resources on what could be categorized as advertising and outreach? The answer to this question is easily explained when one views software factories through the network lens.

Network participation requires organizations to actively seek and create social connections for a variety of reasons. First and foremost, organizations must interact with their environment to gain access to resources. The exchange of resources, in a network context, includes the exchange of information as well as tangible goods, services, and money. This horizontal exchange of information between organizations is emphasized within innovation-heavy industries such as software (Castilla et al., 2000). Like all types of exchange, information exchange places a financial burden on organizations. With software factories, this cost is visible through their social activities: personnel travel, website costs, and media messaging. This phenomenon, the cost of an exchange or transaction, is a fundamental concept within economics and is referred to as *transaction cost* (Williamson, 1979). Since networks have a heavy social component, organizations must expend resources to participate.

In addition to transaction costs, organizations participating in networks must also expend resources to build and maintain social capital, and as an extension, reputation. Since networks lack the formal control mechanisms found within hierarchies, they rely

on organizational reputation and social standing to signal trustworthiness. *Trust* is a critical component of successful networks because it facilitates cooperation between organizations and it enables self-governance. This is especially true in emergent networks like software factories (Provan & Lemaire, 2012). Due to the interdependent nature of networks where organizational success is often shared, organizations with poor reputations or social standing may be denied access to critical resources or even wholesale rejected from the network ultimately leading to their failure. Additionally, organizations with high social capital have access to greater power, influence, and resources within the network (Provan & Lemaire, 2012). Thus, to ensure continued success within the network, participating organizations expend resources to build and maintain their social standing and reputation. Thus, we have explained the active social outreach.

The social component of networks extends well beyond the fundamental concepts that we have provided in this section. Inter-organizational networks can be incredibly complex and difficult to measure, much less quantify. As a practical example, during our studies, we observed a new software factory less than a year old with a rapidly growing network comprised of 53 individual organizations. To illustrate this interconnectivity, a

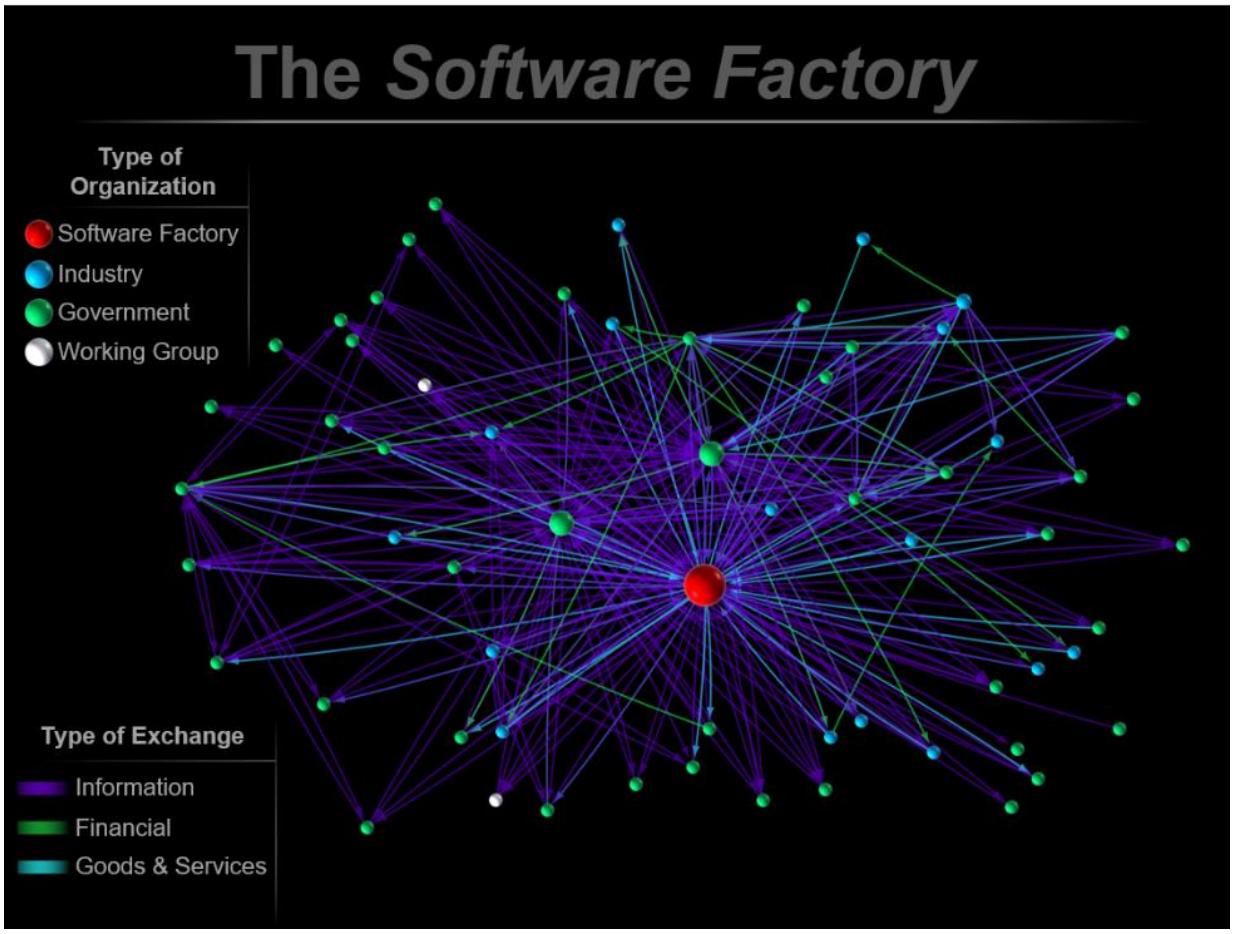

Figure 4: A software factory network. Software factory networks are comprised of interdependent organizations exchanging resources within shared economic and social environments. We can explore the interorganizational behaviors of software factories by observing these exchanges.

visualization of this organization's network is provided in Figure 5. Extending this complexity to the greater software factory ecosystem, we can begin to envision the intricate web of social connectivity that underpins the software factory movement. This

network complexity is intertwined with and supports the economic activities of software factories.

Economic Environment

While the social environments of software factories can be understood by observing the exchange of information, the economic environment is best studied by focusing on the tangible exchanges of goods, services, and finances. Ryan et al (2022) observed the basic financial exchanges of a single software factory noting that it appeared to have an unusual customer-driven funding model. Our observations of software factories yielded similar results. While not all software factories rely on customer-driven funding instead receiving direct appropriations, a significant portion leverage network-driven funding strategies. In a traditional programmatic model, this lack of a formalized hierarchical funding mechanism may appear as an inherent weakness; however, from a network perspective the distributed funding model illustrates how networks can efficiently self-distribute limited resources amongst competing organizations based on customer requirements.

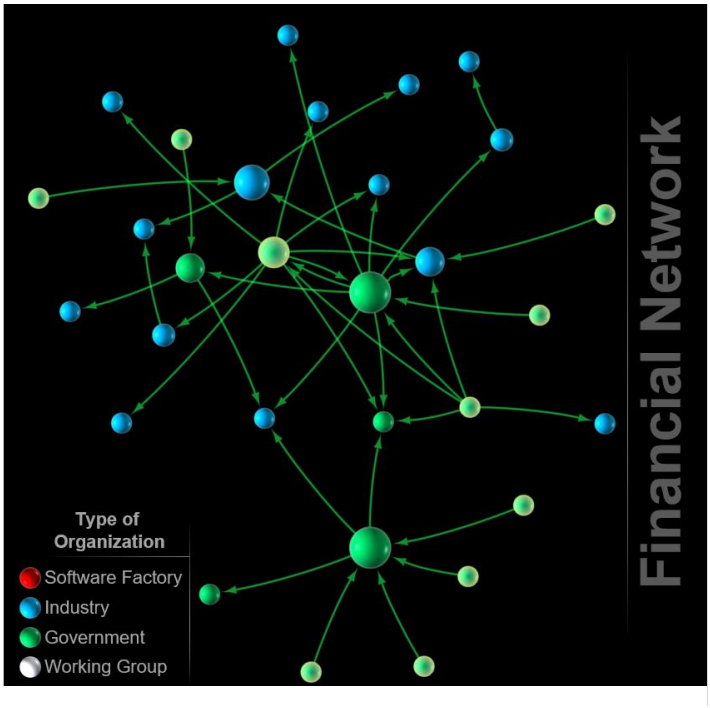

Figure 5: A software factory financial network. Software factories often employ distributed, customer-based funding models. Customer organizations that are actively funding the sample software factory are highlighted in yellow.

To further explore network-based funding mechanisms and illustrate their effectiveness, we reference the financial network of a sample software factory--provided in Figure 6. This sample network captures the economic exchanges of a single software factory revealing its highly distributed nature with funding sources extending across multiple MAJCOMs. This model reveals a break from the hierarchical MAJCOM or congressionally controlled funding model in favor of a domain-centric, customer-controlled model typically found within commercial markets. Under this model, software factories are able to extend their economic networks beyond formally established organizational boundaries and end customers are able to leverage the network aspects of trust and reputation to shape software factory behaviors without relying on hierarchical authority. Within these networks, the software factory that fails to satisfy its customers will likely develop a poor reputation eventually leading to network exclusion and its ultimate failure. This effect illustrates the power of the customer-driven funding model as well as demonstrates how the concepts of interdependence and trust are intertwined within both the social and economic environments of networks.

The social and economic environments of the software factories provide them with a higher degree of autonomy not typically seen within traditional DoD acquisition programs. Theoretically, a distributed funding model prevents a singular organization from exerting authoritative control over an individual software factory organization via financial leverage. Additionally, while a software factory may have to provide status reports to a leadership organization, in actuality authoritative control is limited. The widespread adaptation, or lack thereof, of Platform One illustrates this effect in practice. While the adoption of Platform One has not yet been mandated, Platform One has been

designated as an official enterprise service provider for the DoD. Even so, many organizations have still chosen to employ alternative technical solutions. This effect illustrates the network form at work – resources are distributed based on network dynamics. In networks the individualized needs of organizations often drive decisions; not the desires of external parties.

Discussion & Recommendations

Our observations and analysis demonstrate that the Software Factories' organizational characteristics closely align with the network form of economic structure. While we believe that this conclusion is well supported using empirical evidence, we would be remiss to continue without addressing the perspective from which it was derived. There is little doubt that classifying organizations using the Hierarchy-Network-Market continuum can provide interesting insights into organizational behaviors but it can also prove challenging. This is because the perspective with which one chooses to approach organizational structure can have an unintended impact on the outcome. Take for example a hypothetical assessment of the software factories that begins with the underlying expectation that military organizations must be hierarchical.

In this hypothetical scenario, an assessment may begin by seeking out organizational characteristics known to support the hierarchical model. For example, among other documents and policies, the assessment could limit its focus to existing organizational charts; this limited approach could lead to the forgone conclusion that structures are hierarchical. Building upon this conclusion, to the assessment could then be extended to include external behaviors. Given the historically contractual nature of government-industry exchanges, this may lead to the conclusion that the external

behaviors of software factories can be characterized exclusively as market driven interactions. While this assessment may align with existing preconceptions, it fails to thoroughly explain other behaviors. For example, it does not explain the interdependent relationships that have evolved between the software factories, their partner organizations, their government customers, and the greater software factory ecosystem. Additionally, it fails to acknowledge the frequency and intensity of the social exchanges that exist between the software factories and their industry partners. In this way perspective can be limiting.

The exact number of software factory organizations within the DoD is currently unpublished; however, the DAF Chief Software Office website does identify 12 software factories, 3 software engineering groups, and 3 enterprise service providers under the existing Air Force Ecosystem. Given this diversity, it is expected that our observations and the characteristics of networks cannot apply to all software factory organizations. Additionally, it is likely that some behaviors have remained unobserved. Some may choose to point towards this lack of totality as a characterization failure; however, it is not our intent to perfectly define the software factory space nor do we believe it is possible. Instead, we simply seek to provide a much-needed foundational base by which to establish a shared context. Defining and understanding organizational structure drives us toward that goal.

Our findings have significant implications for the Department of Defense's current efforts to formalize the software factories into a cohesive ecosystem. Drawing on the precedence discussed within our introduction, it is clear that the DoD has historically struggled to manage, control, and grow its software-centric programs. We believe that

these past shortfalls could be partially attributed to a misalignment of management strategies in relation to the organizational structures of software programs. Network-based organizations, like those found within the software factory ecosystem, are fundamentally different from traditionally defined hierarchical structures and as such, they require different strategic management approaches that may be unfamiliar to career practitioners. To continue forward under the premise that familiar hierarchical practices and techniques will succeed with software factories when they have previously failed is inviting a repetition of past failures. To prevent this, we have provided the following strategic recommendations which are informed by existing literature on network management. We believe that by employing these principles the DoD can improve strategic oversight of the greater software factory ecosystem while still allowing software factories to maintain their network-centric attributes.

Recommendation #1

The Department of Defense should focus efforts on enabling network dynamics, not controlling individual organizational outcomes.

Traditional hierarchical management approaches focus on controlling risk by internalizing and standardizing business functions to achieve authoritative control. Standardization of communication channels and consolidated decision-making via formal bureaucratic structures are often preferred over distributed systems which encourage organizational autonomy. While the monitoring and control of organizational behaviors (Provan & Kenis, 2008) is an effective means by which to manage traditional programs, these hierarchical control mechanisms are less effective when applied to networks that rely on autonomy and relational information exchange to function. Additionally, from a

strategic perspective, the successful management of individual organizations does not directly translate to the successful management of the whole network. Thus, when managing networks, senior leaders must employ strategic approaches specifically designed toward enabling network dynamics instead of controlling individual organizational outcomes. For example, policies that focus on improving network participation and self-regulation through the reduction of transaction costs and increased inter-organizational trust should be considered.

Recommendation #2

Senior Acquisitions Executives should consider Software Factories' existing structures before granting formal programmatic designation.

Many software factories have a flagship product or focus within a specific niche of the software development community. To fund product development within this space, they rely on multiple funding streams directly from interested customer organizations. This funding strategy, while effective, introduces instability into the software factories' operational profiles. Research into the inter-organizational behavior of organizations suggests that software factories will naturally seek stable access to resources (Benson, 1975), which in many cases takes the form of formal funding through the Future Years Defense Program (FYDP). While this approach will provide financial stability to the software factory, it also directly impacts the organizations participating within the software factories networks by shifting network dynamics. Introducing external funding reduces the software factories' reliance on its customer organizations and it fundamentally shifts power within the network. Since the software factories will no longer be interdependent on their customer organizations for funding, the inter-

organizational relationships will be impacted. The decision to grant centralized funding in effect reduces the financial leverage currently held by customer organizations – removing their primary mechanisms of influence. Thus, before granting centralized funding, network effects on other interdependent organizations outside of the requesting software factory should be considered.

Recommendation #3

The Department of Defense should avoid mandating specific architectures, platforms, or technical solutions to software factories.

Historically the Department of Defense has turned to standardization and consolidation as mechanisms by which to shape software acquisitions. While standardization can reduce variance across programs, it works against network dynamics and can have unintended impacts on innovation. The DoD's choice to standardize around Ada in the '80s and '90s provides a good example of this effect. In mandating Ada, the DoD drifted from the commercial sector which continued to grow and innovate (Boehm et al., 1997). As a response to this, DoD acquisitions policy shifted dramatically in the 90s and 00's as decision-makers realized the importance of the commercial sector for software acquisitions. To prevent a repetition of past mistakes and replicate commercial practices, senior leaders should allow software factory networks to drive technical decision-making.

In addition to the historical lessons learned from standardization, mandates also have impacts on inter-organizational dynamics. By mandating a specific technical solution provider, the DoD reduces the autonomy and flexibility of its software factories. The power within the greater network shifts from the software factories to the designated

platform provider. This shift in dynamics is exaggerated as software factories lose control over internal financial assets that must now be utilized to implement the mandated solution. Instead, the DoD should employ an ecosystem approach that emphasizes the shared value of the platform relationship. Like networks, one of the defining aspects of ecosystems is that of autonomy (Jacobides et al., 2018); organizations *choose* to participate within the ecosystem due to mutual benefit. If organizations are not currently choosing to adopt specific technical solutions, then the value proposition between platform providers and network organizations should be revisited.

Recommendation #4

The Department of Defense should consider facilitating the software development services of software factories through the use of a revolving fund structure.

Our final recommendation, and most specific, is to consider establishing a new DoD-level revolving fund for software services, which would provide the Department of Defense with an alternative means of standardization, control, and accountability over software factory network exchanges without relying on direct mandates. While there are ongoing efforts to provide alternative acquisitions and funding mechanisms within the DAS via the new Software Pathway (OUSD, 2020) and the BA-08 pilot (OSD, 2020), these efforts are tailored towards traditional programs and do not necessarily fit the network-centric structure favored by the software factories. Additionally, our observations reveal that the software factories perform a service function within their greater networks and do not act as traditional programs. While software factories primarily develop software, this is often provided as a service. They also provide other services such as consulting, design, training, and maintenance. Under 10 U.S. Code §,

2208 – the Department of Defense can require working-capital funds to "provide working capital for such industrial-type activities, and such commercial type activities that provide common services within or among departments".[3]

Research into innovation centers such as Silicon Valley has revealed that financial institutions play a critical role in transferring financial resources and enabling organizations to innovate (Castilla et al., 2000; Cohen & Fields, 1999). Our research shows that this role is largely absent in the existing factory ecosystem. Currently, organizations must rely on third parties to act as financial brokers. This fractured and informal banking function prevents the ecosystem from self-regulating because potential customers have no way of assessing the financial reputation of the software factories. If the software factory ecosystem is going to succeed, there must be a means to loudly communicate poor performance to the greater ecosystem. Software factories *must be allowed to fail* and a revolving fund could provide the necessary economic-based control mechanisms.

In establishing a separate fund for software services, the DoD can provide a flexible funding mechanism for the software factories while also establishing accounting, reporting, and auditing controls over software factory operations. The revolving fund would also allow software activities to be tracked at the customer level instead of by traditional cost per application, SLOC, or other historically ineffective metrics. This shift would provide essential customer-centric data related to development costs, performance, and customer satisfaction further enabling self-regulation. Additionally, the fund would

[3] GSA's 18F unit successfully utilizes its acquisitions services fund to provide developmental services similar to those provided by software factories. Given these similarities, a model for effective revolving fund implementation of technology and software already exists (*18F: Digital Service Delivery* , 2022).

provide improved accountability for the appropriation of funds. Finally, a revolving fund would provide access to tangible economic-based metrics for software factory performance. Those in the software development space will often state that the only metric that matters is user satisfaction; a revolving fund would allow *money* to be used as a measure of satisfaction. Customer metrics such as lifetime value (LTV), repeat customer rate, and customer churn rate, as well as financial metrics such as overhead costs, cash runway, and fully burdened developer costs, could all be tracked through the use of a revolving fund structure.

Conclusion

We have gone to great lengths in an attempt to educate the reader on the foundational aspects of inter-organizational networks while simultaneously defining software factories as networks. First, we provided a brief overview of the basic forms of economic organization: hierarchy, markets, and networks. We then explored the organizational structures of the non-traditional organizations formally known as software factories and subsequently defined them as networks. Next, we performed an exploratory analysis that assessed both the social and economic behaviors of these organizations while also engaging the reader with a discussion on key network concepts. Finally, we provided a set of broad strategic management recommendations that we believe will help align the software factory ecosystem with existing departmental intent while also avoiding past management failures.

While there is a berth of existing literature on inter-organizational networks and their management in both the public and private sectors, their applicability within Department of Defense acquisitions programs is largely unexplored. Additionally, there

exists a shortfall in existing programmatic guidance as it relates to managing and understanding programs in relation to their greater economic and social networks. Strategically, defense acquisition guidance focuses on the internal structures and behaviors of programs. If program managers are going to successfully manage network-centric organizations then they must first have a means by which to assess and understand the strategic "big-picture" aspects of their programs. Expanding the existing body of defense acquisition knowledge to create and provide these assessment mechanisms as well as educate practitioners on network structures should be a focus. Finally, metrics specific to network-based organizations within the DoD should be explored as an additional means to enable network self-governance and measure progress.

In February 2022, the Office of the Secretary of Defense formally acknowledged the strategic importance of the growing software factory ecosystem in its DoD Software Modernization Strategy (Hicks, 2022). This strategy outlined three primary goals designed to shape the future of defense software development and enable JADC2: accelerate the DoD enterprise cloud environment, establish the department-wide software factory ecosystem, and transform processes to enable resilience and speed. In order to accomplish this overarching modernization objective, the DoD must figure out how to transform its organic software factories into a cohesive movement. The first and most critical step in achieving that objective is understanding what currently exists at a foundational level. This article is the first to address and drive the Department of Defense toward that objective.

References

Arrow, K. (1975). The Limits of Organization. In *The ANNALS of the American Academy of Political and Social Science* (Issue 1). Norton.

Benson, J. K. (1975). The Interorganizational Network as a Political Economy. *Administrative Science Quarterly*, *20*(2), 229. https://doi.org/10.2307/2391696

Boehm, B., Baker, T., Embry, W., Fox, J., Hilfinger, P., Holden, M., Moss, E., Royce, W., Scherlis, W., Taft, T., Vaughn, R., & Wasserman, A. (1997). Ada and Beyond: Software Policies for the Department of Defense. In *Ada and Beyond*. National Academies Press. https://doi.org/10.17226/5463

Castilla, E. J., Hwang, H., Granovetter, E., & Granovetter, M. (2000). 11 Social Networks in Silicon Valley. In *The Silicon Valley Edge* (pp. 218–247). Stanford University Press. https://doi.org/10.1515/9781503619180-017

Cohen, S. S., & Fields, G. (1999). Social Capital and Capital Gains in Silicon Valley. *California Management Review*, *41*(2), 108–130.

Franke, U. (2017). Inter-Organizational Relations: Five Theoretical Approaches. In *Oxford Research Encyclopedia of International Studies*. Oxford University Press. https://doi.org/10.1093/acrefore/9780190846626.013.99

Gouldner, A. W. (1960). The norm of reciprocity: A preliminary statement. *American Sociological Review*, *25*, 161–178.

Granovetter, M. (1985). Economic Action and Social Structure: The Problem of Embeddedness. *Source: American Journal of Sociology*, *91*(3), 481–510. https://doi.org/10.2307/2780199

Hansen, M., & Nesbit, R. (2000). *Defense Science Board Task Force on Defense Software*. https://dsb.cto.mil/reports/2000s/ADA385923.pdf

Hicks, K. (2022). *Department of Defense Software Modernization Strategy*.

Jacobides, M. G., Cennamo, C., & Gawer, A. (2018). Towards a theory of ecosystems. *Strategic Management Journal*, *39*(8), 2255–2276. https://doi.org/10.1002/smj.2904

Keohane, R. O. (1986). Reciprocity in international relations. *International Organization*, *40*(1), 1–27. https://doi.org/10.1017/S0020818300004458

Levine, S., & White, P. E. (1961). Exchange as a Conceptual Framework for the Study of Interorganizational Relationships. *Administrative Science Quarterly*, *5*(4), 583. https://doi.org/10.2307/2390622

OSD. (2020). *Budget Activity (BA) "BA-08": Software and Digital Technology Pilot Program*. https://discover.dtic.mil/section-809-panel/,

OUSD. (2020). *DOD INSTRUCTION 5000.87 OPERATION OF THE SOFTWARE ACQUISITION PATHWAY*. https://www.esd.whs.mil/DD/.

Powell, W., Staw, B., & Cummings, L. (1990). Neither Market Nor Hierarchy: Network Forms of Organization. *Research in Organizational Behavior*, *12*, 295–336.

Provan, K. G., & Kenis, P. (2008). Modes of network governance: Structure, management, and effectiveness. *Journal of Public Administration Research and Theory*, *18*(2), 229–252. https://doi.org/10.1093/jopart/mum015

Provan, K. G., & Lemaire, R. (2012). Core Concepts and Key Ideas for Understanding Public Sector Organizational Networks: Using Research to Inform Scholarship and Practice. *Public Administration Review*. https://doi.org/10.2307/41687977

Ryan, Z. O., Reith, M. G., & Beach, P. M. (2022). Defining the DoD Software Factory: A Network value Approach. *Crosstalk*. https://community.apan.org/wg/crosstalk/m/documents/420788

Williamson, O. E. (1979). Transaction-Cost Economics: The Governance of Contractual Relations. *The Journal of Law and Economics*, *22*(2), 233–261. https://doi.org/10.1086/466942

Summary

This was the second of three manuscripts written for this thesis and it was the first within departmental and academic literature to research and formally identify the underlying structures of the software factories. This article, and the article presented in Chapter Four, where briefed to the Department of Defense CIO and a joint audience at the 5th Software Factory Working group where they garnered the attention of the Department of the Air Force Chief Software Officer and the Department of Defense Strategic Software Factory Working Group who recognized its innovative approach to organizational definition. In addition to directly informing strategic decision makers within the Department of Defense, this article also highlighted a gap in defense acquisitions guidance and training as it related to the management and definition of non-traditional software organizations. The output of this article, the network theory of software factories, provided the community with an alternative perspective into how the organizational structures of the software factories affected strategic management decisions within the DoD.

This article built upon the data analyzed during the first phase of research by collecting additional data through a participant observational study of a sample software factory and the software factory community. The use of a participant study was necessary due to the closed nature of the software factory community. In addition to enabling data access, this research approach provided two primary benefits. First, it allowed for the observation of the relationships of a sample software factory and it also provided insight into the social and economic behaviors of other organizations acting within the ecosystem. Second, the data collected during the observational period

established a baseline of behavior prior to moving into the third phase of research, which utilizes participant interviews to gather data. This mixed methods approach helped control for interviewee bias and it increased the credibility of both data sets.

IV. Deconstructing the Software Factory: A Practical Application of Inter-Organizational Network Analysis

Chapter Overview

This chapter contains the final manuscript written for this thesis, *Deconstructing the Software Factory: A Practical Application of Inter-Organizational Network Analysis,* written by Ryan, Reith, and Koschnick. The first two articles primarily focused on defining the software factories through their characteristics and through their structural composition. In doing so, they highlighted how understanding organizational structure is an important first step in the strategic management of non-traditional software acquisitions programs. The third article in this collection expands upon the first two phases of research by developing, describing, and demonstrating a practical inter-organizational network analysis (ION-A) methodology that acquisitions practitioners can employ to develop a big picture view of non-traditional network-based software organizations like software factories.

Publication Details

This article has been submitted for publication and is currently in peer review for *Defense Acquisitions Research Journal*. It follows the case-history format prescribed in DARJ publication guidelines and it has been stylistically modified from its submitted format to align with the prescribed format of this thesis.

Abstract

Over the past five years, there has been an increase in the number of Department of Defense (DoD) software organizations that employ non-traditional organizational

structures. These organizations, commonly referred to as software factories, often follow structures found within commercial industries. This article demonstrates how network analysis techniques can be used to develop and visualize a strategic, big picture view of these non-traditional organizations. Drawing on methodologies employed by network researchers, we develop and present an inter-organizational analysis (ION-A) process that captures a program's social and economic structures. Following the case history approach, we then demonstrate the applicability of this approach by analyzing an emergent DoD software factory. Throughout the article, we discuss how network principles and accessible analysis techniques can be applied to the real-world challenges faced by modern, network-based DoD organizations. Finally, we conclude by presenting the results of our analysis and a guiding framework that the acquisition community can use to analyze non-traditional programs.

Two-sentence summary: This article shows how network analysis can be used to understand and visualize the inter-organizational networks of non-traditional Department of Defense (DoD) software organizations. It presents a network analysis technique that captures economic and social relationships through the observation of exchanges and applies it in a case history analysis of a DoD software factory.

Introduction

Program Managers (PMs) have at their disposal a variety of assessment and tracking tools designed to provide actionable insights into the cost, schedule, and performance of acquisition programs. Existing tools such as Earned Value Management, Work-Breakdown Structures, and Technical Performance Measures (Driessnack, 2017)

provide valuable insight into the internal characteristics of acquisitions organizations while also helping PMs monitor and control program dynamics. While these tools are effective at providing oversight into specific aspects of program performance, they do not provide direct insight into the social and economic interactions that shape the organization's overall structure. To understand this big picture, PMs must rely on alternative information sources like organizational charts, employee interviews, and established institutional knowledge.

Understanding the people, processes, and program perspectives that compose the strategic big picture of an acquisition program is central to the core responsibilities of the program manager and the overall success of the program; the DoD has created and published a myriad of tailored and accessible guidance available to PMs (DAU, 2022a, 2022b). While this guidance is applicable to the majority of defense programs, it does make the underlying assumption that programs will follow the traditional structures outlined within the Defense Acquisition System (DAS). When organizations fall outside of these bounds, (e.g., some non-traditional software organizations), this guidance is less applicable. From a systems viewpoint, abstractions used to represent organizational structures (such as organizational charts or architectural models) may not be wholly adequate when applied to non-traditional software organizations. This raises the question: if existing guidance is not sufficient for non-traditional programs, how can the program manager begin to understand their organization's structure and subsequently develop the strategic big-picture perspective? More simply stated:

How can program managers assess the inter-organizational structures of non-traditional software organizations?

In this article, we aim to address the challenge of understanding the structures of non-traditional programs by proposing and demonstrating an analysis methodology designed to capture and visualize inter-organizational networks. Specifically, we present a tailored methodology derived from a multiplex, egocentric network analysis technique[4] and demonstrate its utility by analyzing the social and economic structures of a non-traditional DoD software factory. We conclude with a discussion on the implications of our research, a list of strategic analysis questions, and the approach's applicability to other non-traditional defense programs.

Background

DoD software factories often adopt inter-organizational structures more commonly found in commercial industries. These structures influence the economic and social relationships between organizations differently than those defined by traditional acquisition program boundaries. As a result, non-traditional programs do not always align with the established hierarchical structures outlined within the DAS. This case history analyzes one such program, a non-traditional Department of the Air Force (DAF)

[4] *Multiplex, Egocentric Network Analysis:* A multiplex network displays multiple types of relationships or ties between organizations. Egocentric networks focus on a specific organization of interest, defined as the ego (Perry et al., 2018). This differs from a socio-centric analysis approach which seeks to derive insights into the whole network. Multiplexity is discussed in-depth in the analysis section of this article.

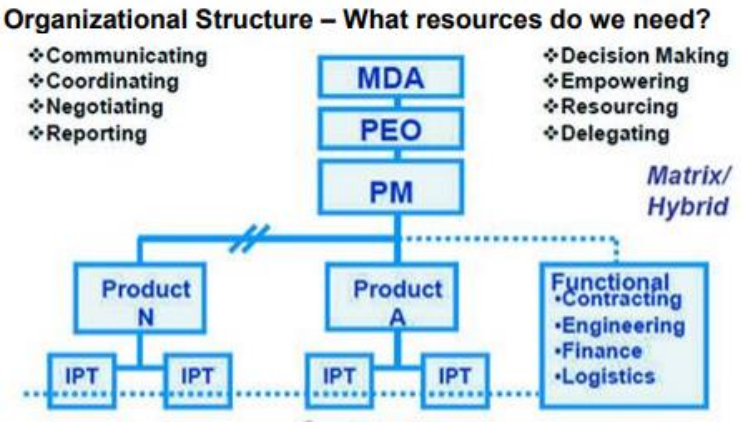

Figure 6: Hierarchical organizational structure. Departmental literature is primarily tailored toward acquisition programs that follow formally defined hierarchical structures. In this chart extracted from the DAU Program Manager's Toolbox (Driessnack, 2017), financial resourcing is implicitly tied to the PEO and MDA. Non-traditional programs, like software factories, can exhibit alternative structures that do not align with this framework. As a comparison, the PM of a traditional program may request additional funding from formal channels, while the PM of a software factory may seek funding directly from customers or partner organizations.

Software Factory, using a process called inter-organizational network analysis (ION-A). This background section introduces the concept of inter-organizational networks by explaining how they are related to familiar organizational models. It then describes their composition and explains how inter-organizational networks can be used to visualize the economic and social relationships of an organization. Finally, this section closes by introducing the ION-A process as a method to characterize these relationships.

Visual models of inter-organizational structures allow acquisition practitioners to easily draw on past experience and training to derive a baseline understanding of the behaviors and dependencies of a target organization. To illustrate this effect, we break down Figure 7, which depicts a common organizational structure composed of boxes and lines representing entities and their relationships. Because the structure of boxes and lines is easily recognizable and familiar to practitioners, the roles, motivations, and influences of the depicted entities can be assumed even if not explicitly stated on the

diagram. Additionally, basic economic and social interactions between entities can be inferred. Figure 7, in its simplicity, begins to frame the big picture of a program by invoking a familiar structure by which to understand economic and social relationships. However, when organizations do not follow these established patterns, the underlying assumptions regarding the economic and social relationships inferred from the hierarchical structure may no longer be valid. Therefore, additional steps should be taken to deliberately characterize the behaviors and patterns of exchange that were previously assumed or inferred.

The generic organizational structure depicted in Figure 7 can be characterized as an abstract representation of the inter-organizational relationships of a program. If this representation were expanded to represent all organizations that interact with the program, it would in effect begin to depict the program's network. By adding additional detail and explicitly defining the lines to represent relationships between the entities, we can formally define the resultant representation as the program's *inter-organizational network* (Brass et al., 2004). An inter-organizational network encompasses a collection of related organizations (called nodes), and

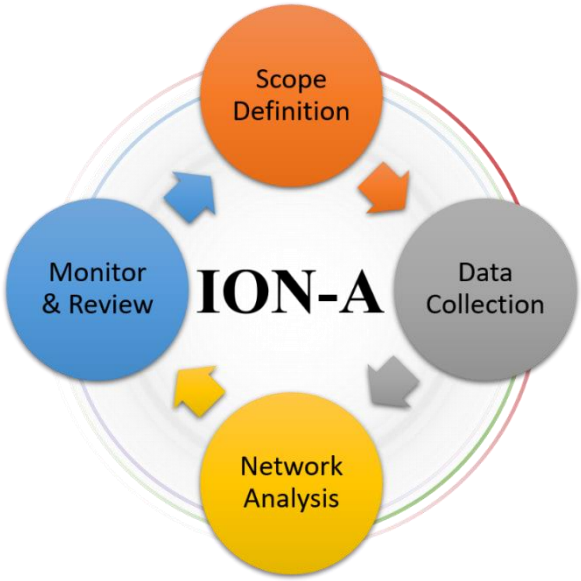

Figure 7: The Inter-Organizational Network Analysis (ION-A) process. The 4-step Inter-Organizational Network Analysis process can be used to develop a "big picture" view of an organization. ION-A focuses on understanding the social and economic relationships of a target program by visualizing exchanges of money, goods& services, and information.

their relationships (called ties, links, or edges). These relationships, which can be observed through exchanges[5], can be further subset by type. Economic relationships can be observed by identifying exchanges of money and goods & services. Social relationships can be observed by identifying exchanges of information. When multiple types of relationships are represented within an inter-organizational network, the network is defined as *multiplex*. When viewed holistically, this collection of nodes and ties provides a model of the organization's economic and social structures that can be analyzed.

Modeling and analyzing an organization's inter-organizational network can be accomplished by conducting an inter-organizational network analysis (ION-A). This 4-step process, depicted in Figure 8, outlines how practitioners can deliberately capture and visualize the inter-organizational networks of their programs; these networks can then be analyzed using established network methods and by employing a deliberate assessment approach designed to facilitate strategic critical thinking. The following sections further define the ION-A process and demonstrate its application in a case history analysis of a DAF software factory.

Situation

To demonstrate how the (ION-A) process can be utilized to better understand non-traditional software programs, we embedded ourselves within a DAF software factory that could not be easily characterized via the formally established structures

[5] The utilization of the exchange as a way to measure organizational relations was initially proposed as a method for studying inter-organizational behavior and relations in the early 60's by Levine & White (1961). Since its initial definition, the exchange has been broadly employed by researchers to measure both social and economic transactions between both individuals and organizations (Wasserman & Faust, 1995).

outlined within the DAS. In preparation for this case history, and to develop a contextual understanding of the organization's patterns of behavior as well as establish trust, we observed the daily stand-up meetings of the software factory leaders for three months. This approach mimics the experience of someone new to the organization and shows that practitioners do not need extensive experience within the program to conduct a meaningful analysis.

The software factory analyzed for this case history has existed for one year and it does not have any formally defined military or civilian manpower requirements. Instead, this organization is composed of participants from partnering organizations who have volunteered to manage and grow the software factory. This software factory exists to provide a specific niche of goods and services in the form of consulting services, design services, educational outreach, application development, and systems development and maintenance to external customers. Since the organization is in the early stages of growth it relies heavily on organizations within its local network to supplement core acquisitions functions such as contracting, finance, and legal services. This organization also maintains autonomy and control over its assets and it has been formally recognized by the AF CSO as a software factory. Programmatically, this software factory operates outside of existing acquisition pathways and it does not have the compulsory reporting requirements of a formal program. Collecting information on software factories was challenging because many candidates were uneasy about sharing organizational information. We addressed this concern by not attributing these results.

Analysis

This section describes in detail how to assess the social and economic relationships of the software factory using the inter-organizational analysis process (ION-A). Each step in the ION-A process is described and demonstrated using the software factory as an example. There are a wide variety of network analysis methodologies employed by academic researchers; however, many of these approaches can be overwhelming to those without a network background. Considering this, the ION-A process focuses on basic concepts that should be intuitive to many professionals and applicable to most programs. Finally, this section summarizes the specific insights gained into the software factory. We now begin the four-step ION-A process.

Step 1 – Scope Definition

Step 1 in the ION-A process is to define the scope of analysis. Begin by defining the nodes and ties of the inter-organizational network. Nodes will encompass all organizations that interact with the program and ties will represent relationships between those organizations. When defining nodes, the specific boundaries of individual military organizations will need to be considered based on the size and environmental context of the program of interest. To maintain a strategic focus, military organizations were aggregated and captured at the branch or squadron level in this analysis. Three types of ties representing both economic and social relationships are defined using the exchange variables: goods & services, finances, and information. A detailed definition of these variables is included in Table 4[6].

[6] The variables goods & services, financial, and information were selected based on the findings of recent research into the structures of software factories (Ryan et al., 2022) and research on multiplex economic networks (Maghssudipour et al., 2020).

Table 4: Network variables of exchange.

Exchange Variable	Definition	Relationship Type
Goods & Services	The transfer of goods & services from one organization to another.	Economic
Financial	The transfer of money from one organization to another.	Economic
Information	The active and deliberate transfer of meaningful information, knowledge, communication, or coordination where one or both parties benefit.	Social

Step 2 – Data Collection

Step 2 in the ION-A process is to collect and organize data. Data collection is the most challenging step of an ION-A but it can be broken down into two major efforts— the identification of nodes and the identification of ties. Node identification begins with the creation of a list containing all known organizations that the program has interacted with. This initial list may be built from a variety of sources including internal websites, existing documentation, or employee interviews. The list used to analyze the software factory was built by identifying the organizations named within the software factory's internal knowledge management system, Confluence. Each organization was labeled based on affiliation as government, industry, or working group before being organized in

a matrix format using excel. Figure 9 demonstrates how a matrix can be created, organized, and interpreted.

After creating a matrix, identify inter-organizational exchanges. It is unlikely that a consolidated data source containing a comprehensive listing of exchanges currently exists. With this in mind, we recommend data be collected through a series of semi-structured interviews or brainstorming sessions with experienced personnel using the matrix as an aid.

	Software Factory	Organization A	Organization B	Organization C	Organization D	Organization E	Orga
Software Factory				I	I, S	I, S	I, S
Organization A	S						
Organization B	S						
Organization C	I					I	
Organization D	I					I	
Organization E	I, S			I	I		I
Organization F	I, S					I	
Organization G	I					I, F	
Organization H	I						
Organization I	I						
Organization J	I						

Figure 8: An Example sociometric matrix. A sociometric matrix can be used to capture exchanges between organizations. Organizations identified in the roster are organized in the first column and duplicated in the first row. Exchanges between these organizations are then noted within each corresponding cell of the matrix. The matrix is interpreted by reading "column A sends (Money, information, goods & services) to Row 1. Example: The software factory sends information and services to organization D

In this case history, a series of video interviews were conducted with software factory leadership to identify inter-organizational exchanges. The matrix served as a guide and tool to document individual exchanges. Each intersection within the matrix was reviewed and interviewees were asked to provide information on exchanges between organizations. All exchanges were assessed for directionality. The matrix was completed through four interviews and responses were validated through a review of the video recordings. Inconsistent or unclear responses were addressed in subsequent interviews. The final matrix was comprised of 62 organizations and their exchanges.

Once data is collected it must be converted to a network format for analysis. There are several existing open-source network tools that may be used for this task. For this case history, the finalized matrix was processed using the open-source network graphing software *Gephi* (Bastian et al., 2009) and *Cytoscape* (Ideker, 2003). Step 2 is complete once the data has been initially processed using the network graphing software. The networks of the software factory are displayed in Figure 10. The composite network (10a) is created by combining the three variables of exchange: informational (10b), financial (10c), and goods & services (10d).

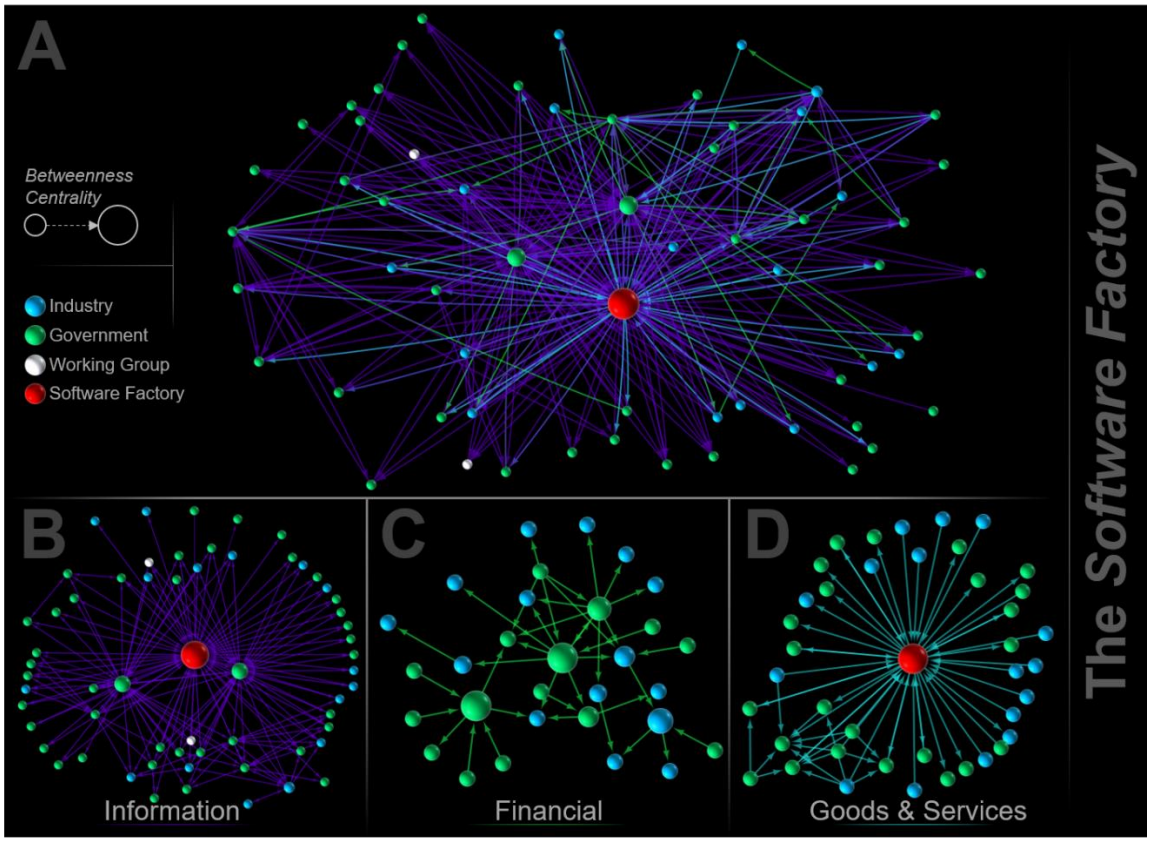

Figure 9: Composition of Software Factory networks. Four separate inter-organizational networks were generated from the exchange variables finances, goods & services, and information collected in step 2 of the ION-A. The software factory's composite network is comprised of government (69.3%), industry (25.8%), and working groups (3.2%) totaling 62 organizations. These networks provide a visualization of the economic and social structures of the program and they are analyzed in step 3 of the ION-A process.

Step 3 – Network Analysis

Step 3 in the ION-A process is to analyze the inter-organizational network of the software factory. In this stage, we focus on assessing four strategic areas: relationship strength, organizational influence, patterns of exchange, and communities of interest. A summarized assessment framework that includes these four areas of interest along with assessment questions and common network indicators is presented in the conclusion in Table 6. The remainder of this section addresses these areas individually. The first area of analysis is relationship strength.

Identify the weakest and strongest relationships within the organization's network.

The analysis begins at the composite network level (4a) by assessing the strength of relationships between nodes using the concept of multiplexity. The existence of multiple types of ties (i.e., social and economic) between entities is generally believed to indicate greater strength and durability of the relationship (Granovetter, 1973; Perry et al., 2018). Using this concept, a basic assessment of strength can be determined by counting the ties between nodes. More ties indicate a stronger relationship while fewer ties indicate a weaker relationship. Important relationships can be identified manually or network analysis software can be used to

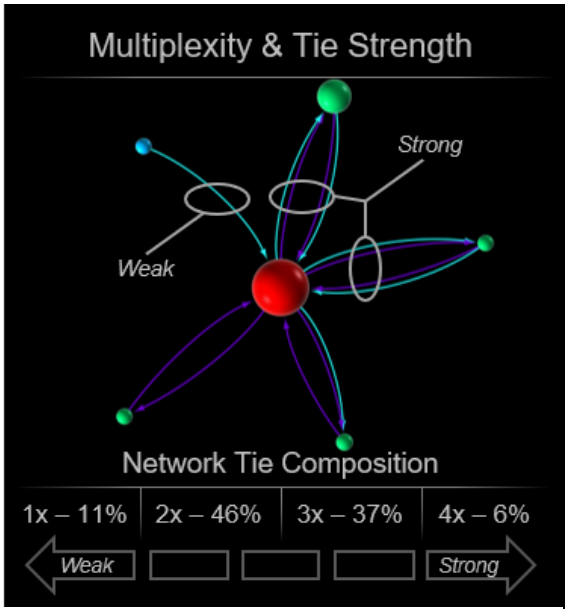

Figure 10: Tie Strength. Tie strength for the composite software factory network was assessed using a multiplex approach. Individual exchanges between nodes were counted and summed to provide a relative measure of exchange-relationship strength.

easily assess the entire network. Figure 11 illustrates how tie strength and durability can be analyzed via multiplexity.

The network of the software factory is primarily composed of ties constituting medium relative strength with only 6% of ties reaching the strongest threshold. These ties, which represent shared, reciprocal relationships exist primarily between the software factory and its partnering organizations. Notably, this tie pattern is also present between the sample software factory and a government platform provider where a two-way exchange of services occurs. The weakest ties within the software factory network are primarily informational. A unidirectional reporting relationship is an example of a weak tie.

Assessing relationship strength provides useful insight into common organizational interactions. Strong ties can be deliberately built by engaging in different types of exchange. Weak links can be identified and deliberately strengthened as the organization matures. Strategic social partnerships can be reinforced by engaging in economic exchange. For example, the software factory provides software services at zero cost to an organization that controls access to informational resources. This deliberate action strengthens the software factory's relationship with an influential organization. This leads to the next area of analysis; organizational influence.

Identify the most influential organizations within the network. Consider both the social and economic environments.

Most practitioners are familiar with the authoritative power structures that exist within a traditional hierarchy. However, in situations where these structures are less well-defined, an organization's positional power can often play a greater role in its ability to exert power and control over the program. Positional power structures can be

identified by analyzing centrally positioned organizations, critical resource flows, and exchange brokers within the program's inter-organizational network. By considering these factors in conjunction with the program's requirements, it is possible to identify influential organizations.

To identify centrally positioned organizations we assessed the composite inter-organizational network of the software factory for *betweenness centrality*- which is represented on the network models using node size. Betweenness centrality measures how often a node exists on the shortest path between nodes within the network in order to identify nodes that have greater influence due to their positional embeddedness (Freeman, 1977). The two primary partner organizations of the software factory had the highest centrality indicating their importance relative to other organizations in the network. When assessing the economic network for centrality, (displayed in Figure 12a), additional influential organizations can be identified.

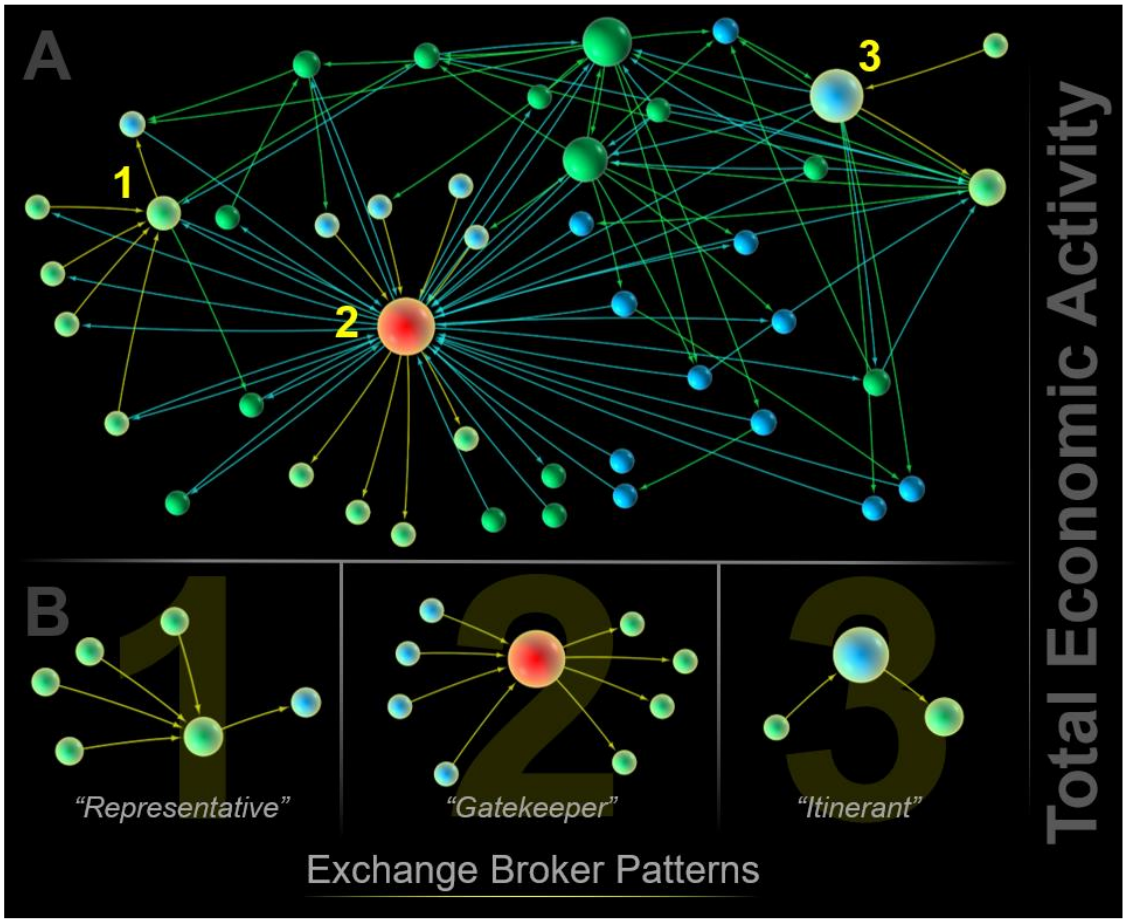

Figure 11: The software factory economic network. The economic network of the software factory is generated by combing the exchange variables finances and goods & services. In addition to identifying centrally positioned organizations, three broker roles were deemed notable within the Software Factory economic network: the representative, the gatekeeper, and the itinerant. Tracking these brokers is an essential component of identifying and controlling external program risks.

The economic network, which was generated from the finances and goods & services exchange variables, provides valuable insight into the flow of resources between organizations. One notable observation is that the flow of financial resources through the network is varied; the software factory is funded by customer organizations as opposed to a centralized source. In order to utilize these resources, the software factory relies on multiple contract vehicles owned by other organizations within its network. The organizations that manage and maintain these vehicles act as economic hubs. Because

they control the flow of resources from customers to the software factory, they have positional power. Another observation is the flow of goods & services is largely centered around and converges on the software factory with additional convergence occurring around the organizations acting as brokers- which are organizations that help coordinate exchange.

There are three primary broker roles within the software factory network highlighted in Figure 12b: the representative, the gatekeeper, and the itinerant. These roles are defined based on their affiliation and positioning within the network (Gould et al., 1989). The *representative* broker exists when a government organization coordinates a transaction from a government organization to an industry partner. One government organization is *representing* another. This occurs when customers fund a contractor through a third-party government organization. The *gatekeeper* role exists when a government organization purchases services from a third party via a contractor organization. The contractor exists as a *gatekeeper* between the third party and the purchaser. Gatekeeping also occurs when an industry partner passes services to a government organization which then passes services to a government customer. The software factory is primarily acting as a gatekeeper. The final brokerage role, the *itinerant*, occurs when an industry partner acts between two government organizations. All itinerant brokerage relationships within the software factory network are managed via formal contract vehicles. The influence of brokers must be considered in concert with their organizational affiliation and the flow of resources through the network.

In situations where traditional hierarchical structures are not clearly defined, identifying the organizations which hold the most influence over a program can be

challenging. However, ION-A can assist practitioners in this task by providing a visual representation of the program's inter-organizational relationships. By analyzing these resultant models, we can assess centrality, identify resource flows, and identify broker roles. This information can then be used to identify influential organizations. Next, we will explore how identifying patterns of exchange can inform the big-picture perspective of a program and assist in decision-making.

Identify common patterns of exchange.

At this point in the ION-A process, important organizations and their relationships within the program's network have been identified and considered with the assistance of network-based measures such as multiplexity and centrality. Conversely, identifying patterns of exchange requires a more nuanced analysis approach. Begin by focusing on the identification of the exchange patterns required to complete known inter-organizational transactions. Since the relative importance of specific transactions will be dependent on the specific program, consider the program's mission statement, its organizational objectives, and its core functions to assist in identification. We focused our analysis on the transactions that occur between the software factory and its customers due to their frequency and relative importance to the program's strategy.

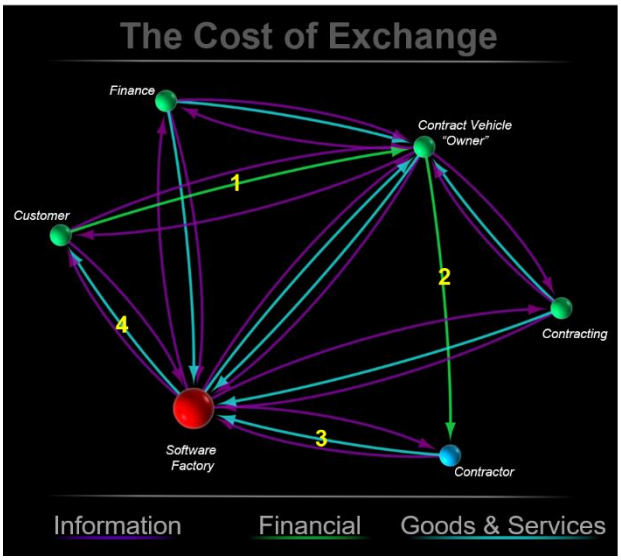

Figure 12: A typical factory-customer transaction. Four primary exchanges must occur to complete a transaction between the software factory and a customer. First, the customer sends money to a third-party organization that maintains an existing contract vehicle. Second, money is sent to the contractor via the vehicle. Third, the contractor provides services to the software factory. Finally, the software factory provides services to the customer organization. The software factory orchestrates every step of the transaction through its social connections.

Figure 13 illustrates the exchanges required to execute a typical transaction between a customer organization and the software factory. In addition to the primary exchanges, there is a secondary set of service exchanges representing the support functions such as contracting, finance, and legal services required to execute funds. Since the software factory does not maintain these functions organically, it must engage the help of external organizations. These exchanges, which must be repeatedly coordinated by factory personnel, represent a hidden cost of the transaction that would be missed using other analysis methodologies.

After the transactions of interest have been analyzed, it may be beneficial to assess the network models holistically to investigate areas of activity that appear unusual or out of place. Engage with experienced members of the organization and ask clarifying questions when the reasoning behind an exchange pattern is unknown using the network model as a visual guide. Finally, consider how patterns of exchange can be simplified to reduce the overall cost of individual transactions. Step 3 of the ION-A process concludes with an assessment of the communities within the inter-organizational network.

Identify clusters of organizations or communities of interest.

Identifying communities within the program's inter-organizational network allows practitioners to understand and navigate the complex relationships and interdependencies that exist between organizational subgroups. Community detection algorithms can be used to quickly identify these communities. We applied a hierarchical community detection algorithm (Pons & Latapy, 2005) to the software factory network; seven distinct communities of interest were highlighted (see Figure 14). The largest communities roughly aligned with the parent commands of the two partner organizations, activity associated with a local "innovation group", and a community of organizations associated with an industry partner acting under a Partnership Intermediary Agreement (PIA). The algorithm also highlighted three additional smaller communities: two of these communities can loosely be categorized as subgroups of Partner B; and one highlights a "gatekeeper" brokerage pattern between an industry partner and a cloud vendor.

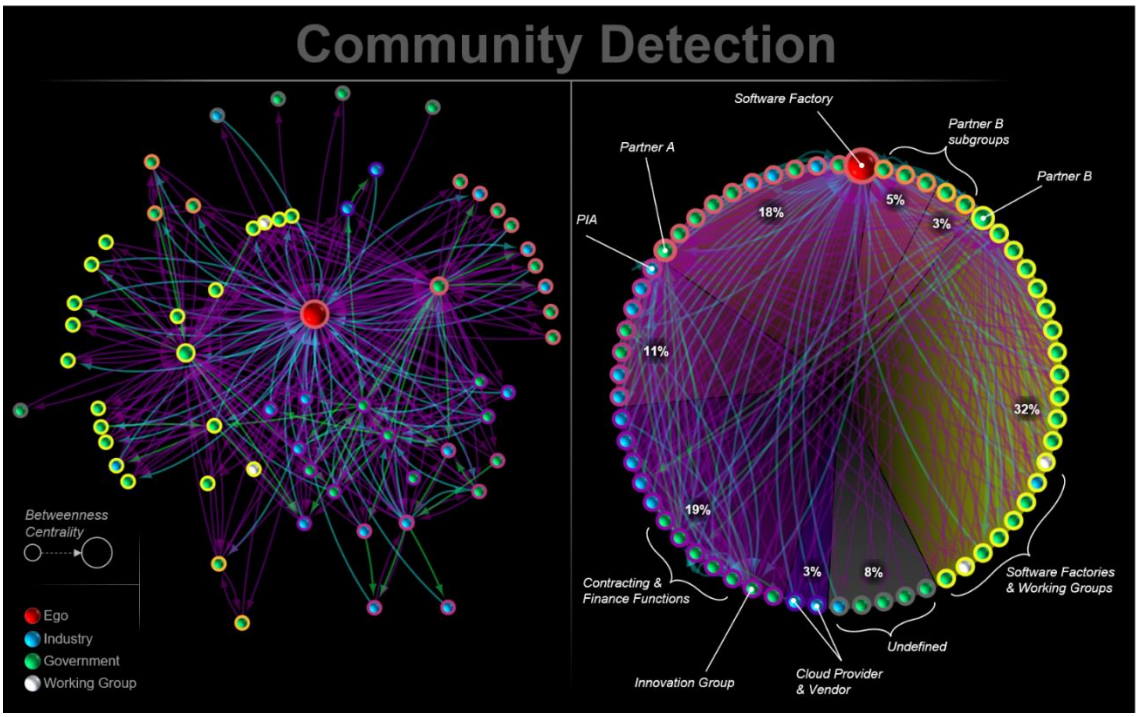

Figure 13: Community detection in software factory networks. A random walk community detection algorithm was utilized to identify communities of interest within the composite software factory network. Community detection algorithms identify sub-groups within a community by measuring connections between nodes. When applied to our network, 7 separate clusters were identified. While many clusters were easily attributable to known communities, others provided additional insight by highlighting previously unknown subgroups**.**

When applied to the software factory network, the community detection algorithm accurately identified known communities indicating its validity as an assessment tool. Additionally, its identification of smaller subgroups within the software factory network provided an alternative perspective on sub-group dynamics. By identifying communities of interest early in the program lifecycle, the program manager can plan the growth of the organization to align with community needs.

This completes the analysis step of the ION-A process. This step utilized the network models created in step 2 to identify influential organizations, organizational

relationships, common patterns of exchange, and communities of interest within a non-traditional program's inter-organizational network. This information, can provide the program manager with a foundational understanding of the inter-organizational dynamics of their program and can then be used to establish an informed big picture perspective.

Step 4 – Monitor and Review

Step 4 in the ION-A process is to periodically monitor and review the inter-organizational network. After completing an ION-A the program manager should regularly revisit the network as new contextual information becomes available. Networks by nature are temporal and previous assessments will become outdated if not maintained. Revisiting or even updating the network on a recurring basis will continue to provide value by providing a measurable visualization of organizational evolution and growth.

Assessing a program's inter-organizational network provides useful insight into organizational interactions and behaviors. Identifying influential organizations and relationships can be used to inform stakeholder management plan development and to assist in the prioritization of future organizational interactions. Assessing patterns of exchange can help identify dependencies regarding economic or social resources and mitigation strategies can be proactively developed in accordance with existing risk management best practices. Community identification can provide insight into social dynamics and it can help identify previously unknown communities of interest. In summary, the ION-A process can provide acquisition practitioners with a strategic perspective of the social and economic relationships of their programs. This section demonstrated this by presenting a case history of a non-traditional software program.

Table 5 summarizes the most relevant outputs and strategic insights from the ION-A process.

Table 5: Software factory inter-organizational network analysis results. The ION-A process provides practitioners with a method to assess the social and economic relationships of their programs. This case history assessed an existing DAF software factory in four key areas: organizational influence, relationship strength, patterns of exchange, and communities of interest. Strategic insights, organizational risks, and potential opportunities tailored towards the specific needs of this non-traditional software factory organization are derived from the ION-A process.

Influential Organizations	Relational Assessment	Exchange Patterns	Community Assessment
Partner Organizations A & B, Contract Vehicle Owners, Financial Brokers, Cloud Contract Liaison, PIA Org, Innovation Group	Weakest Relations: Uni-directional information reporting, Brokered vendor relations Strongest Relations: Factory – Partners, Factory – Platform Provider, Factory –Personnel Services Org	Customer – Factory Transaction: Requires 6 Orgs & minimum 4 economic exchanges; 3 industry partners funded through multiple sources; financial convergence through contract vehicle owners	Four dominant communities: Partner A, Partner B, Innovation Group, & PIA; 3 subgroups: 2 Partner B & Cloud Vendor

Strategic Insights:

- Industry partners are funded through multiple sources/customers with competing expectations, interests & requirements: this complicates program-level cost, schedule, and performance metrics/assessments. Alternative, customer-centric metrics should be developed.
- The software factory primarily acts in the "gatekeeper" broker role. Industry partner services are rendered to customers through the software factory: this is an unstated benefit/service of the software factory structure. Consider emphasizing this role when engaging potential customers.
- The software factory relies on many organizations within its network for resources (financial, services, & information): this prevents power centralization and provides increased autonomy to the factory at a cost to organizational stability.
- Accounting data is fragmented and dispersed across 5 orgs and two MAJCOMs: this makes it difficult to communicate total program cost.
- The software factory relies heavily on its social network to operate: this places a strain on key members of the organization (Cook, 1977) and limits organizational mobility (Blau, 2017). Consider implementing a customer liaison/outreach role to manage social exchanges.
- The social costs required to execute economic exchange is high: the factory should identify ways to reduce the cost of customer-factory transactions. Transaction steps are currently "tribal" knowledge and they should be formally documented to combat knowledge attrition.
- The relationships between cloud providers are weak and rely on third-party industry partners: this introduces a risk to a critical resource.
- Partner communities dominate the inter-organizational network: the software factory should strengthen relationships with additional communities to reduce partner dependencies.

Conclusion

This article began by pointing out that existing guidance for acquisition programs is tailored towards organizations with traditional hierarchical structures; thus it may not be sufficient for assessing non-traditional programs. This led to the formulation of a research question designed to address this gap in understanding:

How can program managers assess the inter-organizational structures of non-traditional software organizations?

To answer the research question, we applied a process called inter-organizational network analysis (ION-A) to demonstrate its effectiveness using the case history method on a non-traditional Department of the Air Force (DAF) software factory. The ION-A process, which is derived from the network analysis principles of multiplexity and egocentricity, has been simplified for practicality yet it still provides a technically sound foundation for practitioners to analyze and understand the organizational structures of their programs.

The ION-A process describes how practitioners can collect and organize network-centric program data that can be used to model the social and economic relationships of their programs. It also recommends common data sources, describes accessible data collection techniques, and identifies open-source software programs that can be used to create an inter-organizational network from the collected data. The ION-A process continues by guiding practitioners through an analysis of four focus areas of inter-organizational networks using a framework of contextual assessments. Table 6 includes a summary of this framework which addresses these four focus areas, provides initial assessment questions, and identifies a collection of relevant analysis techniques and network models that can be used in the analysis.

Areas of Interest	Contextual Assessment	Network Indicators
Relationship Strength	**Identify the weakest and strongest relationships within the organization's network.** a. Are the social and economic costs of these relationships in alignment with the goals of the organization? b. Can strategic relationships be strengthened by establishing additional types of exchange relations? Should they? c. Are any inter-organizational relationships stronger or weaker than anticipated? Why?	Composite Network (Figure 4a), Multiplexity & Tie Strength (Figure 5)
Organizational Influence	**Identify the most influential organizations within the network. Consider both the social and economic environments.** a. Which organizations are centrally located? Why? b. Which organizations are acting in broker roles? What is their affiliation? c. Which organizations control access to resources? Information? Is this an acceptable risk? d. Does the flow of resources converge at any specific organization? e. What happens if these organizations exit the network?	Economic, Social, and Composite Networks (Figure 4a-d), Centrality Measures, Network Positioning, Resource Flows, Broker Roles (Figure 6)
Communities of Interest	**Identify clusters of organizations or communities of interest.** a. Is it possible to explain the existence of each community? b. Are there any subgroups that were previously unknown? c. Do any organizations seem misplaced? Why? d. Are communities clustered around specific organizations? What are they?	Composite Network (Figure 4a), Community Detection (Figure 8)
Exchange Patterns	**Identify common patterns of exchange.** a. How many exchanges must occur to complete a transaction? Should this be reduced? b. Are critical patterns of exchange formally documented? c. Are there any patterns of exchange that could be standardized? Simplified? d. Are economic exchanges supported by social exchanges? If not, why?	Economic, social, and composite networks (Figures 4a-d), Exchange patterns (Figure 7), Resource flows, Directionality

Table 9: A framework for conducting the ION-A process. The ION-A process guides practitioners through an assessment of the inter-organizational network of their programs. The analysis framework used in step 3 of ION-A focuses on four primary areas of interest in order to provide a foundational understanding of program dynamics. Questions designed to guide the assessment are provided along with relevant network-based indicators.

In addition to developing and demonstrating a new assessment approach, the case history also provided an in-depth look into the underlying mechanisms that drive non-traditional software factories. Additionally, many of the network-derived attributes highlighted in our assessment can be explained using established academic literature. Strategic network management is a rapidly maturing field that has expanded beyond academia and is currently being applied by industry practitioners to provide new insights into organizational behaviors (Cross & Gray, 2021; McDowell et al., 2022). As open-

source graphing applications (such as those used in our analysis) continue to evolve and become more accessible, the ability of practitioners to apply network analysis to gain insight into their own organizations will continue to increase. The cultivation and management of inter-organizational partnerships and stakeholder relationships is a critical component of any program. The application of network assessment techniques like those demonstrated in this article allows practitioners to better understand these relationships early in the program lifecycle.

This article contributed to both the Department of Defense and academia by demonstrating a new and intuitive approach to organizational analysis while also providing an in-depth look into an emergent acquisition organization the software factory. This case history provided insight into the structure and relationships of an organization prior to a formal entry into the acquisition system. Identifying alternative analysis methods for such organizations is important since this early period is largely undocumented and thus a possible source of confusion for many acquisition professionals.

In conclusion, the inter-organizational network analysis (ION-A) process was demonstrated as a valuable assessment tool for acquisition practitioners seeking to understand the complex structures and interactions within non-traditional software organizations. By providing a deeper understanding of inter-organizational structures and behaviors, network analysis methods can help inform decision-making and improve the success of non-traditional defense acquisition programs.

References

Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi: An open source software for exploring and manipulating networks. *International AAAI Conference on Weblogs and Social Media*. https://gephi.org/users/publications/

Blau, P. M. (2017). Exchange and power in social life. *Exchange and Power in Social Life*, 1–352. https://doi.org/10.4324/9780203792643/EXCHANGE-POWER-SOCIAL-LIFE-PETER-BLAU

Brass, D. J., Galaskiewicz, J., Greve, H. R., & Tsai, W. (2004). TAKING STOCK OF NETWORKS AND ORGANIZATIONS: A MULTILEVEL PERSPECTIVE. *Academy of Management Journal*, *47*(6), 795–817.

Cook, K. S. (1977). Exchange and Power in Networks of Interorganizational Relations. *The Sociological Quarterly*, *18*(1), 62–82. https://doi.org/10.1111/j.1533-8525.1977.tb02162.x

Cross, R., & Gray, P. (2021). Optimizing Return-to-Office Strategies With Organizational Network Analysis. *MIT Sloan Management Review*. https://sloanreview.mit.edu/article/optimizing-return-to-office-strategies-with-organizational-network-analysis/

DAU. (2022a). *A Guide to DoD Program Management Business Processes*. https://www.dau.edu/pdfviewer?Guidebooks/DAG/A-Guide-to-DoD-Program-Management-Business-Processes.pdf

DAU. (2022b). *A Guide to Program Management Knowledge, Skills and Practices*. https://www.dau.edu/pdfviewer?Guidebooks/DAG/A-Guide-to-Program-Management-Knowledge-Skills-and-Practices.pdf

Driessnack, J. (2017). *Program Manager Toolkit*. https://www.dau.edu/tools/Lists/DAUTools/Attachments/143/Program%20Manager%20Toolkit.pdf

Freeman, L. C. (1977). A Set of Measures of Centrality Based on Betweenness. *Sociometry*, *40*(1), 35. https://doi.org/10.2307/3033543

Gould, R. v, Fernandez, R. M., & Fernandezt, R. M. (1989). Structures of Mediation: A Formal Approach to Brokerage in Transaction Networks. *Source: Sociological Methodology*, *19*, 89–126.

Granovetter, M. S. (1973). The Strength of Weak Ties. *American Journal of Sociology*, *78*(6), 1360–1380. https://doi.org/10.1086/225469

Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, *13*(11), 2498–2504.

Levine, S., & White, P. E. (1961). Exchange as a Conceptual Framework for the Study of Interorganizational Relationships. *Administrative Science Quarterly*, *5*(4), 583. https://doi.org/10.2307/2390622

Maghssudipour, A., Lazzeretti, L., & Capone, F. (2020). The role of multiple ties in knowledge networks: Complementarity in the Montefalco wine cluster. *Industrial Marketing Management*, *90*, 667–678. https://doi.org/10.1016/j.indmarman.2020.03.021

McDowell, T., Horn, H., & Witkowski, D. (2022). *Organizational Network Analysis Gain insight, drive smart*. https://www2.deloitte.com/us/en/pages/human-capital/articles/organizational-network-analysis.html

Perry, B. L., Pescosolido, B. A., & Borgatti, S. P. (2018). *Egocentric Network Analysis*. Cambridge University Press. https://doi.org/10.1017/9781316443255

Pons, P., & Latapy, M. (2005). Computing communities in large networks using random walks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3733 LNCS*, 284–293. https://doi.org/10.1007/11569596_31/COVER

Ryan, Z. O., Reith, M. G., & Beach, P. M. (2022). Defining the DoD Software Factory: A Network value Approach. *Crosstalk*. https://community.apan.org/wg/crosstalk/m/documents/420788

Wasserman, S., & Faust, K. (1995). *Social Network Analysis: Methods and Applications*. Cambridge University Press.

Summary

This manuscript was the third and final written for this thesis. Its primary goal was to provide an assessment methodology that acquisitions practitioners can use to understand the dynamics of non-traditional defense programs in light of their non-hierarchical forms. In parallel, the article also described and demonstrated how a multiplex, ego-centric network analysis approach could be used to study a department of defense software factory, focusing on its economic and social environments. Within the analysis, the article also highlighted fundamental concepts of networks and it discussed how these concepts could be applied by program managers to garner insight into the strategic big picture perspectives of their programs.

This article primarily focused on the Inter-Organizational Network Analysis (ION-A) process which was derived from the network analysis techniques employed in the case study. While this is a notable contribution, the data generated from the case study also further reinforced the network theory of software factories developed during the second phase of research. In doing so, this article continued to push the ball forward within the field of defense acquisitions by building upon the findings of the first two articles and providing a new methodology for understanding the complex social and economic relationships that influence the success of non-traditional acquisitions programs.

V. Conclusions and Recommendations

This chapter summarizes the results of the three research phases conducted in support of this thesis and it discusses how the credibility and relevance of these results are strengthened when viewed in accordance with the principles of the mixed methods methodology that was used to guide the overarching research process. It begins by addressing each article individually before discussing how when viewed holistically, the composite results directly address the overarching exploratory theme of the thesis. Next, this chapter discusses the limitations of this research before recommending additional areas of further study. Finally, this chapter concludes by highlighting how the research contained within this thesis contributed to both the Department of Defense (DoD) and academia by establishing the initial foundation of knowledge necessary to guide future software factory research and strategic growth.

Summary of Research Phases

This thesis employed a mixed methods research methodology which broke down the overarching objective of this thesis, to expand the DoD's knowledge of the Software Factory Ecosystem, into three sequential phases: *Understand*, *Define*, and *Describe*. This section summarizes the results of the research phases.

Understand the characteristics of software factories.

Primary Research Question: What are the defining characteristics of software factories?

The first phase of research established a baseline understanding of the software factories by employing a narrative research process to assess the current state of the software factory ecosystem. The research began with the development of a working

theory that software factories shared internal organizational characteristics. This phase then utilized primary research data collected from multiple sources to include interviews and internal software factory documentation in an attempt to identify commonalities between software factory organizations. Contrary to the working theory, the findings of the first phase of research revealed that software factories cannot be solely defined by their internal characteristics. Nonetheless, the phase still addressed the first primary research question by capturing a range of common characteristics of software factories and it presented them to the community in an accessible framework.

Define the software factories.

Primary Research Question: What are the organizational structures of the software factories?

The second phase of research looked beyond the internal characteristics of software factories and instead it attempted to define them based on their underlying organizational structures. Building on the previous stage and continuing through the inductive-deductive reasoning cycle, a new working theory was developed. This second theory, that software factories shared non-traditional organizational structures, was developed based on data collected in the first phase of research and from data collected from an in-depth review of historical strategic software acquisition documentation. A participant observational analysis was then conducted; the researcher embedded themselves within the software factory community and a sample software factory and its interactions with various software factory working groups was observed. Additionally, previously collected documentation was reassessed against the findings of the analysis.

The findings of the participant observation and the review of previously collected data indicated that software factories exhibited structural characteristics that align closely with the network structure defined by Powell et al.'s (1990) market-network-hierarchy continuum of organizational structures. While this phase did not produce a traditional software factory definition, it did present the network theory of software factories which effectively defined the software factories as economic networks based on their organizational structures.

Describe the software factories.

Primary Research Question: How can acquisition practitioners analyze non-traditional organizations like software factories?

The third phase of research studied the inter-organizational networks of a sample software factory using the case study approach. Building upon the network-based theory of software factories developed in the second phase of research, economic and social behaviors were captured through a series of semi-structured interviews with software factory leaders in order to better understand the software factory's relationships. A multiplex, egocentric network analysis approach was employed to capture these behaviors due to its relevance to the network organizational structure identified in the second phase and due to its accessibility as a metaphorical tool. The data collected from this phase was then analyzed and compared against the results derived from the participant observational study conducted in Phase 2. The results from the research phase were presented to the software factory of interest which deemed them both credible and relevant.

To address the primary research question, the multiplex, egocentric network process used within the case study was tailored for repeatability and accessibility. The finalized process, which was specifically developed for use by acquisition practitioners, was then presented to the acquisition community as the Inter-Organizational Network Analysis (ION-A) process.

Methodological Insights & Study Limitations

The first two manuscripts discussed in this thesis employed purely qualitative methods in their analysis: narrative research and participant observation. These manuscripts primarily utilized interviews, observations, and literature reviews to establish a foundational understanding of the software factories. For example, the senior leader and software factory founder interviews referenced in the first article helped establish the internal context of the software factories by focusing on the experiences and knowledge of the interviewees. The approach used in the second article was similar, additional observations and community interactions were used to supplement the data collected during the first phase; however, the intent of the research phase differed. Instead of seeking only to understand the software factories, its intent was to identify a unifying definition or theory that could describe the greater software factory phenomenon. The third article also employed a qualitative approach, a case study, and augmented it using quantitative network data to measure the relationships and centrality of organizations within the software factory's inter-organizational network.

When viewed holistically, the three manuscripts each address the overarching research objective by approaching the software factory ecosystem from different methodological perspectives. This approach reinforces the trustworthiness of the

research. Demonstrating this, the network-based theory presented in article two is reinforced by the results of the first and second articles. Simply put, the characteristics identified in the first article support a network theory of software factories and the network models generated within the third article illustrate how the studied organization maintains a network-based structure. This effect is called triangulation and it supports the credibility and inference quality of the combined works (Tashakkori & Teddlie, 2009).

In addition to triangulation, the credibility of the research stream was supported through persistent observation and community engagement. Observation of the software factory community began early in the research cycle with the combined observation window expanding 12 months of study. This extended duration ensured that multiple perspectives were captured during the research cycles. In totality, the combined research contains perspectives and insights from many viewpoints to include senior leaders, software factory founders, directors, engineers, and mid-level managers. The outputs from the three phases of research were presented to the community of study to solicit feedback and assess the inference quality of the works. Feedback from the community being studied was largely positive which added weight to the credibility of the conclusions.

The mixed methods approach offers notable benefits in terms of credibility and quality, yet this thesis is not without limitations. One limitation is associated with the scope of this research. Although various perspectives from the software factory community were gathered, the number of participants from an organizational perspective is limited. The software factory community is vast; 18 organizations are listed on the AF

CSO's website and a significant number of software factories remain unidentified or belong to sister services. Considering this, additional research that encompasses a diverse set of organizations should be conducted to improve the transferability of this thesis's results.

The limitations of this research also extend to the methods used to collect data on organizational behaviors. The first challenge was accessing relevant data. The software factory ecosystem is a closed community and gaining access to organizational information was very challenging. Because of this, an interactive observational approach was necessary to develop trust within the community. This is especially evident during the second phase of research. The second limitation is related to the Hawthorne Effect, or the tendency for individuals to modify their behavior when they are observed (Adair, 1984). Because of the inherently social nature of the topics addressed within this thesis, it must be assumed that the data provided by the software factories and interviewees is biased based on their perspectives, organizational goals, and opinions. While this thesis tried to account for this bias through active engagement, bias that extends throughout the community must be assumed to exist. The software factory ecosystem is in the early stages of growth and thus, it is in the community's best interest to maintain a positive outward appearance. Finally, this thesis was limited by the lack of metrics on software factory performance. Without internal metrics, it is impossible to associate organizational effectiveness with the results in this study. Instead, this study can only identify the existence of organizational characteristics; this inherently limits the operational applicability of these results until internal performance metrics can be developed.

Recommendations for Future Research

Exciting opportunities exist to expand upon the theoretical foundation and the network-based analytical approaches established within this thesis. The first opportunity is to expand network data collection to other organizations within the software factory ecosystem. Assessing the inter-organizational networks of the more established and mature organizations like DoD's Platform One or Kessel Run would provide interesting insights into the relationships that drive the greater ecosystem. Assessing multiple organizations could also provide insight into the attributes of the organizations in the software factories networks and this would help strengthen or weaken the network-based theory presented within this thesis. Alternatively, assessing the social networks of the individual participants acting within the ecosystem could also provide new insights and an alternative perspective.

A second opportunity for future research lies within the socio-economic field. Article three of this thesis discussed structural dependency and power within software factory networks however, non-structural dependency between organizations can also be measured. Understanding the dependency relations between the software factories and their industry partners would provide actionable information to inform the structuring of future programs which closely interact with industry to co-develop materiel solutions.

Finally, and arguably the most relevant extension to this research is expanding it to include the development of an ecosystem architecture to guide future organizational growth. While this thesis was able to identify many of the core characteristics of the software factory ecosystem, it was unable to develop a formal model based on its findings. Model-based systems engineering techniques could be utilized to develop and

present a useful ecosystem framework that could be used by decision makers within the DoD.

Significance of Research & Conclusion

The three manuscripts included within this thesis each individually contributed to the Department of Defense and to academia. The initial article, *Defining the DoD Software Factory*, was the first to formally study the internal organizational characteristics of the software factories and subsequently capture those characteristics in a formal framework. The second article, *Hierarchy, Networks, and Software Factories*, built upon the first by presenting an alternative theoretical perspective by which to view and define the software factory ecosystem. This article was the first to identify the disconnect that exists between the DoD's hierarchical management strategies historically used to manage software programs and the network-centric structures of the software factories. The manuscript assessed this dichotomy and it recognized that the organizational structures of software factories fundamentally differ from traditional defense programs. The resultant output of the article was a structural theory of software factories which formally identified and defined these organizations as economic networks. The third and final article, *Deconstructing the Software Factory*, acknowledged that the non-traditional characteristics and structures of the software factories that were identified within the first two manuscripts had left the acquisitions community without an accessible methodology to analyze and understand these organizations. It developed, demonstrated, and presented an assessment framework grounded in network analysis techniques which was specifically tailored towards acquisition practitioners seeking to understanding non-traditional software organizations.

The resultant output, the Inter-Organizational Network Analysis (ION-A) process, provides practitioners with a practical and repeatable method of analysis by which they can assess their own programs.

The theories, inferences, and outputs of the research phases included within this thesis were presented to and were well received by leaders within the DoD and the joint software factory community. In addition to the tangible contributions of the three individual manuscripts, this thesis' emphasis on active community engagement also helped shape widespread discussions and explorations into the fundamental characteristics of the software factory and it directly influenced existing department level ecosystem architectures and factory definitions. As a direct consequence of the active, engaged, and iterative approach used to develop and refine the theoretical propositions employed within this thesis, heuristic value was continuously provided to the DoD's Software Factory community. In short, while the tangible contributions of the included manuscripts have provided demonstrable value to the Department of Defense, the intangible benefits associated with the ongoing community discussions, questions, and new perspectives that arose from the holistic process should be emphasized as the hallmark benefit and contribution of this research.

Appendix A —The Historical Evolution of DoD Software Acquisitions

Understanding the historical evolution of Department of Defense (DoD) software development programs and by extension, the software factory, played a critical role in informing the direction and development of this thesis. This appendix captures the extensive historical review that was conducted in support of this thesis starting with the formal introduction of the factory concept in a journal article that described development processes at the System Lifecycle Development Corp (Bratman & Court, 1975). Next, the discussion progresses by identifying important software challenges within the DoD, through the use of primary government source documents from the early 1970's through 2020's that organizational constructs like software factories have been intended to solve. Finally, this appendix concludes by discussing the ongoing software development efforts that were outlined in the introductory chapter of this thesis by illustrating how the growth of organizations like the modern software factories represents a fundamental break from the organizational constructs and strategic approaches employed by the DoD in the past. Upon concluding this appendix, the reader will understand historical challenges associated with DoD software acquisitions and how the DoD has repeatedly employed various organizational constructs and strategies in an attempt to solve those problems.

The Software Factory – (1960 – 2022)

The software factory and subsequently its definition has shifted in focus over time varying in meaning since its initial conception. Two threads exist in academic literature, starting in the late '60s which refer to the factory as both a specific type of organizational structure and to the automation and tools necessary to improve software quality and the efficiency of developers (Cusumano, 1991). The software factory thus exists and is

referred to as two separate states throughout literature- the software factory as an organizational construct and the software factory as a technical capability delivery mechanism.

Multiple examples of what would be considered software factory-type organizations being deployed within the defense and software industries dating back to the mid ’70s. Cusumano in his in-depth overview of the history of the Software Factory construct identified multiple early attempts to utilize a factory-based concept to develop software both in Japan and the United States, with varying degrees of success (Cusumano, 1989). His findings, captured through a series of publications, illustrate that many of these early examples such as Hitachi Software Works and System Development Corp (SDC), whose work was intertwined with multiple Department of Defense projects at the time, worked to perfect a centralized factory model that allowed it to consolidate the often-limited technological workforce into a single location while providing them with all the tools, processes, and functions necessary to develop and deliver software.

Cusumano highlights through his series of articles, how in addition to ultimately failing to solve some of the most prevalent challenges with software development, the early software factories all encountered difficulties and challenges associated with their management and growth. For example, the software factory model demanded a long-term commitment of resources due to the management and systems engineering resources required to make them function. Additionally, their scale complicated requirements definition, and due to process and toolchain limitations, they demanded a narrow scope in order to function efficiently. Due to these challenges, and with advances in development technologies, the formal software factory eventually fell out of favor as industry within

the United States moved away from the centralized model to more customer-centric development practices. Academic research on software factories as an organizational construct largely falls off in the mid-1990's in response.

Within the last 5 years the software factory term has seen a resurgence within departmental documentation by illustrating the software factory as a developmental construct. The 2018 DIB report, introduced in Chapter 1 of this thesis, mentions the term software factory 9 times – primarily referring to the software factory as a technical development mechanism. The CIOs DevSecOps playbook (Department of Defense, 2021) also refers to the software factory as a development mechanism referencing it as a critical component in developing software; the CSO's website does however also acknowledge and promote the factory utilizing an organizational definition through its "Software factories" registration and designation (Assistant Secretary of Acquisition & Air Force Chief Software Office, 2021). Interestingly, this interpretation and change in definition from technical to organizational also occurred in early factory implementations as SDC documentation defined the software factory not as the organization it would eventually become but as "an integrated set of software development tools to support a disciplined and repeatable approach to software development" (Bratman & Court, 1975).

The software factory examples of the '70s, '80s, and early '90s offer important insights into some of the challenges that the Department of Defense has grappled with during the past 50 years. Some of the problems that the original factories were meant to solve such as standardization and control over the development process, difficulty with design and requirements specification, technical standardization, limited reuse, and strategic management of assets have been issues that the Department of Defense has

historically struggled to resolve. These problems are systemic in nature and are repeatedly reflected in various departmental reports and memorandums. The following sub-section highlights key examples of these ongoing challenges.

DoD Systemic Software-centric Challenges - (1970 – 2022)

The DoD has struggled to resolve the same core systemic issues since it first began procuring and acquiring software type solutions via major acquisitions programs in the 1950's. During software's infancy, the '50s-'60s era largely saw software transition from a hardware-centric mindset to the rapid expansion into a field of study of its own (Boehm, 2006). It is during the 1970's, however where software systems development saw an explosion of growth and the core challenges associated with its activity became readily apparent within literature. While many challenges can be loosely categorized as specific to the era in which they were identified, some are more broadly applicable to the larger study of military software acquisitions. Broadly speaking, four recurring themes with direct relevance to the DoD's historical management of its software enterprise were identified in this review. Debate and opinions on how best to resolve these core challenges have varied over the past 70 years; however, the issues have largely remained unchanged. This section reviews departmental documentation in order to identify policy, recommendations, and management challenges inherently linked to these recurring themes (see Table 7) while simultaneously illustrating their historical significance.

Table 10: Historical DoD software themes.

Department of Defense Systemic Software Challenges & Issues
Disconnects between the DAS and software development best-practices.
Standardization of systems, software, tools, and methodologies.
Strategic management and growth of the DoD software workforce.
Measuring and controlling cost, schedule, and performance of DoD software programs.

Studies into software acquisitions and development within the DoD began in the 1970's with reports for how best to approach challenges relevant to software coming out of both the department and academia. Early research into the field identified core issues or focus areas that would continue to remain at the forefront of the software acquisitions debate for the next 60 years. Thematic challenges include the existence of disconnects between DAS policies and software development practices to include: requirements management and program flexibility; how and or if to standardize tools, solutions, and methodologies; how best to strategically manage software personnel; and how to manage and control cost, schedule, and performance measures for DoD programs (McMillan et al., 1977; Delauer et al., 1974; Keller, 1977; Buchsbaum, 1978;, Gansler, 1975).

Recommendations on how to tackle challenges associated with maintaining flexibility within the Defense Acquisitions System (DAS) vary and are addressed through different avenues in the '70s. Early reports such as the *Report of the Task Force on Electronics Management*, recommend working within the DAS by identifying alternative means to meet requirements early in the acquisitions cycle. Other recommendations propose what can be considered relatively modern approaches such as encouraging incremental delivery strategies for software and suggesting wholesale modification or

exclusion of the DAS to allow for increased adaptability of requirements by software systems (Buchsbaum, 1978; Walden et al., 1978). These recommendations were at odds with DoD software development policies at the time which favored the highly structured waterfall model (Boehm, 2006). The literature does not offer a universally accepted solution for how best to develop software within departmental constraints. However, a general consensus exists on the numerous disconnects between how software is procured and developed and how the DoD formally directs the execution of those programs through the DAS.

Discussions and debates related to the standardization of software development tools, software systems solutions, and development methodologies have also been ongoing since the DoD began procuring software. While the majority of the 1970's reports arrive at the general consensus that standardization as related to procurement could potentially reduce cost, how best to implement standardization and at what level is appropriate remained debated. For example, broad system standardization is identified as "in conflict" with DoD policy in the 1974 Electronics Management report, yet the same report also suggests that there could be "substantial positive impact" if an appropriate standardization program can be conceived. The discussion around standardization also extends into development methodologies and development languages with a 1978 scientific advisory board report recommending "freezing" the methodology used by programs as a standardization solution (Walden et al., 1978); on the other hand, a 1977 report by the Air Force Studies Board recommended continued research and growth into design methodologies. Finally, numerous GAO reports identify how the DoD's own

policies and guidelines have hampered the standardization of data and code elements to the detriment of programs (Keller, 1977).

The debate around the benefits of standardization, and the emphasis on its importance as a primary focus area can be highlighted in a GAO report to the SECDEF titled *The Department of Defense's c for Military Computers – A More Unified Effort is Needed* (Gutmann, 1980). This report advocates for broad standardization across the departments to include systems, architectures, and management frameworks. While not all of the primary sources reviewed for this thesis are as direct in their recommendations as the 1980 GAO report, most contain either direct or indirect themes associated with how best to manage the DoD's software acquisitions enterprise.

How best to control and manage software acquisitions programs cost, schedule, and performance also rises as a primary theme within early literature. The 1974 Electronics Management report acknowledges that the DOD cost accounting system does not allow for the proper management of indirect and direct costs associated with software making it "impossible" for the DoD to accurately determine true costs associated with electronics development. GAO reports at the time further expand on these issues identifying multiple cost-schedule performance challenges to include failure to follow procurement practices, difficulties in determining user-needs, and poor design and planning (Keller, 1977). A 1977 study by the Air Force Studies Board further highlights these challenges, and as a solution, provides a detailed acquisitions strategy and contracting plan built around an incremental approach designed to improve the management of programmatic cost, schedule, and performance risk.

Agency reports continue into the 1980's where a large body of literature centers around how best to address the four primary themes identified in the 1970's. Of notable mention, the National Research Council conducted an in-depth study into software development policies titled *Adapting Software Development Policies to Modern Technology* (1989), in response to the growing realization that existing acquisitions management practices and policies were failing to keep pace with modern technology systems. In addition to providing 26 recommendations geared towards addressing acquisitions shortfalls, the report included a review and summary of preceding major software acquisitions studies. Notable recommendations from the report include maintaining the flexibility of acquisitions programs through alternative management methodologies, contract flexibility, and incremental acquisition approaches and tailoring; improved software personnel management; standardization of "Software Engineering Environments" across programs, and increase investment in technology transfer and growth.

Additional reports throughout the 1980's provide similar recommendations in response to the recurring challenges identified in the '70s. The following themes, from various reports, were drawn from the NRC report summary: challenges dealing with strategic management of programs are noted to include the lack of departmental guidance and structure (Glaseman & Davis, 1980); the poor management of the software lifecycle (General Accounting Office, 1981); and the insufficient management of software personnel (Druffel, 1982). Challenges stemming from the inherent inflexibility of the acquisition's cycle are also noted in multiple reports (Booz Allen, 1981; Druffel, 1982;

Fowler, 1987). Standardization[7] around software architectures, practices, and measures are heavily mentioned to include recommendations pertaining to development environments (Vick et al., 1985;McDonald, 1988), languages, and metrics (Fowler, 1987). Finally, challenges associated with appropriately managing the cost, schedule, and performance of software programs are addressed. Notable recommendations for these issues include centralized oversight and uniform tracking, user engagement and requirements development (Munson, 1983), as well as development of universal management tool-sets (Vick et al., 1985).

The 1990's were largely dominated by a trend of widespread standardization and control of development models, languages, and methodologies across the Department of Defense. It is during this era that the DoD chose to mandate the Ada language for use in all new software development projects (Boehm et al., 1997). The strategic decision to top down direct a specific technological solution had widespread impacts on both the DoD's organic software development capabilities and to the wider software industrial base. A 1997 report by the National Research Council reviews in-depth the impacts of the DoD's Ada mandate and notes that, as a result of policy, the DoD was no longer aligned with the commercial sector in development tools, languages, and skillsets. The DoD's choice to lock-in a language had essentially left it as the sole user of a specific development language and architecture which significantly hampered technological progression.

[7] The DoD and GAO actively debated through policy documents the benefits of standardizing all software development around the ADA language throughout 1985-1987. The DSB heavily favored standardization however, the GAO was against this approach citing the potential for negative impacts on the technological advancement of the defense industrial base (Brooks, 1982). DoD decision-makers eventually won the debate and Ada was mandated for all new software development projects from 1987-1997.

Breaking from previous assessment formats, the Defense Science Board issued its third major report on DoD Software in 2000 centered primarily around a review of the validity of past recommendations and as such included a summary of reports made by the board and other investigative agencies throughout the 1990s (Hansen & Nesbit, 2000). The board concluded that the 134 past recommendations made throughout the 90's could be grouped into five primary areas. The areas identified within the report were software architecture, software technology, workforce issues, contract strategy, and acquisitions policy. In addition to categorizing past recommendations the board reiterated additional key observations such as consistently poor requirements-setting and management, disparities between commercial best-practices and practiced favored by the DoD, and a lack of experienced DoD software practitioners.

While the overall themes outlined at the beginning of this section largely remained consistent over time, one notable change in strategic directive that is still relevant to today's software strategy was a formal acknowledgment that the DoD had been largely outpaced as a driver of software technology development by the commercial sector. This shift can be noted in literature as research began to explore and subsequently recommend an increased reliance on acquiring software through the commercial sector. This transition was also evident through policy directive as the DoD's software standardization requirements shifted--effectively ending the DoD's decade long Ada era.

The post Y2K era can be defined by an explosion of growth in connectivity through the internet throughout both the government and commercial sectors and the rise in influence of agile methodologies that had begun to develop in the late 90's (Beck et al., n.d.). Trends within the DoD at this time mirrored commercial sector movements and a

shift to “Net-centric, service oriented” strategy and a push for enterprise IT acquisition can be noted as core strategic movements (Grimes, 2007). Standardization also continued to be a major theme, however its focus shifted from language, as was the case in the 90’s, and instead was illustrated through a call to standardize around open standards and architectures for both data and software in order to facilitate the departments connectivity goals.

While standardization could be noted as the hallmark theme throughout the 90’s, the 2000’s saw a shift in focus towards defense acquisitions reform as legislative actions sought to reduce program costs and risk for software programs by increased reporting and oversight requirements. The FY2007 NDAA implemented multiple legislative mandates targeting IT acquisitions to include defining criteria for systems, mandating reporting requirements, and prescribing specific timeframes for business systems (*John Warner National Defense Authorization Act for Fiscal Year 2007*, 2006). In addition to these legislative changes the DoD updated the acquisition process in 2003 in an attempt to provide some additional flexibility for modern practices and methodologies while still maintaining oversight. Notably, the new model still maintained the hallmarks of past traditional acquisitions pathways such as a heavy reliance on formal documentation and milestone-based based approach to programmatic progression (Vitto et al., 2009).

In 2018 the Defense Science Board released its most recent report on the state of software development within the Department of Defense (LaPlante et al., 2018). Within the executive summary the board acknowledges that the DoD is still struggling with the same underlying issues that have hampered DoD software acquisitions programs over the past twenty years. The recommendations and supporting reasoning provided by the

board also largely remain unchanged. Recommendations focused on aligning the DAS with commercial best practices, implementing agile and incremental delivery models, managing and controlling programs throughout their lifecycles, and growing the software workforce. Finally, the DSB report resurrected the term "Software Factory" and incorporated into their primary recommendation – suggesting its implementation should be a cornerstone around which the future of DoD software acquisitions is built.

Chapter One of this thesis provided an overview of recent reports, academic literature, and legislative documents that highlighted how the Department of Defense has continued to face challenges in the acquisition and management of software centric materiel solutions through the modern day. It also reviewed the results of the most recent Defense Innovation Board Study and it's four primary recommendations as an illustrative tool by which to "set the stage" and inform the reader about the core drivers behind existing software acquisitions policies. It is important to highlight that the DIB recommendations, while highly relevant, are not new and are directed at the same four primary thematic challenges that have been discussed throughout this appendix.

Appendix B – An Overview of the 2018 DIB Study *Software is Never Done*

This appendix includes an analysis of the four primary lines of effort established in the 2018 Defense Innovation Board Study. In 2017 the National Defense Authorization Act contained a provision, Section 872, which directed the Defense Innovation Board (DIB) to study the current state of affairs of software acquisitions within the DoD. The NDAA tasked the DIB with 4 primary objectives specifically related to streamlining software acquisitions within the DoD. The four objectives were to review acquisition regulations and organizational structures, review ongoing software development and acquisition programs, produce recommendations for legislative and non-legislative actions to drive change within the DoD, and produce any other recommendations as appropriate (House of Representatives, 2017).

The resultant DIB study titled "Software is Never Done. Refactoring the Acquisition Code for Competitive Advantage" covered in-depth the state of software within the Department of Defense and delved into great detail covering the different aspects that influence the DoD as an organization. Within the report, the DIB highlights 4 specific lines of effort with 10 recommendations that, since their initial publication, have influenced and driven congressional legislation and reform specifically designed to improve the effectiveness of the software acquisition process. The DIB report spurred renewed interest into the challenges associated with software acquisitions and due to its emphasis on commercial practices, it enabled the software factories to explore non-traditional approaches. Because the growth of the software factories was impacted significantly by this report, this appendix includes a detailed breakdown of the board's recommendations.

1. Congress and the DoD should refactor statutes, regulations, and processes for software.

The most visible and potentially impactful outcome of the first line of effort's recommendation was the creation and implementation of DoD Instruction 5000.87, Operation of the Software Acquisition Pathway (Lord, 2020). The new pathway provides an avenue for software programs to develop capability utilizing modern development processes outside of the traditional JCIDs regulatory limitations. While less prescriptive in nature the new pathway still provides a framework via formal documentation requirements required for entry to include a Capability Needs Statement (CNS) and an Acquisition Decision Memorandum. Part of CNS development involves formally defining desired capabilities prior to moving forward with execution. As of September ten USSF/USAF sponsored programs have moved to adopt the Software Pathway – JCC2, Space C2, UP, WARP Speed, WDA Inc. 5, AOC, ADCP, C2IMERA, Mod & Sim, T&G (Carney & Konwin, 2021a, 2021b). The Software Acquisitions pathway is a key component that spans and touches on all of the DIB's recommendations.

In addition to developing a new acquisition pathway, OSD is currently piloting a new budget authority, BA-08 which is specifically designed to meet the needs of programs implementing DevOps methodologies. BA-08 provides a single color of money that combines what would normally be separate RDT&E, procurement, and sustainment dollars into a single pool (OSD, 2020). The Air Force has budgeted $418M for FY22 under this new software pilot program with the Space Force budgeting $155M (USAF, 2021). Two of the existing software factories involved in the pilot program are Kessel Run and Kobayashi Maru (*Department of Defense Fiscal Year (FY) 2022 Budget*

Estimates Justification Book Volume 3b, 2021; *Department of Defense Fiscal Year (FY) 2022 Budget Estimates, RDT&E, Space Force*, 2021).

2. *The Office of the Secretary of Defense and the Services should create and maintain cross-program/cross-service digital infrastructure.*

In alignment with the DIB study, in 2018 the DoD initiated a standup of a joint OUSD, DoD CIO, DISA, and Services wide DevSecOps initiative which began producing guidance on both the technical and organizational components necessary to begin implementation of shared cross-program digital infrastructure. The DevSecOps initiative made a number of changes to address DIB recommendations with infrastructure efforts primarily aligning under an Enterprise IT initiative which stood up Platform One and Cloud One as the first DoD-wide managed services (Chaillan, 2020). Additionally, the USA&S and DoD CIO signed a DoD-wide DevSecOps enterprise reference design which laid a framework for future software program MVPs via the Platform One effort and laid out the first step towards DoD-wide ATO reciprocity (Deasy & Lord, 2020).

Since its initial standup Platform One and Cloud One have rapidly grown as a potential solution for the cross-program/cross-service digital infrastructure goal outlined by the DIB. Platform One's services--which are used by some of the AF's software factories--have grown the department's organic digital infrastructure capabilities via multiple IAC repositories and both on-prem and cloud-native environments serving all the services within the DoD. Furthermore, Platform One provides shared hardened containers and enables the distribution of approved code via a shared repository designated the Iron Bank (*Platform One*, n.d.).

3. The Services and OSD will need to create new paths for digital talent (especially internal talent).

Software factories, by nature of their design, could provide an avenue to meet the third area of effort as directed by the DIB. Specifically, the board's primary recommendation for action was to "Create software development units in each service consisting of military personnel…" (McQuade et al., 2019). In addition to cultivating talent through the widespread use of DevOps practices via the numerous software factories, the DoD was also directed via the FY20 NDAA section 230 to develop policy on its software talent management core competencies (House of Representatives, 2019). In response to this directive, the DoD created a formal software workforce working group tasked with the identification and creation of career tracks for software professionals. The most recent 2020 DIB assessment indicated that there was still a significant amount of work that needed to occur to meet the intent of the digital talent recommendations (*FY2020 NDAA DIB Assessment: Software Development and Software Acquisition Training and Management Programs*, 2020).

4. DoD and industry must change the practice of how software is procured and developed.

The final recommendation specifically targets the DoD's need to cultivate a positive culture change in order to drive the changes necessary to improve software development. In addition to providing technical guidance to meet the recommendations of the DIB, the CIO efforts under the DevSecOps initiative have also been focused on facilitating the framework necessary to drive these cultural changes. The board made

three initial recommendations which involved improving DoD access to source code and frameworks, making baked-in security a priority when developing software, and shifting to a feature-based as opposed to a requirements-based development process.

As discussed under the first line of effort, the new software acquisitions pathway takes a step back from the initial requirements-driven JCIDS documents and replaces them with the capability needs statement. The DoD DevSecOps initiative also addresses a feature-based approach and specifically addresses security as a major component of software factory development. In summary, the DIB report's recommendations attempted to identify the pre-conditions necessary to address existing shortfalls while still considering the unique needs of Department of Defense software acquisitions.

References

John Warner National Defense Authorization Act for Fiscal Year 2007, (2006) (testimony of 109th Congress). https://www.govinfo.gov/content/pkg/PLAW-109publ364/pdf/PLAW-109publ364.pdf

Adair, J. G. (1984). The Hawthorne effect: A reconsideration of the methodological artifact. *Journal of Applied Psychology*, *69*(2), 334–345. https://doi.org/10.1037/0021-9010.69.2.334

Allee, V. (2009). Value-creating networks: Organizational issues and challenges. *Learning Organization*, *16*(6), 427–442. https://doi.org/10.1108/09696470910993918

Assistant Secretary of Acquisition, & Air Force Chief Software Office. (2021). *Software Factories*. https://software.af.mil/software-factories/

Bastian, M., Heymann, S., & Jacomy, M. (2009). Gephi: An open source software for exploring and manipulating networks. *International AAAI Conference on Weblogs and Social Media*. https://gephi.org/users/publications/

Beck, K., Beedle, M., Bennekum, A., Cockburn, A., Cunningham, W., Fowler, M., Grenning, J., Highsmith, J., Hunt, A., Jeffries, R., Kern, J., Marick, B., Martin, R., Schwaber, K., Mellor, S., Sutherland, J., & Thomas, D. (n.d.). *The Agile Manifesto*.

Biem, A., & Caswell, N. (2008). A Value Network Model for Strategic Analysis. In *Proceedings of the 41st Annual Hawaii International Conference on System Sciences (HICSS 2008)*. https://doi.org/10.1109/HICSS.2008.43

Boehm, B. (2006). *A View of 20th and 21st Century Software Engineering*.

Boehm, B., Baker, T., Embry, W., Fox, J., Hilfinger, P., Holden, M., Moss, E., Royce, W., Scherlis, W., Taft, T., Vaughn, R., & Wasserman, A. (1997). Ada and Beyond: Software Policies for the Department of Defense. In *Ada and Beyond*. National Academies Press. https://doi.org/10.17226/5463

Boldi, P. (2020). Fine-Grained Network Analysis for Modern Software Ecosystems. *ACM Trans. Internet Technol*, *21*(1). https://doi.org/10.1145/3418209

Booz Allen. (1981). *Defense Automatic Data Processing Acquisition*.

Boucharas, V., Jansen, S., & Brinkkemper, S. (2009). *Formalizing Software Ecosystem Modeling*. http://softwareecosystems.com

Brass, D. J., Galaskiewicz, J., Greve, H. R., & Tsai, W. (2004). Taking Stock Of Networks And Organizations: A Multilevel Perspective. *Academy of Management Journal*, *47*(6), 795–817.

Bratman, H., & Court, T. (1975). The Software Factory. *Computer*, *8*, 28–37.

Buchsbaum, S. (1978). *Report of the Defense Science Board Task Force on Command and Control Systems Management*.

Carney, G., & Konwin, K. (2021a). Software Acquisition Pathway CoP News. *DoD Agile Acquisition Community of Practice*, *21*. https://federalnewsnetwork.com/defense-main/2021/05/pentagon-wants-to-use-its-biggest-it-

Carney, G., & Konwin, K. (2021b). Software Acquisition Pathway CoP News. In *DoD Agile Acquisition Community of Practice* (Issue 22).

Chaillan, N. (2019). *DoD Enterprise DevSecOps Reference Design*.

Chaillan, N. (2020). *DoD Enterprise DevSecOps Initiative*. https://software.af.mil/training/

Chiesi, A. M. (2015). Network Analysis. *International Encyclopedia of the Social & Behavioral Sciences: Second Edition*, 518–523. https://doi.org/10.1016/B978-0-08-097086-8.73055-8

Creswell, J. W., Hanson, W. E., Clark Plano, V. L., & Morales, A. (2007). Qualitative Research Designs: Selection and Implementation. *The Counseling Psychologist*, *35*(2), 236–264. https://doi.org/10.1177/0011000006287390

Cross, R., & Gray, P. (2021). Optimizing Return-to-Office Strategies with Organizational Network Analysis. *MIT Sloan Management Review*. https://sloanreview.mit.edu/article/optimizing-return-to-office-strategies-with-organizational-network-analysis/

Curtin, K. M. (2017). Network Analysis. *Comprehensive Geographic Information Systems*, *3*, 153–161. https://doi.org/10.1016/B978-0-12-409548-9.09599-3

Cusumano, M. A. (1989). *The Software Factory: A Historical Interpretation*.

Cusumano, M. A. (1991). Factory Concepts and Practices in Software Development. In *Annals Hist Comput* (Vol. 13).

DAU. (2022a). *A Guide to DoD Program Management Business Processes*. https://www.dau.edu/pdfviewer?Guidebooks/DAG/A-Guide-to-DoD-Program-Management-Business-Processes.pdf

DAU. (2022b). *A Guide to Program Management Knowledge, Skills and Practices*. https://www.dau.edu/pdfviewer?Guidebooks/DAG/A-Guide-to-Program-Management-Knowledge-Skills-and-Practices.pdf

Deasy, D., & Lord, E. (2020). *Software Development, Security, and Operations for Software Agility*. https://www.milsuite.mil/book/

Delauer, R., Boileau, O., Bridge, C., Campobasso, T., Fox, R., Gansler, J., Gates, H., Larsen, P., Lehmann, H., Livesay, M., Schroter, A., Shea, J., Smith, H., Sonenshein, N., Templeman, J., Webster, D., White, J., & Woll, H. (1974). *Report of the Task Force on Electronics Management*. https://dsb.cto.mil/reports/1970s/Electronics%20Management%2030%20April%201974.pdf

Department of Defense. (2021). *DevSecOps Fundamentals Playbook*. https://software.af.mil/wp-content/uploads/2021/05/DoD-Enterprise-DevSecOps-2.0-Playbook.pdf

Department of Defense Fiscal Year (FY) 2022 Budget Estimates Justification Book Volume 3b. (2021).

Department of Defense Fiscal Year (FY) 2022 Budget Estimates, RDT&E, Space Force. (2021).

Driessnack, J. (2017). *Program Manager Toolkit*. https://www.dau.edu/tools/t/Program-Manager-Toolkit

Druffel, L. (1982). *Strategy for a DOD Software Initiative*. https://apps.dtic.mil/sti/pdfs/ADA121737.pdf

Enos, J. R., & Nilchiani, R. (2018). *Using Social Network Analysis to Identify Systems of Systems in a Network of Systems*. https://doi.org/10.1109/SYSOSE.2018.8428791

Fowler, C. (1987). *Report of the Defense Science Board Task Force on Military Software.*

Freeman, L. C. (1977). A Set of Measures of Centrality Based on Betweenness. *Sociometry*, *40*(1), 35. https://doi.org/10.2307/3033543

FY2020 NDAA DIB Assessment: Software Development and Software Acquisition Training and Management Programs. (2020).

Gansler, J. (1975). Comment. *Defense Management Journal*, *11*(4). https://archive.org/details/sim_defense-management-journal_1975-10_11_4/page/n1/mode/2up

General Accounting Office. (1981). *Federal Agencies' Maintenance Of Computer Programs: Expensive And Undermanaged*. https://www.gao.gov/assets/afmd-81-25.pdf

Glaseman, S., & Davis, M. (1980). *Software Requirements for Embedded Computers*.

Gould, R. v, Fernandez, R. M., & Fernandezt, R. M. (1989). Structures of Mediation: A Formal Approach to Brokerage in Transaction Networks. *Source: Sociological Methodology*, *19*, 89–126.

Graff, C. (2013). Mixed Methods Research. In H. Hall & L. Roussel (Eds.), *Evidence-based practice: an integrative approach to research, administration, and practice*. Jones and Bartlett Learning.

Granovetter, M. S. (1973). The Strength of Weak Ties. *American Journal of Sociology*, *78*(6), 1360–1380. https://doi.org/10.1086/225469

Grimes, J. G. (2007). *Net-Centric Services Strategy*.

Guércio, H., Ströele, V., David, M. N., Braga, R., & Campos, F. (2018). Complex Network Analysis in a Software Ecosystem: Studying the Eclipse Community. In *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*. https://doi.org/10.1109/CSCWD.2018.8465170

Gutmann, R. (1980). *The Department Of Defense's Standardization Program For Military Computers-- A More Unified Effort Is Needed*. https://www.gao.gov/assets/lcd-80-69.pdf

Hansen, M., & Nesbit, R. (2000). *Defense Science Board Task Force on Defense Software*. https://dsb.cto.mil/reports/2000s/ADA385923.pdf

Hicks, K. (2022). *Department of Defense Software Modernization Strategy*. https://media.defense.gov/2022/Feb/03/2002932833/-1/-1/1/department-of-defense-software-modernization-strategy.pdf

House of Representatives. (2017). National Defense Authorization Act for Fiscal Year 2018. In *House of Representatives*. https://www.congress.gov/bill/115th-congress/house-bill/2810

House of Representatives. (2019). *National Defense Authorization Act for Fiscal Year 2020*. https://www.congress.gov/bill/116th-congress/senate-bill/1790

Ideker, T. (2003). Cytoscape: a software environment for integrated models of biomolecular interaction networks. *Genome Research*, *13*(11), 2498–2504.

Jansen, S., Brinkkemper, S., & Cusumano, M. (2013). *Software Ecosystems*. Edward Elgar Publishing. https://doi.org/10.4337/9781781955635

Jansen, S., Finkelstein, A., & Brinkkemper, S. (2009). *A sense of community: A research agenda for software ecosystems*. https://doi.org/10.1109/ICSE-COMPANION.2009.5070978

Jorgensen, D. L. (2015). Participant Observation. *Emerging Trends in the Social and Behavioral Sciences*, 1–15. https://doi.org/10.1002/9781118900772.ETRDS0247

Keller, R. (1977). *Problems Found with Government Acquisition and use of Computers from November 1965 to December 1976*. https://www.gao.gov/products/fgmsd-77-14

Kronenfeld, D. B. (2004). Cognitive Research Methods. *Encyclopedia of Social Measurement*, 361–374. https://doi.org/10.1016/B0-12-369398-5/00325-X

LaPlante, W., Wisnieff, R., Coleman, V., Nielsen, P., Lynch, C., Schneider, F., Markowitz, J., Thaer, L., Nesbit, R., & Velosa, A. (2018). *Design and Acquisition of Software for Defense Systems*.

Levine, S., & White, P. E. (1961). Exchange as a Conceptual Framework for the Study of Interorganizational Relationships. *Administrative Science Quarterly*, *5*(4), 583. https://doi.org/10.2307/2390622

Lord, E. M. (2020). *DOD INSTRUCTION 5000.87 OPERATION OF THE SOFTWARE ACQUISITION PATHWAY*. https://www.esd.whs.mil/DD/.

Maghssudipour, A., Lazzeretti, L., & Capone, F. (2020). The role of multiple ties in knowledge networks: Complementarity in the Montefalco wine cluster. *Industrial Marketing Management*, *90*, 667–678. https://doi.org/10.1016/j.indmarman.2020.03.021

McDonald, C. (1988). *Results of Follow-up 1983 SAB Report on Mission-Critical Software*. https://apps.dtic.mil/sti/citations/AD1048883

McDowell, T., Horn, H., & Witkowski, D. (2022). *Organizational Network Analysis Gain insight, drive smart*. https://www2.deloitte.com/us/en/pages/human-capital/articles/organizational-network-analysis.html

McMillan, B., Cairns, R., Galt, J., Mueller, G., O'Brien, B., Oder, F., Villard, O., & Ware, W. (1977). *Operational Software Management and Development for the USAF Computer Systems*. https://apps.dtic.mil/sti/pdfs/ADA069804.pdf

McQuade, M. J., Murray, R. M., Louie, G., Medin, M., Pahlka, J., & Stephens, T. (2019). *Software Is Never Done: Refactoring the Acquisition Code for Competitive Advantage*. https://media.defense.gov/2019/Mar/26/2002105909/-1/-1/0/swap.report_main.body.3.21.19.pdf

Munson, J. (1983). *The High Cost and Risk of Mission-Critical Software*.

National Research Council. (1989). *Adapting Software Development Policies to Modern Technology*. The National Academies Press. http://nap.naptionalacademies.org/19037

OSD. (2020). *Budget Activity (BA) "BA-08": Software and Digital Technology Pilot Program*. https://discover.dtic.mil/section-809-panel/,

Perkins, J., & Long, J. (2020). Software Wins Modern Wars. What the Air Force Learned from Doing the Kessel Run. *Modern War Institute*.

Perry, B. L., Pescosolido, B. A., & Borgatti, S. P. (2018). *Egocentric Network Analysis*. Cambridge University Press. https://doi.org/10.1017/9781316443255

Platform One. (n.d.). 2021. Retrieved November 29, 2021, from https://p1.dso.mil/#/

Pons, P., & Latapy, M. (2005). Computing communities in large networks using random walks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *3733 LNCS*, 284–293. https://doi.org/10.1007/11569596_31/COVER

Powell, W., Staw, B., & Cummings, L. (1990). Neither Market Nor Hierarchy: Network Forms of Organization. *Research in Organizational Behavior*, *12*, 295–336.

Ryan, Z. O., Reith, M. G., & Beach, P. M. (2022). Defining the DoD Software Factory: A Network value Approach. *Crosstalk*. https://community.apan.org/wg/crosstalk/m/documents/420788

Space CAMP. (2021a). https://spacecamp.il2.dso.mil/#/home

Space CAMP. (2021b). *Space CAMP Survival Guide 2.0*.

Tashakkori, A., & Creswell, J. W. (2007). The new era of mixed methods. *Journal of Mixed Methods Research*, *1*, 3–7.

Tashakkori, A., & Teddlie, C. (2003). Issues and dilemmas in teaching research methods courses in social and behavioral sciences: U.S. perspective. *International Journal of Social Research Methodology*, *6*, 61–77.

Tashakkori, A., & Teddlie, C. (2009). *Foundations of Mixed Methods Research*. Sage Publications, Inc.

USAF. (2021). *Department Of the Air Force Fy22 Budget Overview*. https://www.saffm.hq.af.mil/Portals/84/documents/FY22/SUPPORT_/FY22%20Budget%20Overview%20Book.pdf?ver=SMbMqD0tqIJNwq2Z0Q4yzA%3D%3D

Vick, C., Boehm, B., Davidson, J., Giese, C., McMillan, B., Martin, J., Miller, E., Mohler, S., Ramamoorthy, C., Urban, J., Ware, W., & Yeh, R. (1985). *Methods for Improving Software Quality and Life Cycle Cost*. http://www.nap.edu/catalog.php?record_id=19315

Vitto, V., Kerber, R., Guthrie, P., Hoeper, P., Kaminski, P., Lengerich, T., Longuemare, N., Maybury, M., Roca, R., Stenbit, J., & Wade, A. (2009). *Department of Defense Policies and Procedures for the Acquisition of Information Technology*.

Vorraber, W., Mueller, M., Voessner, S., & Slany, W. (2019). Analyzing and managing complex software ecosystems: A framework to understand value in information systems. *IEEE Software*, *36*(3), 55–60. https://doi.org/10.1109/MS.2018.290100810

Vorraber, W., Voessner, S., Ssner, S. V., & Vössner, S. (2011). *Modeling Endogenous Motivation and Exogenous Influences in Value Networks of Information Service Systems*. https://doi.org/10.4156/jcit.vol6.issue8.43

Walden, D., Muntner, M., Nehama, I., Hendricks, G., & Beam, D. (1978). *Software Management in the Air Force*.

Wasserman, S., & Faust, K. (1995). *Social Network Analysis: Methods and Applications*. Cambridge University Press.

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 074-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	
02-13-2023	Master's Thesis	Sep 2021 – Feb 2023

TITLE AND SUBTITLE

Illuminating the unknown: A mixed methods exploration of the DoD software factory ecosystem

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)

Ryan, Zachary O., Captain, USAF

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAMES(S) AND ADDRESS(S)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER

AFIT-ENV-MS-23-M-229

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Assistant Secretary of Acquisition, Chief Software Office
1060 Air Force Pentagon, Washington DC 20330-1060
EMAIL: Af.cso@us.af.mil
PHONE: 714-458-2303
ATTN: Maj Marvin Poquiz

10. SPONSOR/MONITOR'S ACRONYM(S)

DAF/CSO

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
DISTRUBTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES
This material is declared a work of the U.S. Government and is not subject to copyright protection in the United States.

14. ABSTRACT

The Department of Defense's (DoD) software factories are a collection of modern software acquisition programs that commonly employ the agile, network-based business strategies often found within commercial industries. Having been formally recognized by senior leaders for their revolutionary software development approaches, the software factories highlight a cultural shift within the DoD away from traditional organizational practices. As a result of the factories demonstrated successes, the number of programs employing non-traditional strategies is expanding. While this is notable, it also presents a challenge because a comprehensive understanding of the characteristics, structures, and behaviors of the DoD's software factories does not currently exist.

This thesis addresses this knowledge gap by employing a sequential mixed methods methodology to explore the organizational characteristics and structures of the DoD's software factories using a three-phased research approach designed to facilitate active community engagement and feedback. Primary research data was collected from the software factory community through personnel interviews, participant observation, and a case study. Results from this research include a software factory characterization framework, a structural definition of software factories, and a new programmatic assessment process designed to help acquisition practitioners understand the organizational behaviors of non-traditional programs.

15. SUBJECT TERMS
Inter-Organizational Behavior, Software Factories, Software Development, Agile, Network Analysis

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Mark G. Reith, AFIT/ENG
U	**U**	**U**	**UU**	131	**19b. TELEPHONE NUMBER** *(Include area code)* (937) 255-7777 Mark.reith@afit.edu

Standard Form 298 (Rev. 8-98)
Prescribed by ANSI Std. Z39-18