NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

THESIS

ACTIVITY MAPPING OF DEVELOPMENT METHODS AS A DECISION AID FOR HARDWARE DEVELOPMENT PROGRAMS

by

Aimee K. McCarthy

September 2021

Thesis Advisor: Clifford A. Whitcomb
Second Reader: Walter E. Owen

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE			Form Approved OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	September 2021	Master's thesis

4. TITLE AND SUBTITLE
ACTIVITY MAPPING OF DEVELOPMENT METHODS AS A DECISION AID FOR HARDWARE DEVELOPMENT PROGRAMS

5. FUNDING NUMBERS

6. AUTHOR(S) Aimee K. McCarthy

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Naval Postgraduate School
Monterey, CA 93943-5000

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
N/A

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government.

12a. DISTRIBUTION / AVAILABILITY STATEMENT
Approved for public release. Distribution is unlimited.

12b. DISTRIBUTION CODE
A

13. ABSTRACT (maximum 200 words)

Adoption of alternate development methods throughout engineering domains is resulting in more effective execution. However, implementation of such methods for hardware development programs has not been widely adopted or their processes well documented. To understand their potential for implementation, the generic life cycle defined in the *INCOSE Systems Engineering Handbook* is set as the baseline for comparison of development methods. This work analyzes six alternate development methods including design thinking, lean product development, agile, set-based concurrent engineering, systems thinking, and development and operations, focusing on their activities and overall flow to not only align them with the generic life cycle for visual representation of development progression, but more importantly, to map the similarities between the activities of the concept and development stages of the generic life cycle. Presented in this work is a mapping that compares the execution of the generic life-cycle baseline development activities to their occurrence in the six methods discussed. This mapping can be used as a decision aid within the hardware domain for determining the feasibility of an alternate method, giving programs a tailorable approach for hardware development.

14. SUBJECT TERMS
hardware development process, development methods, design thinking, human centered design, lean product development, agile, set-based concurrent engineering, systems thinking, development and operations, activities, mapping

15. NUMBER OF PAGES
93

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89)
Prescribed by ANSI Std. 239-18

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release. Distribution is unlimited.

ACTIVITY MAPPING OF DEVELOPMENT METHODS AS A DECISION AID FOR HARDWARE DEVELOPMENT PROGRAMS

Aimee K. McCarthy
Civilian, Department of the Navy
BS, Union College, 2014
MPS, Pennsylvania State University, 2019

Submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE IN SYSTEMS ENGINEERING MANAGEMENT

from the

NAVAL POSTGRADUATE SCHOOL
September 2021

Approved by: Clifford A. Whitcomb
Advisor

Walter E. Owen
Second Reader

Oleg A. Yakimenko
Chair, Department of Systems Engineering

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

Adoption of alternate development methods throughout engineering domains is resulting in more effective execution. However, implementation of such methods for hardware development programs has not been widely adopted or their processes well documented. To understand their potential for implementation, the generic life cycle defined in the *INCOSE Systems Engineering Handbook* is set as the baseline for comparison of development methods. This work analyzes six alternate development methods including design thinking, lean product development, agile, set-based concurrent engineering, systems thinking, and development and operations, focusing on their activities and overall flow to not only align them with the generic life cycle for visual representation of development progression, but more importantly, to map the similarities between the activities of the concept and development stages of the generic life cycle. Presented in this work is a mapping that compares the execution of the generic life-cycle baseline development activities to their occurrence in the six methods discussed. This mapping can be used as a decision aid within the hardware domain for determining the feasibility of an alternate method, giving programs a tailorable approach for hardware development.

THIS PAGE INTENTIONALLY LEFT BLANK

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF ACRONYMS AND ABBREVIATIONS

devops	development and operations
DOD	Department of Defense
INCOSE	the International Council on Systems Engineering
ISO/IEC	International Organization for Standardization/International Electrotechnical Commission
ISO/IEC/IEEE	International Organization for Standardization/International Electrotechnical Commission/Institute of Electrical and Electronics Engineers
LPD	lean product development
MVP	minimum viable product
NASA	National Aeronautics and Space Administration
SBCE	set-based concurrent engineering
SE	systems engineering
TR	technical report

THIS PAGE INTENTIONALLY LEFT BLANK

EXECUTIVE SUMMARY

The *INCOSE Systems Engineering Handbook* documents the generic life cycle as defined by ISO/IEC/IEEE 15288-2015. This standardized process, well understood throughout engineering development, suggests that a product can progress through its life cycle by means of serial or iterative processes, or a combination thereof (INCOSE 2015). Though, it does not provide guidance as to the method in which could be applied for that progression. Contrary to software or information technology services, implementation of development methods for hardware development are not well documented throughout engineering and Department of Defense industries. Furthermore, there is increasing pressure to adopt new practices to align with the DOD's mission of providing capabilities at the speed of relevancy (Garamone 2017; Ferdinando 2018). This paves the path of an immediate need for hardware development programs to have a foundational level of knowledge of existing development methods that would allow for effective program execution.

With a lack of guidance for applicability of development methods, this work analyzes the activities that occur during development given the implementation of an explicit method, and those defined by the generic life cycle for the concept and development stages as baseline for development activities. By means of literature review, the activities of six methods are compared to such baseline. The result is a high-level mapping of activities, in matrix form, that can be used as a decision aid, or as a general educational resource for method determination in hardware development programs.

This work analyzes development methodologies including design thinking, lean product development, agile, set-based concurrent engineering, systems thinking, and development and operations. For each, hardware development processes are presented based on the literature review and case study analyses that are used to describe the activities that occur in a specific stage or phase or the process in addition to high-level details, and any governing technical principles or requirements that are part of the process. The literature and case studies presented in this work are limiting in the sense that adoption of development methods specifically for hardware development has not been widely applied

or documented. For such methods that do not heave specific implementations for hardware development, more general applications are considered.

Analysis of the activities of which occur throughout hardware development show that no matter the method of execution there is similar intent, and the expected outcome can be similarly achieved. The results indicate that between different system engineering methods, many of the same activities occur, though a methods defining characteristics in addition to their problem-solving approach affect the execution of those activities.

Further, the results show that in many cases development methods are successful when utilized collectively. Particularly, development and operations implement lean principles and is in agile in structure (Kim et al. 2016), agile on its own may implement lean principles (Pries and Quigley 2010), and set-based concurrent engineering principles are inherently part of lean product development (Al-Ashaab et al. 2013). The only development methods discussed in this work that could be exclusively implemented are design thinking and systems thinking. Design thinking is a well-defined and well-documented process utilized mostly within the commercial industry, and systems thinking is a holistic approach to problem solving that is typically used from an organizational perspective. There is evidence that hard systems thinking, or systems dynamics can be used for product development, however current literature does not expand upon its implementation.

The mapping of activities presented in this work only represent a subset of all the activities that occur and does not map activity to activity. Additional work is needed to expand upon not only these activities, but to also identify other development. Furthermore, while having an awareness to development methods for determining an appropriate method for use, knowing the defining characteristics of a given method is also valuable in determining feasibility. Used in conjunction with an activity mapping would provide a more cohesive package for aid in decision making for development method implementation within an organization.

List of References

Al-Ashaab, Ahmed, Matic Golob, Usama M Attia, Muhammad Khan, Jon Parsons, Alberto Andino, Alejandro Perez, et al. 2013. "The Transformation of Product Development Process into Lean Environment Using Set-Based Concurrent Engineering: A Case Study from an Aerospace Industry." *Concurrent Engineering, Research and Applications* 21 (4): 268–85. https://doi.org/10.1177/1063293X13495220.

Ferdinando, Lisa. 2018. "DOD Must Be More Agile in Technology Development, Official Says." DOD NEWS. April 19, 2018. https://www.defense.gov/Explore/News/Article/Article/1497393/.

Garamone, Jim. 2017. "DOD Restructures Acquisition, Technology Office to Improve Military Lethality, Speed." DOD News Defense Media Activity. August 7, 2017. https://www.army.mil/article/191904/dod_restructures_acquisition_technology_office_to_improve_military_lethality_speed.

INCOSE. 2015. *Systems Engineering Handbook: A Guide for System Life-cycle Processes and Activities*. Hoboken, NJ: John Wiley & Sons.

Kim, Gene, Jez Humble, Patrick Debois, and John Willis. 2016. *The Devops Handbook How to Create World-Class Agility, Reliability, & Security in Technology Organizations*. Portland: IT Revolution Pres, LLC.

Pries, Kim H., and Jon M. Quigley. 2010. *Scrum Project Management*. Boca Raton, FL: CRC Press.

THIS PAGE INTENTIONALLY LEFT BLANK

ACKNOWLEDGMENTS

First and foremost, I would like to thank and recognize my husband, TJ, for supporting me through this journey. Navigating this program while becoming first-time parents has been challenging and the greatest blessing, and it was his positivity that continued to motivate me. I owe so much of my success to him.

Next, I'd like to thank my advisor, Distinguished Professor Cliff Whitcomb. His guidance allowed me to progress through this thesis with a sense of defined creativity. More importantly, his expertise in methods and development processes in general was critical to the success of this work.

I must also show my gratitude to my PD-21 cohort members for the constant, never-ending thesis motivation. No matter the time of day or place, I could always count on them for guidance, point-directed help, and uplifting humor. It has been a pleasure to learn from you all, and I wish you all the best in your careers.

Finally, this thesis would not have been possible without the encouragement of my supervisor Dan O'Connell and those I work with at PMO SPSP. I so appreciate the flexibility provided to attend this program and complete this thesis. I look forward to sharing it with SSP as we continue to explore and adopt new methods within our development efforts.

THIS PAGE INTENTIONALLY LEFT BLANK

I. INTRODUCTION

A. BACKGROUND

In 2018, Michael D. Griffin went before the House Armed Services Committee for a special hearing for the Department of Defense's (DOD) culture of innovation and delivered a powerful testimony. He emphasized how critical it is that the DOD pursue new technologies and breakthrough research to preserve its technological advancements (Ferdinando 2018). Ferdinando also noted that today's adversaries systematically and strategically develop and field advanced systems more rapidly than the United States. Griffin added that it is a priority for the DOD to drive the innovation cycle to sustain technological superiority by developing innovative capabilities in addition to being innovative in our processes (Ferdinando 2018). The typical waterfall development process cannot keep pace in aligning with this vision (Moore 2021). A sequential approach does offer benefits including predictability, stability, repeatability, and high assurance throughout the development process, though it has slow accommodation to change and lack of adaptability and flexibility, and longer delivery times due to its serial nature (Smartsheet n.d.). Sequential approaches are best used in a development program that is incredibly rigid where tasks and deadlines are set and maintained (Smartsheet n.d.), and not conducive in an environment where change is inevitable, and programs must adapt and remain flexible to stakeholder needs and required changes. To be successful and rapidly deploy capabilities to the warfighter, the DOD must "move at the speed of relevancy" (Garamone 2017). Deputy Director Secretary Pat Shanahan perfectly explained this when he said, "baseball doesn't get a phenomenal 17-year-old player and finally get him to the major leagues when he is 45, and neither should the DOD" (Garamone 2017). Prompting the DOD into action, it now faces the challenge of developing hardware more effectively and by use of innovative processes.

Consider the development buzzword "agile" that has quickly become a development standard in software, and "development and operations" likewise for information technology applications. These development methods aim to develop software and related services to their users quicker and with the same fidelity otherwise achieved

using waterfall development methods. For hardware development however, there is a lack of literature and case studies that define similar processes to streamline development efforts resulting in faster deliveries of hardware products. The DOD has a need to execute hardware development more rapidly, given that the status quo for development has been serial or sequential in nature. This work, then considers a need to provide program managers, designer, engineers, and organizations with knowledge of not only existing development methods and processes that are available for implementation, but also the activities which occur as part of those processes. Provided as a resource, such information could be used to help programs tailor specifically to the activities in which they need to complete to meet their objective, without sacrificing quality. Specifically, this work will compare existing development methods by analyzing main activities, phases or process flows against a defined, baseline hardware development life cycle. Throughout this work, the development methods discussed and analyzed will be referred to as methods.

B. GENERIC LIFE-CYCLE

This work uses the generic life-cycle model defined in the *International Council on Systems Engineering (INCOSE) Systems Engineering (SE) Handbook* as a baseline for hardware development stages and process. The scope of this work considers the concept and development life-cycle stages because they involve the physical design and development of a hardware product. The focus throughout this work is strictly on the design activities from concept, through development up until, but not including production readiness.

1. Life-Cycle Model

According to the International Organization for Standardization/International Electrotechnical Commission (ISO/IEC) Technical Report (TR), ISO/IEC TD 24748-1, a "system progresses through a common set of life cycle stages" including concept, development, production, utilization, support, and retirement (INCOSE 2015, 29). The life-cycle model represents the stages that the product will progress through to ensure it meets the intended functionality through its life, shown in Figure 1. The stages are shown in general sequential order, though stages in practice can be interdependent, overlapping, and

concurrent (INCOSE 2015). Possible progressions of a product through its life cycle may include sequential, iteration and recursion, and incremental and iterative methods (INCOSE 2015).

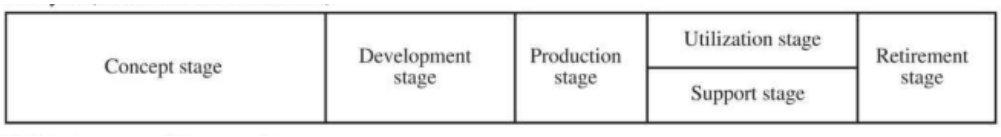

Figure 1. Standardization/International Electrotechnical Commission/ Institute of Electrical and Electronics Engineers (ISO/IEC/IEEE) 15288:2015 Generic Life-Cycle Stages. Source: INCOSE (2015).

The purpose of each phase is further delineated in the *INCOSE SE Handbook* and is shown in Table 1. For example, the concept stage is more extensive than others due to the amount of up-front work required including defining the problem space, characterizing the solution space, and identifying stakeholder needs, to name a few. Without fulfilling that purpose, rework may be required later during the development process resulting in cost and schedule impacts, or the result is a system that does not meet the needs of the stakeholders.

Table 1. Generic Life-Cycle Stage Purpose. Source: INCOSE (2015).

Life-Cycle Stages	Purpose
Concept	Define Problem Space • Exploratory Research • Concept Selection Characterize solution space Identify stakeholders' needs Explore ideas and technologies Refine stakeholders' needs Explore feasible concepts Propose viable solutions
Development	Define/refine system requirements Create solution description-architecture and design Implement initial system

2. Life-Cycle Stage Description

a. Concept Stage

The *INCOSE SE Handbook* breaks down the concept stage into two parts: exploratory research and concept selection. During exploratory research, the team develops a high-level preliminary concept to a depth that allows them to "identify technological risks and assess the technology readiness level of the project" (INCOSE 2015, 29). Ulrich and Eppinger (2016) simply suggest these phases investigate the feasibility of product concepts where a preliminary concept "is a description of the form, function, and features of a product accompanied by specifications" and technical documentation, "analysis of competitive products, and economic justification" (15). Key activities of exploratory research include defining the problem space, identifying mission requirements and stakeholder needs, establishing target specifications, and providing "an estimate of the cost and schedule for full-scale development" (INCOSE 2015, 30).

During concept selection, the team analyzes multiple candidate solutions (concepts) and presents justification for the selected candidate based upon refined stakeholder needs and concept of operations (INCOSE 2015; Ulrich and Eppinger 2016). This effort can include prototyping or building mock-ups and developing models and simulations. This would allow the team to perform architectural tradeoffs and explore risks and opportunities (Ulrich and Eppinger 2016). The concepts are tested to show preliminary results again critical stakeholder needs and scored for overall adherence to stakeholders needs and requirements. Key activities of concept selection include creating the functional definition of the system, defining the architecture, and planning for integration, verification, and validation testing (INCOSE 2015; Ulrich and Eppinger 2016).

b. Development Stage

During development, the team further defines a product of interest that will ultimately be produced. Development should not start until a concept, or concepts are chosen or down selected into this phase. In this stage, stakeholder needs, and requirements are formally defined and used to develop the system architecture (INCOSE 2015). One key activity during development "is to specify, analyze, architect, and design the system so that

the system elements and their interfaces" are understood and specified, and testable (INCOSE 2015, 31). Operator or end user interfaces are specified, tested, and evaluated. Feedback is provided by stakeholders through technical reviews and decision gates for information awareness and approval to proceed (INCOSE 2015). As part Ulrich and Eppinger's (2016) generic product development process, development is broken into two major phases, system-level design, and detail design, and supported by a testing and refinement stage. The activities accomplished throughout the development stage as a whole or broken in multiple phases are the same (Ulrich and Eppinger 2016). For example, in Ulrich and Eppinger's (2016) generic product development process, system level design feeds the design of the product, while detail design focuses on documenting the system configuration, ensuring quality assurance processes are in place, and placing early procurement orders. They further define a testing and refinement phase which covers system testing for overall performance, reliability, and durability in addition to prepping for production (Ulrich and Eppinger 2016). It is important to recognize that the generic life-cycle stage of development defined in the *INCOSE SE Handbook* serves as a large umbrella for many design activities, or phased activities. Nonetheless at the end of the development stage, is a decision gate requesting approval to move into production or agreement that the system has met is production readiness criteria (INCOSE 2016).

C. RESEARCH SCOPE

In support of the DOD's mission to move at the speed of relevancy and the desire to be innovative within the development process, this work proposes a mapping of methods against the generic life cycle defined by the *INCOSE Systems Engineering Handbook* that would allow programs to choose development phases and activities specifically tailored to their program. How a program implements the chosen method(s) to fit within their organization and their schedule is not proposed as part of this work. Rather, this mapping will align the stages, activities, and major characteristics of the generic life cycle and methods identified via literature research and review.

The methods chosen for analysis were based on their applicability in solving problems from a systems perspective, their implementation into software or IT services

development, or their implementation into hardware development. Even if one method lacks the literature supporting application for hardware development, its activities and governing principles or characteristics could be translated more generically and applied in a hardware development environment.

1. Research Questions

The following research questions are analyzed:

- How do the identified development methods align with generic life-cycle process within concept and development phases as defined by the *INCOSE SE Handbook*?
- What are major phases and activities of the identified methods?
- Have these methods been executed within hardware development projects/ programs? If not, are there implementations of the given method that may support hardware development?

2. Research Approach

The research for this work will begin by analyzing six different methods including:

- Design Thinking
- Systems Thinking
- Lean
- Agile
- Set-Based Concurrent Engineering
- Development and Operations

Each method in the context of understanding its phases or effect on the identified generic life-cycle stages, will be described. Case studies or implementations of each

method will be further analyzed to better understand the method as it pertains to hardware development.

D. PURPOSE

There are two opportunities in which this work can contribute. First, organizations need to find a way to make their existing processes innovative and more effective. To do so, organizations should consider employing other methods in executing their hardware development programs. For example, the DOD Acquisition Framework has been modified with more flexible pathways so programs can execute as they see fit. This flexibility is lost if programs, including its design teams, continue to execute in a serial nature, or as Moore (2021) refers to as waterfall. Programs need a way to compare how their hardware development process or how a generic hardware development process aligns to others so that they can tailor their program accordingly.

The *INCOSE SE Handbook* offers a comparison of the generic life-cycle stages to other life-cycle viewpoints including DOD acquisition, National Aeronautics and Space Administration (NASA), and the typical high-tech commercial systems integrator, to name a few. It also offers three ways in which the generic life-cycle stages can progress through a life cycle. It does not, however, describe what these methods are and how they fit within the generic life cycle. Likewise, there is an opportunity to provide some foundational knowledge of existing development processes and methodologies to hardware designers and engineers who may not otherwise know exist. Doing so will work to close the gap by providing designers and engineers material and resources to understand how different methods, that for example may be sequential or iterative in nature, fit within a product or hardware life cycle. There is no mapping that is readily available for programs to use as a decision aid, or as general guidance. Furthermore, there is no available descriptive resources that provide this information.

The information herein will provide the ability for programs to understand how each method aligns with the generic life-cycle process, so programs can tailor their own product development process to best fit its teams, its schedule, and the value it seeks to provide the customer and end user based on its activities. The methods in this work were

chosen because they aim to provide a new pathway for hardware development, or when applied to current development processes, enhance the product life cycle. Understanding how the phases of a given method are executed and what activities occur aid in the process of identifying whether such method would be beneficial within their program.

E. DELIVERABLES

This work will deliver high-level descriptive figures for each of the methods described, and a comparative alignment mapping to visually represent how each method aligns to the generic life cycle as defined in the *INCOSE SE Handbook.* More importantly, an activity mapping is created based on literature and case study reviews that identify specific activities that are compared to the *INCOSE SE Handbook* generic life cycle. These can be used separately or collectively as decision aids or guiding tools for identifying one or more methods for implementation. The information provided as part of these deliverables is based on literature research identified in this work and is not wholly inclusive of all literature.

F. ORGANIZATION OF STUDY

This work is organized in five chapters. Chapter 1 provides the necessary background information and lays the road map for the remainder of this work. Chapter II presents a review of literature and other resources providing descriptive summaries and activities for the given methods. Chapter III reviews and discusses literature for specific applications of the given development methods related to hardware development and discusses implementation characteristics for each. Chapter IV contains the synthetization of data gather via literature reviews and provides decision aids in the form of descriptive figures along with mapping of activities for each identified development method. Chapter V includes the final conclusions, recommendations, and future work.

II. DEVELOPMENT METHODS OVERVIEW

This chapter presents key information for methods considered for this work. Literature research resulted in the analysis of six approaches or methodologies to be considered based on the criteria that they have or could be utilized in product or system development from either software, hardware, or human-centered perspectives. Each method is described in brief, and is not inclusive of all characteristics, details, and nuances. The six methods are:

A. Design Thinking/Human Centered Design
B. Lean Product Development
C. Agile
D. Set-Based Concurrent Engineering
E. Systems Thinking
F. Development and Operations

A. DESIGN THINKING

Design thinking for purposes of this work is inclusive of "true" design thinking (as defined by Stanford University) and human centered design methodologies. Design thinking is an innovative approach to solving problems and suggesting many solutions, while human centered design approach revolves around the human experience (DiMeo 2018). The term "design thinking" will be referenced herein with the intent of describing both design thinking and human centered design.

This approach creative in nature and is defined by Wrigley, Nusem, and Straker (2020) as the ability to solve problems by using cognitive processes to identify and address stakeholder needs. Schallmo, Williams, and Lang (2018) describe the four principles of design thinking as: "human needs, multidisciplinary teams, iterative processes, and creative environments" (12). Recently, researchers have shown design thinking is linked to higher levels of innovation and is more prominent in the fields of design and development (Linke 2017). First utilized by IDEO, a design team was challenged to think

beyond just designing a product and instead, focus on the process in which the designer uses during the process (Björgvinsson, Ehn, and Hillgren 2012). IDEO suggests that,

> (1) designers should be more involved in the big picture of socially innovative design, beyond the economic bottom line, (2) that design is a collaborative effort where the design process is spread among diverse participating stakeholders and competencies, (3) that ideas must be envisioned, "prototyped," and explored in a hands-on way, tried out early in the design process in ways characterized by human centeredness empathy and optimism. (Björgvinsson, Ehn, and Hillgren 2012)

Even those who are not necessarily trained as designers can apply their human experience to the design process, providing for more creative potential solutions (What is Design Thinking n.d.).

There is a phased approach to implementing design thinking. While there are many processes in which define design thinking, Tim Brown's (once CEO of IDEO) process uses different methods including brainstorming, observations, and sketching to iteratively execute through the phases of design thinking (Wrigley, Nusem, and Straker 2020). Researchers agree that the phases of design thinking include empathize, define, ideate, prototype and test. Mueller-Roterberg (2018) further delineates these phases into two major categories, analytic and synthetic. The analytical phases fall within the problem space and include collecting, organizing, and evaluating information, while the synthetic phases fall within the solution space and include developing, testing, and improving solutions (i.e., concept and development activities) (Mueller-Roterberg 2018).

During the empathize phase, the team develops a deep understanding of the challenge. This involves getting to know the customer, and not assuming what someone is thinking or feeling. Gathering information is a critical piece of this phase, and can be done via interviews, conversations, and observations of the target audience (IDEO n.d.). According to the Institute of Design at Stanford, to empathize, an individual must observe in a context relevant to the issue, engage in conversations always looking for deeper meaning, and be observant of the user in their environment and how they engage (Hasso Plattner Institute of Design 2019). Linke (2017) suggests the development team try to empathize with the target audience to understand their needs and form potential solutions.

Then the process transitions into the define phase where the team clearly articulates the needs based on their information gathering of the target audience and uses the information to characterize issues and needs, into a well-defined problem statement (Linke 2017; Mueller-Roterberg 2018). Creating a specific point of view will also help designers to express their insights and create an actionable problem statement (Hasso Plattner Institute of Design 2019). Next in the ideate phase, both Linke (2017) and Mueller-Roterberg (2018) agree that the team engage in creativity techniques to identify potential inventive or innovative arrangements or solutions where the quantity of solutions is more important than few quality solutions. The Hasso Plattner School of Design indicates the importance that the team go beyond what is obvious and explore all possibilities (2019). Once potential solutions are identified, the team transitions into the prototype phase where they create and build prototypes that differ in form and fidelity that will be tested with real user prior to releasing to the market (Linke 2017). In the test phase, according to Linke's (2017) description, the prototypes are then tested amongst an audience to determine whether the product functions or operates as it is supposed to. During test, the point of view is continuously refined, and designers are encouraged to learn more about their users (Hasso Plattner Institute of Design 2019). Finally, Linke (2017) indicates feedback is provided to the team, and modifications or updates are made to the design and retested amongst an audience.

While well understood, there is more than one approach to design thinking. According to Mueller-Roterberg (2018) design thinking is an iterative process consisting of six process steps including "observing, defining problems, finding ideas" (ideate), "developing prototypes, and testing." (Mueller-Roterberg 2018, 10). The first three are within what Lindberg et al. (2010) and Mueller-Roterberg (2018) call the problem space, while the latter three are within the solution space. In the problem space, the team identifies, clarifies, analyzes, and understands the problem and in the solution space the team evaluates and selects ideas (Lindberg et al. 2010; Mueller-Roterberg 2018). They agree that design teams should utilize different techniques to create prototypes that are then tested and analyzed against the problem space. Compared to Brown's design thinking process, the major difference is how the empathize phase is broken into understanding the problem

and observing, and the define phase also includes observing and defining the problem (Lindberg et al. 2010). The overlap in observing is because there is a level of empathizing and understanding the problem from the customer's view in addition to collecting information from the customer. In Brown's method, these actions are split between empathize and define. The design thinking model presented by Plattner et al. (2011) is categorized as a very comprehensive, user-oriented approach that applies human centered design techniques within a phase in a process with iteration loops. Further, the model proposed by Plattner et al. (2011) may have more phases than Brown's original and pioneering design thinking process, but the techniques and the inner workings of the teams follow the same paradigm.

Schallmo, Williams, and Lang (2018) propose a roadmap for design thinking that include the following steps: defining the design challenge, understanding the design challenge, defining perspectives, gaining ideas, developing prototypes, testing prototypes, and integrating prototypes. This roadmap is a culmination of existing design thinking theories including those of Platter et al. (2011) that map the activities from start to finish to fill the gap of innovation management (Schallmo Williams and Lang 2018). It further outlines the objectives and outcomes for each phase. Step two through step five when considered based on their activities, align with other literature presented in this section. The additional phases closely align to business and management practices rather than development and are not considered for analysis.

B. LEAN PRODUCT DEVELOPMENT

Lean product development (LPD) has adapted over many years, originating from Toyota's production system (TPS) (Khan et al. 2013). Coined and created by Taiichi Ohno, TPS was built to increase value and reduce waste, while offering continuous improvement. Gaining traction through the 1990s, TPS has become industry standard and the much of the world has converted to lean production. The components of lean production, for example Kaizen and Gemba, amongst others, are instrumental to the success of executing a lean production program. While these components were designed for lean production, much research shows the intent can be applied to lean product development. Further, the

foundational characteristics or principles, also considered as a thought process, of lean production should be applied wholly to lean product development (Mynott 2012). These include understanding value, value stream mapping, flow, pull, and continuous improvement. Each of these principles is adapted to be specific to product development:

- Principle #1 – Precisely specify the *value* of a new product.
- Principle #2 – Identify the *value stream* for creating the new product.
- Principle #3 – Allow value to *flow* without interruptions.
- Principle #4 – Let the customer *pull* value from the development team.
- Principle #5 – Continuously pursue (economical) *perfection* (Mascitelli 2006, 15).

LPD in one form can be the result of applying such principles to a given, already defined, development process where the tasks and activities are now part of a value stream that establishes a leaner product development process that aligns closely with the customers' expectations and needs. Further, it provides in the ability to deliver products to market faster, by maximizing customer value and minimizing waste. By implementing LPD, a team can better predict schedule and eliminate redesign activities that cause delays, develop products in a shorter time-period, spend less time on activities and tasks that add no value, lower costs through the total product life cycle, lessen the uncertainty within the design and development process, and meet the needs of the customer or end user more completely (Radeka 2013).

Minimizing waste, per Radeka (2013), allows product developers to focus on their tasks and prevents them from being interrupted by repetitive status meetings, excess documentation, or task balancing from participating in other programs. Mascitelli (2006) proposes his "top-ten" sources of product development waste in Figure 2.

Figure 2. Top-Ten Sources of Product Development Waste. Source: Mascitelli (2006).

Mynott (2012) indicates time should be taken to purposefully remove or minimize waste throughout the entirety of the development process and the team should be aware of ways in which they can add value and incorporate such value along the way. His main point relating to the identification of waste as part of lean product development, is that any part of a process can be removed if it does not add value to the customer or the process, including product development. He further suggests the hardest part of identifying waste in product development is that most of it in invisible.

Development teams should aim to ensure value adding "activities that build knowledge about customers, activities that build knowledge about our product technology, activities that iterate customer and technical knowledge into product that we can produce, and customers want to buy" (Radeka 2013, 19). Value stream mapping while a beneficial tool, will not produce a process with zero waste until an organization has worked through the process multiple times (Mynott 2012). Even then, no development project is the same, so some waste is simply inevitable.

From a startup perspective, Ries (2011) in his book *The Lean Startup* proposes a three-phase process consisting of a build-measure-learn feedback loop. This process is not specific to one-kind of development but aims to determine whether a product should be developed and whether an organization can sustain itself around that product (Ries 2011). The most significant component of Ries' proposed three-phased process is the creation of a minimum viable product, also known as an MVP. He describes an MVP as a version of a product that delivers a defined minimum capability or functionality that allows the team to continue in the process and have some measurable impact.

C. AGILE

Agile development is formed via several different methods including scrum, feature driven development, and extreme programming, amongst others. This section presents scrum as an implementation of agile because it is the most used method for software development and is also suitable for any project-based work including hardware development (Capers 2018; Cooke 2012). In its purest form, scrum provides a structure for completing development activities or a given development project in an agile manner. Other tools, including utilizing lean principles, can be implemented to identify value streams that focus the development efforts and allows for organization of tasks into sprints that deliver a feature in a short amount of time (Pries and Quigley 2010).

Measey et al. (2015) defines 12 principles of agile that should absolutely make a program successful. Utilizing scrum is a way that development tempo is increased, in addition to team responsiveness and communications, and reducing overall risk (Pries and Quigley 2010). Pries and Quigley (2010) mention when using the scrum approach, products are improved over time and delivered to the customer for use, at which time feedback is provided back to the development team. They also state the major "principle of Scrum is that frequent repetition allows for changes that arise during product development" (15). Furthermore, they describe agile as an iterative process where there are continuous opportunities for feedback, and as a deliverable driven development with every iteration.

Compared to a serial development process, agile focuses on delivering one feature in a short amount of time, as quickly as a few days or weeks (Capers 2018). Rather than moving serially through the process, each phase of the process is executed to some extent for each feature or capability being developed through a series of sprints (Pries and Quigley 2010). At any given time, the team can be writing documentation, designing/developing, or testing and as the team progresses, work becomes more detailed as a product is realized (Pries and Quigley 2010).

When using scrum, high level requirements are defined in a work breakdown structure early in the process, with a lesser level of detail (Vanderjack 2015). Use cases are created in the beginning to streamline and capture functional requirements and provide a means for customers to be involved and engaged (Pries and Quigley 2010). The use case method, also known as user stories, provides a minimum documentation level for the team to execute (Walsh and Mahesh 2015; Pries and Quigley 2010). These user stories become the product features that are ultimately delivered to the customer (Walsh and Mahesh 2015; Pries and Quigley 2010). Given the user stories are created, they are set into a backlog to which the team pulls tasks from and are executed in a sprint (Walsh and Mahesh 2015; Vanderjack 2015). Tasks can include concept development activities for example developing a code feature, updating a requirements specification. In software applications code features are tested and released, and as the team progresses and iterates through sprints additional design details and functionality defined as well as execution of verification activities including documentation creation and updates (Pries and Quigley 2010).

Consider an iteration cycle. According to Vanderjack (2015), once the user stories are created the team completes iteration and sprint planning, and begins choosing the tasks to work or features to develop. Next the team begins their build or design activities, from a software development perspective this is when the detail design activities occur including initial unit/specific test (Vanderjack 2015). Vanderjack (2015) then explains that the team goes one step further in its testing that would include system test and regression testing. Once test activities are done, he indicates the team will declare the feature to be complete and test cases to be successfully addressed and reports delivered. At this point the team continues to iterate back through development activities or if ready prepares for production.

D. SET-BASED CONCURRENT ENGINEERING

Set-based concurrent engineering (SBCE) is a method that Toyota implemented as part of its product development system. While Toyota follows a lean product development process, it applies lean principles in addition to set-based concurrent methodology for its overall product development execution (Sobek et al. 1999). SBCE is included in this work as a methodology on its own, rather than solely a part of the Toyota product development process. SBCE uses these principles to progress through a development process.

According to Sobek et al. (1999) SBCE has three principles, and associated with each are three stages in which work is achieved, described as follows (73):

1. Map the Design Space
    a. Define Feasible Regions
    b. Explore Trade-Off by Designing Multiple Alternatives
    c. Communicate Sets of Possibilities
2. Integrate by Intersection
    a. Look for the Intersection of Feasible Sets
    b. Impose Minimum Constraint
    c. Seek Conceptual Robustness
3. Establish Feasibility before Commitment
    a. Narrow Sets Gradually while Increasing Detail
    b. Stay within Sets Once Committed
    c. Control by Managing Uncertainty at Process Gates

By contrast, Khan et al., identifies five major categories each with a set of principles that meet the category objectives. These five categories include (1) strategic value research and alignment, (2) map the design space, (3) create and explore multiple concepts in parallel (4) integrate by intersection, (5) establish feasibility before commitment (Khan et al. 2011, 3). In relation to concept and design/development of a product, Khan's first category aligns more with planning prior to execution of development, the latter four however, could be applied to development activities. Generally, the principles proposed by Sobek et al. (1999) and categories proposed by Khan et al. (2011) agree on the fundamental characteristics and activities of SBCE.

To execute SBCE, the team starts with mapping the design space, where they define the design bounds, explore trade spaces of multiple candidate solutions, and distributes sets

of possibilities to its team and its stakeholders (Khan 2011). Khan (2011) describes those feasible regions are defined by using checklists or design standards that detail design guidelines and best practices that help to define the initial conditions or defines them as subsystem targets. According to Sobek et al. (1999), in this phase, the team explores tradeoffs by designing multiple alternatives via simulations or prototypes, and then defines evaluation criteria or communicates acceptance criteria for those alternatives. Instead, Khan et al. (2011) defines the concept development as a separate phase where sets of design concepts are created and sets for each subsystem are defined. Sobek et al. (2011) suggests prototypes are created that fit within the defined sets, tested against targets, and then analyzed for further feasibility to which they are then communicated to the team. Further, Sobek et al. (1999) and Khan et al. (2011) agree most activities tend to result in some overlap between phases, however the design is narrowed over time as the project matures.

Khan et al. (2011) defines the next phase as concept convergence and Sobek et al. (1999) as integrating by intersection. Both are consistent in the activities relating to determine where set intersections occur and that intersections identify solutions to a set. Feasible sets are identified based on analysis and critiques of a design (Khan et al. 2011; Sobek et al. 1999). Sobek et al. 1999 also suggests that minimum constraints be defined to provide some level of flexibility in exploring designs or improve integration, rather than being locked into solution early on, and such constraints also allow for a design to be functional regardless of physical variations. They emphasize that when engineering functions create designs that work well with all possibilities in other function sets, then it can be further developed without needing any more information. Both Khan et al. (2011) and Sobek et al. (1999) agree that as the sets begin to converge the team uses more detailed models and designs, that can be further narrowed via testing. For example, testing can occur as part of concept prototyping, while feasible sets are defined, and when the possibilities are converging (Khan et al. 2011). Testing as part of detailed design verifies and validates the design in accordance with the final specification (Khan et al. 2011; Sobek et al. 1999).

Sobek et al. (1999) and Khan et al. (2011) also describe the last portion of their proposed process and activities are devoted to the detailed design of the development effort where the final design occurs from many possibilities converging gradually into one

design. Preliminary specifications become more defined, and each involved function agree on design sets that results in convergence to a solution (Sobek et al. 1999). The final design specification is released, including the final set of requirements, interfaces, and standards, and is managed through process gates that can include design reviews, early procurement of long lead parts, provisioning, and project milestones (Sobek et al. 1999).

E. SYSTEMS THINKING

Existing literature lacks a focus in understanding how systems thinking applies to new development or development programs. Rather, the literature focuses on systems theory as it applies to systems thinking and how systems thinking is utilized from an organizational and business practice (Galli 2018). Arnold and Wade (2015) argue that there are many definitions of what systems thinking is or encompasses and the term systems thinking is more abstract in nature is applied under different system theories. Peter Senge (2006) takes a business context perspective and describes systems thinking as the fifth discipline and argues that it is the cornerstone of the learning organization. Senge believes that systems education only allows people to breakdown pieces into smaller parts making complex tasks manageable, but connections within the system and the big picture are lost. In his book, he describes the necessary mind shift from linear thinking to systems thinking. More importantly he argues systems thinking requires one to see whole systems that have inter-linkages and causal relations, processes that change overtime, and leverage points or systemic behaviors. Peter Checkland, another pioneer of systems thinking, spent a large part of his career identifying the difference between a hard system and a soft system, and the methodologies of problem within each (Stowell 2013). Specifically, Checkland came up with what is now known as soft systems methodology, to which systems thinking applies the best when applied to real world problems (Ramage et al. 2020).

More generally, systems thinking can be used to solve existing problems, finding fast tracks for improvements, and preventing possible future problems (Rutherford 2019). According to Senge (2006), people tend to see reality linearly, rather applying systems thinking results in the ability to see circles of causality where there are inter-relationships and constant change. Many of the systems thinking experts agree that systems are

interconnected and former approaches to solving complex problems by breaking a system down into smaller manageable elements is no longer sufficient (Goede 2015; Senge 2006; Rutherford 2019).

Putting systems thinking more in the context of problem solving, when an emergent issue or failure presents, it can be linked to or categorized as a system archetype, or as a pattern of behavior (Senge 2006; Rutherford 2019). Rutherford (2019) defines an archetype as "commonly repeating variation of reinforcing and balancing feedback" where "each archetype has a typical pattern of behavior overtime, structure, and effective interventions" (92). He further describes that a system archetype gives structure to the systems thinking process, as it provides a method for investigations when analyzing errors in a system that have either occurred or can be predicted.

Arnold and Wade (2015) propose that through the many different definitions of systems thinking there is not one that is fully correct. Though their analysis shows that many authors agree on the fundamentals of systems thinking, they indicated the definition provided by Sweeney and Sterman (2000) relating to education is the most useful because it outlines six necessary skills to execute in any scenario, including an approach to product development or for troubleshooting failures as part of evaluating prototypes or system level test. Their study was based on understanding the inventory of systems thinking concepts including feedback, delays, and stocks and flows. Arnold and Wade (2015) propose systems thinking steps that include (672):

- Understand how the behavior of a system arises from the interaction of its agents over time (i.e., dynamic complexity)
- Discover and represent feedback processes (both positive and negative) hypothesized to underlie observed patterns of system behavior
- Identify stock and flow relationships
- Recognize delays and understand their impact
- Identify nonlinearities
- Recognize and challenge the boundaries of mental (and formal) models

Sweeney and Sterman (2000) argue that an inventory of systems thinking concepts will broaden people's ability to understand dynamic complexity of systems and be able to apply them to everyday reasoning. Senge (2006) also agrees that we need perspective on

problems, and some insight as to how we would do things differently. Systems thinking therefore takes shape via an inventory of concepts, proposed steps, and necessary skills that can be applied to hardware development.

Going one step further, Maani and Cavana (2007) propose a systems thinking and modeling process consisting of five phases including problem structuring, causal loop modeling, dynamic modeling, scenario planning and modeling, and organizational implementation. The casual loop modeling portion of this process focuses on showing the influences within a system, while dynamic modeling is used to model complex processes (Tsuchida and Jones 2019). Maani and Cavana (2007) suggest the focus of this proposed systems thinking process is the modeling aspect, and more specifically the ability to intervene within the problem space. This process is considered to align with a hard systems thinking methodology as it veers away from the soft systems methodology approach from Checkland that is more conceptual in nature (Maani and Cavana 2007).

F. DEVELOPMENT AND OPERATIONS

Since Development and Operations (devops) is a combination of different philosophies, tools, and practices there is a gap in literature describing design activities as they flow through the process; more specifically as it relates to hardware. Typical devops applications are accepted for software or IT applications. Thus, an analysis of design activities is taken literally from the typical continuous loop that results in continuous development and delivery.

Devops is a culture where "collaboration between development, quality assurance and operations" (94) are fluid (Ebert et al. 2016). Its focus is to encourage cross-functional teams that deliver continuous features, products, or services (Leite et al. 2020). It is a collection of philosophies, tools, and practices, rather than a single defined process. Devops is a compliment of lean principles and agile in its structure (Kim et al. 2016). Though devops closely resembles agile in its desire to deliver capabilities quickly, devops brings two traditionally separate practice together development and operations which would normally be isolated.

Devops until now has been mainly used for software and IT development focusing on continuous improvement. There are three main phases to devops: build, deployment, and operations (Ebert et al. 2016). Devops relies on a value stream because of applying lean principles and has the iterative structure of agile (including stories and creating a backlog) (Kim et al. 2016). Kim et al. (2016) argues to be successful, devops must embody flow, feedback, and continual learning and experimentation, and it relies on user engagement throughout the entirely of the process (Kim et al. 2016).

Ideally, there is constant feedback on work completed which enables the designers and developers to independently change, integrate and validate their product. Small changes are completed in a short amount of time, tested, and then deployed into a product (Dornenburg 2018). Kim et al. (2016) suggests a feature, or a capability or product would initially start in the planning phase where its requirements are defined and the creation of any initial up-front documentation. It would then progress through code and build for the design, through operate where it would be monitored and fed back through the loop for continuous improvement. In the first phase of work, design and deployment activities occur and in the second phase testing and operations are executed as value streams (Kim et al. 2016). Lastly, Kim et al. (2016) states that rather than work flowing through the first phase and then the second phase, the goal is for them to happen concurrently – enabling fast flow and high quality, and the process requires small batches to build quality into each part of the value stream.

If the process is considered from Ebert's three-phase model, then once a project is kicked off, it begins in the build phase where the team builds the feature/capability. Ebert et al. (2016) insists that the team focus on continuous integration, and test early and often. Once in the deployment phase, the main goal is to place the feature/capability under configuration management (Ebert et al. 2016). Finally, once in the operations phase, they describe teams use logging and monitoring tools to receive feedback that can be used to continuously improve the system via improvements.

From the developer's perspective there is criteria critical to the start of a project or an improvement iteration. Developers should ensure the systems has met the requisite gates and approvals prior to making updates and they are encouraged to "place code into

production without coordinating with members of other development teams" (Zhu, Bass, and Champlin-Scharff 2016, 33). They add that it can affect overall design choices and the style in which architecture is defined. Accordingly, they inform that configuration management is critical because a system may move through the devops cycle quickly and that the architectural style of system and its interfaces will affect how a system is monitored after deployment (Zhu et al. 2016).

Microsoft implements a four-step process for its devops practices for software and IT services that includes the following phases: plan, develop, deliver, and operate (Microsoft n.d.). The planning phase includes the upfront definition of features and capabilities, and as the team shifts into the develop phase all aspects of the design process including concept and development activities are executed (Microsoft n.d.). Similarly, to Ebert's three-phase model, the product is placed under configuration management control and finally monitored and maintained during the operate phase (Ebert et al. 2016; Microsoft n.d.). Activity-wise the deployment and operate phases fall outside of the concept and development phase per the *INCOSE SE Handbook* – though due to continuous learning and feedback, devops could be considered as always being in the development stage.

THIS PAGE INTENTIONALLY LEFT BLANK

III. CASE STUDY REVIEW

This chapter analyzes literature for available or defined processes for the given methods relating specifically to hardware development. Each are analyzed for their effectiveness based on the literature findings. If such processes are not defined, or not well defined, characteristics and activities of the process are elaborated. Further implementation analysis is provided in addition to example for when to apply such methods.

A. METHOD APPLICATION AND PROCESS IDENTIFICATION

1. Design Thinking

While none of the case studies included a detailed process or details for how design thinking was specifically implemented, they each provide evidence that the general phases of design thinking can be implemented in hardware development. Even the way in which organization implement design thinking for software development can be extrapolated and applied to hardware development. Design thinking is not prescriptive enough to be suited for only lane of product development. Rather, literature suggests it is widely applied across many industries and can be applied to hardware development.

Design thinking is known to solve wicked problems. In solving such problems, Chang, Kim, and Joo (2013) analyzed how Samsung and Apple approached design thinking and propose that firms select "different paths to achieve design thinking depending on environmental dynamics" and organizational capabilities, where some have less or more exposure to changes. In their analysis the authors propose a technology epiphany path that outline how an organization achieves a balanced design thinking team. This team plays the role of the final decision maker and dominates business decisions, which in turn, allows them to make informed decisions and come to a solution (Chang et al. 2013). Although criticized sometimes, this separation resulted in less design limitations and now "Apple products are welcomed by a massive number of consumers, even though the individual features do not necessarily outperform other products" (Chang et al. 2013, 31). Though the design thinking phases are not elaborated as part of this case study, the authors infer Apple and Samsung generally implement the design thinking phases as

defined in Chapter II. In another Apple study, Thomke and Feinberg (2012) noted for the operating system, the development team focused on the physical features people would want and perfected those prior to working the technical features and capabilities. Apples overall design thinking strategy focused on the most minute details (Thomke and Feinberg 2012).

In a study conducted by Marlena Pop (2020) for the development of the Leather Library project, she indicates that all stages of design thinking were involved. For the Ideation stage (third) the design team chose to utilize brainstorming and sketching to come up with candidate solutions (Pop 2020). Another aspect the design team implemented during ideation, was to randomly assign roles during conceptualization including a project manager, three-dimensional design manager, quality manager, environmental design manager, and market manager resulting in the team generating many potential solutions (Pop 2020).

On the contrary, Mazzuchetti, Lopes, and Barbosa (2019) present an approach where design thinking is implemented for new products. The goal was to stimulate new ideas through design thinking (Mazzuchetti et al. 2019). They propose an approach to design thinking for new product development consisting of the following steps: identify where to find innovation opportunity, discover the innovation opportunity, develop the innovation opportunity, test the ideas and prototypes, and implement the solution.

In another study, a software team developed and tested a phone application that provides self-management tools for type-2 diabetes (Peterson and Hempler 2017). The team used a three-phased design thinking method that includes inspiration, ideation, and implementation (Peterson and Hempler 2017). In the first phase, Peterson and Hempler (2017) describe the team activities as making observations and gathering information from the subjects regarding challenges and need related to living with diabetes. In the second phase, they describe that the team executed focus groups to determine the app needs where ideas were explored and developed and finally tested over several weeks where users were interviewed about the app's usability. Ultimately the team refined the application over the course of design activities, prototyping, and receiving feedback iteratively, to support five

major functions in supporting the self-management of diabetes (Peterson and Hempler 2017).

2. Lean Product Development

In the context of LPD, much of the existing literature focuses on the activities in which provide value to an organization. For example, a case study on product development at Ford discussed the necessary process transformations implemented to create a leaner development process, as they found their current process was wasteful and would eventually affect their ability to compete (Liker and Morgan 2011). Ford had to find ways to make immediate and continuous improvements, so as Liker and Morgan (2011) suggest, they created process improvement maps and held value stream mapping events that enabled for more dialogue within the team. Matrices were created to prioritize opportunities and to identify interdependencies and set-based concurrent engineering principles were applied to "work simultaneously for longer periods and delay key decisions until points in the process that were closer to customer" interactions and milestones (Liker and Morgan 2011, 22). The authors also point out that quality of event criteria was used to ensure quality was not only required but measured at each milestone throughout development. More importantly, the team at Ford held cross-functional reflection events at critical milestones to talk opportunities, successes, and waste and created value streams that worked towards cross-functional objectives (Liker and Morgan 2011). Ultimately, Ford found that they needed to implement front end loading and innovation into their process and be proactive early on (Liker and Morgan 2011). The ability to pull value throughout the process was critical to Ford's success.

In another automotive case study, Tuli and Shankar (2015) argue that there are many collaborative activities or processes that the generic development approach cannot support. They detail two case studies in the automotive industry that employ a generic approach and then a collaborative and lean approach referred to as OEM1 and OEM2, respectively, where the collaborative approach consisted of performing a value analysis that was implemented as a value stream. Both development approaches were executed and then compared qualitatively and quantitatively (Tuli and Shankar 2015). Qualitatively,

three major development phases were measured, and the team determine the overall development cycle was greater for OEM and quantitatively, evaluation data shows that of the five parameters considered, OEM1 measured high in all categories except for design cost (Tuli and Shankar 2015). The authors show the organization was much more successful in their development when it implemented a collaborative and lean environment. The results also showed improvement in key performance indicators such as cost and schedule, risk factors, and quality to name a few (Tuli and Shankar 2015).

3. Agile

There is little supporting literature documenting an agile process approach to hardware development. Rather, only descriptive principles are outlined that could be applied more generally to hardware development. One group in particular, Rockwell Collins, agrees that hardware development platforms are lacking commercially, so in turn, they created their own, but the process is not available to the public (Dove 2018).

Some organizations, like the LEGO Group, have been able to implement hardware development as part of a higher-level agile transformation where software and hardware are integrated into system level design. The LEGO Group implemented agile methods in multiple departments and found that it not only drove process change, but also positively affected the behaviors of their employees (Sommer 2019). The LEGO Group "demonstrates that an agile transformation [via scrum] can be successfully executed by applying agile values and principles to the transformation efforts themselves, enabling agile behavior rather than prescribing a particular method or model" (Sommer 2019, 20). The data from the LEGO Group, Sommer (2019) suggests, that the agile transformation was successful in its ability to create product-oriented teams who shared ownership in their responsibilities, for delivering quality products, and value via design iterations. More importantly, the LEGO Group implemented and executed their agile transformation via a 100-day plan that consisted of four value streams, and defined activities that were rescoped overtime and throughout the development process (Sommer 2019).

Lockheed Martin is another example in which agile was applied more generally. The objective of the Agile Systems Engineering Life-Cycle Model (ASELCM) project was

to identify foundational principles of agile that could be applied in multi-discipline systems engineering (Dove, Garlington, and Schindel 2018). Specifically, the article discusses the importance of "systemic, activity-based, continuous innovation" in an agile environment (Dove, Garlington, and Schindel 2018). These are analyzed in the case of Lockheed Martin Aeronautics Integrated Fighter Group (IFG), where the team was faced with the need to create an agile system engineering environment but address the urgent capability need (Dove, Garlington, and Schindel 2018). The authors further suggest that to be successful in both, the organization needs to create interconnection standards for physical connections, data connections and interfaces, security, and services which would enable process activity assembly. To sustain the agility within an SE development process, its infrastructure must remain agile and as outlined by Dove, Garlington, and Schindel (2018).

4. Set-Based Concurrent Engineering

On its own, there is a clear lack and structure of an SBCE process, rather literature is limited to a set of generic descriptive principles (Ashaab et al. 2013). The implementation of set-based concurrent engineering principles has been described as paired with other methods including agile or LPD. Though, Raudberget (2011) proposes a SBCE model derived using the three principles of SBCE shown in Figure 3, that could be used foundationally to define a process. The process begins with creating ideas/concepts and creating a morphological chart (or comparison chart) of those ideas/concepts. Overtime, the morphological chart is updated based on intersecting sets that define detailed requirements and features. In an iterative fashion, the development teams work to narrow the sets to identify and define the final design.

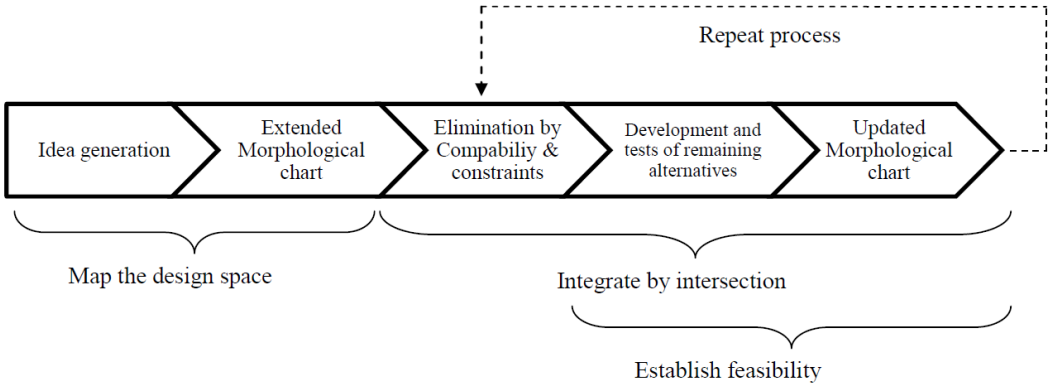

Figure 3. Proposed Set-Based Concurrent Engineering Development Model. Source: Raudberget (2011).

Al-Ashaab et al. (2013) proposes an LPD process model that integrates the principles of SBCE for use in the aerospace industry, specifically the development of a helicopter engine. The developed process focuses on core enablers of "lean product development such as value focus, set-based solutions, integrated documentation, knowledge creation, and innovation" (Al-Ashaab et al. 2013, 282). Al-Ashaab et al. (2013) further argues that constant feedback from engineering highlighted areas of improvement and identified waste. Their proposed model is not available via literature, however much of their research stems from the use of SBCE process and activities defined in Khan et al. (2011) paper "Set-Based Concurrent Engineering Process Within the Lean PPD Environment" as a baseline.

Similarly, Canciglieri et al. (2010) presents an adapted product life cycle between SBCE and traditional development process. They propose that applying SBCE speeds up the process substantially, especially when teams work together and simultaneously, the result of the development can be faster and less expensive. This study does not necessarily outline a process, rather Canciglieri et al. (2010) show that the differences in the number of parts, number of operations, attachments, cost, time, and manpower were significantly lower using an SBCE approach rather than traditional design methods.

5. Systems Thinking

According to Goede (2005), "it is the aim of the systems thinkers to describe social systems where people and their value form part of the system" (83). One of the key principles to the systems thinking approach is the hierarchical nature of any system – it implies that every system is part of another larger systems (Goede 2005). In Goede's (2005) work, Whitten et al. (2004) indicates that for information systems there is no systems thinking process defined, rather it uses descriptive systems thinking principles. According to his research, when applying soft systems methodology, the more important of the principles is getting the user involved to better understand the system.

Rutherford provides a basic example of how systems thinking is applied within an organization (2019). He presents a hypothetical story of the company Acme, that had a persistent problem. As a result of brainstorming, Acme was able to fix the immediate issues independently. This, however, did not solve the problem. Rather, they needed to adopt a system view due to underlying problems, as Acme's issues were just symptoms of a larger issue.

In a study of health care management, Lebcir (2006) demonstrated under performance of health care systems is due to inadequacy of the tools and methods used to analyze and study them. The decision-making processes within the health care system do not appropriately capture the most important components of the system and their interconnectedness (Lebcir 2006). Lebcir (2006) describes how systems thinking principles were applied to "formulate, model, and analyze the system" (6). Their aim was to show, using systems thinking, the complexity in the structure of the health care system, by starting with simpler structures and adding in components to show how it can grow both in size and complexity. Systems thinking was specifically used in Lebcir (2006) to create a model or a mapping to address the bigger problem.

In a different application, systems thinking was implemented with design thinking to identify when doctors and their students provide liver fluke infection information to places within their communities and where the discrepancies are, based on areas that have high infection rates (Samiphak et al. 2016). Samiphak et al. (2016) describes how a team

of medical professionals implemented systems to analyze how to best educate and provide information most effectively to communities. They also discuss how the approach led the team to reanalyze and reframe the data with a human centered (or design thinking) approach, in turn resulting in their ability to identify a disconnect between scientific and cultural knowledge specifically relating to awareness of health effects from eating certain fish.

6. Devops

Devops is mainly implemented in software and IT. Gill et al. (2018), discusses the process view of devops based on information management systems including full product life cycle, continuous delivery pipeline, continuous improvement, multistage testing, multistage deployment, and analytics. For each process, high level descriptive principles that align with those described in Chapter II are referenced. No specific application however is described for hardware development.

According to Farroha and Farroha (2014), devops is the best approach for software development within the DOD mission environment. Though the authors do indicate that a devops culture is made of an integrated, cross-functional team that is tasked with solving problem they do not reference2 or specifically analyze hardware development (Farroha and Farroha 2014). In another article, Banica et al. (2017) propose that devops can be used as a project management tool. They argue that devops is an extension of agile, and where devops aims to test and release components when they are complete, agile delays delivery of the components to the customer and focuses on smaller component completions. They also argue that devops targets to increase efficiency of a design activity, more collaboration between design and implementation, and a faster transition of components from design to operation.

B. IMPLEMENTATION ANALYSIS

Literature review and analysis indicates there is little evidence of specific hardware development processes for the methods described herein. Raudberget (2011) proposed an SBCE development model that could be applied to hardware development, though there is a lack of supporting literature on its application. There is extensive research showing

application within a software development, however hardware development has limitations including hardware availability and failure analysis that make it difficult to define a singular process. Rather, literature describes that the guiding principles for each method can be applied more generically to hardware development in creating new process or enhancing a defined process.

Literature shows methods can be used concurrently. For example, some of the reviewed case studies and literature reveal the use of at least two methods concurrently. In Samiphak et al. (2016) applied systems thinking with a design thinking approach to better understand and solve their given issue. Devops according to Banica et al. (2017) is also implemented concurrently with agile process structure (i.e., following the scrum method). Ahmed et al. (2013) describes Toyota's product development process as lean in nature (i.e., value streams), implements SBCE principles, and further suggests SBCE be applied to a lean environment. The one size fits all narrative becomes too generic when dealing with complex hardware development. Such an endeavor requires cross-functional, cooperative teams willing to adapt to a different development environment and change their culture to align with a new strategic development approach.

A specific example of using methods together is Scale Agile Framework, or SAFe, which is method that provides an agile framework, in conjunction with applying systems thinking, LPD, and devops. In a white paper published by Scaled Agile (2021), implementing agile development on its own is not enough. The big picture of the SAFe is creating business agility with core competencies that results in an agile structured delivery of software products, within large enterprises (Scaled Agile 2021). According to a multivocal literature review done by Putta, Paasivaara, and Lassenius (2018), there are many benefits of adopting SAFe including those of business and organizational benefits, and measurable quality throughout development. More specifically, in addition to improved quality, there is overall reduction in defects, continuous improvement, and waste elimination (Putta et al. 2018).

Nevertheless, these methods should be considered based on what scenarios they may apply best and some examples for consideration are described:

- Design Thinking: prioritizes target audience needs
- Lean Product Development: execute development activities that provide value and remove those that do not, streamline the process – add and pull value based on stakeholder needs
- Agile: speed-to-market delivery, incremental deliveries, self-organizing teams, and early testing that builds in quality during throughout
- Set-Based Concurrent Engineering: cross-functional focus in design and development activities to improve quality and design robustness
- Systems Thinking: applies best in a problem space where there is uncertainty in the problem space
- Devops: used to develop, test, and deliver features and products quickly and deploy modification or updates continuously to improve the product

IV. METHOD ALIGNMENT AND MAPPING

This chapter accumulates the phases and activities for the identified methods as described in Chapters II and III. The first section describes the baseline process for comparison and list assumptions necessary for the proposed activity mapping. The second section presents figures with high-level summaries for each method including its activities, technical principles, and requirements, and present detailed tables for method that list the activities as they occur in each phase. The third section proposes an alignment of the phases of each method to the INCOSE generic life cycle in terms of concept and development phases and activities. The fourth section proposes a mapping of the activities that occur throughout the INCOSE generic life-cycle concept and development stages to those that occur within the six methods chosen for analysis in this work that can be used as a decision aid, a guiding resource, or for educational purposes. Finally, implications and limitations are presented.

A. INCOSE PROCESS BASELINE AND ASSUMPTIONS

Prior to analyzing how each method aligns to the generic life cycle as presented in the *INCOSE SE Handbook,* the following assumptions are made:

- Concept and development phases of INCOSE generic life cycle are considered for baseline comparison
- The alignment is based on literature research presented in Chapter II and Chapter III, and is not representative of cumulative existing literature
- For a given method that does not have a fully defined process model, it will depict concept and development phases
- Each process depicted can be applied to hardware development
- Descriptions for a given method will indicate major activities that occur during a particular phase or stage of the process, and when applicable governing principles and characteristics

Given the above assumptions, the generic life cycle is revisited in Figure 4 as a visual representation of the phases that are considered in scope and those that are out of scope.

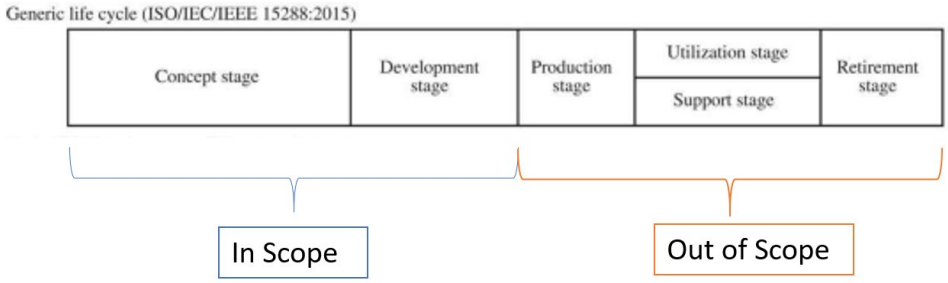

Figure 4. In Scope and Out of Scope Stages of the INCOSE Generic Life Cycle. Adapted from INCOSE (2015).

The generic life-cycle concept and development stages are considered in scope and are further summarized in Figure 5 (previously elaborated on in Chapter 1). The concept stage is broken into two phases, exploratory research and concept selection, and the development phase into two phases, system level design and detailed design. In addition to phase descriptions, governing technical processes are also included.

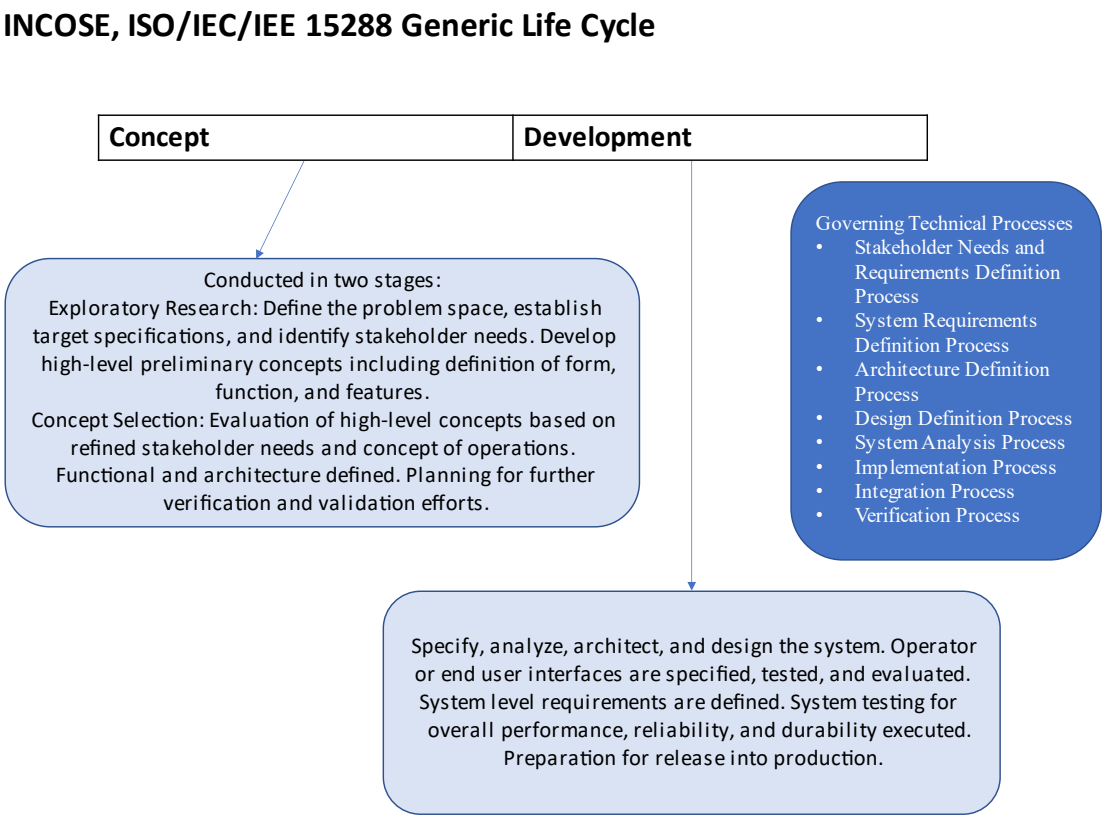

Figure 5. INCOSE Generic Life-Cycle Process Summary, Concept and Development Stages. Adapted from INCOSE (2015) and Ulrich and Eppinger (2016).

Next, the activities as described in the *INCOSE SE Handbook* are listed and associated with the phase and overall generic life-cycle concept and development phases in Table 2. The activities listed are not necessary in chronological order, rather they are listed as activities that occur as part of the phase. The activities included from the *INCOSE SE Handbook* are directly identified in Table 2.

Table 2. Generic Life-Cycle Stages, Phases, and Activities. Adapted from INCOSE (2015) and Ulrich and Eppinger (2016).

INCOSE, ISO/IEC/IEEE 15288:2015 - Generic Life cycle		
Life-Cycle Stage	Phase	Activities
Concept	Exploratory Research	Investigate feasibility of product concepts
		Define form, function, and features
		Create specifications and technical documentation
		Analyze of competitive products
		Economics of justification
		Identify mission requirements
		Identify stakeholder needs
		Establish target specifications
		Estimate Cost and Schedule
	Concept Selection	Evaluate Candidates
		Refine stakeholder needs
		Refine concept of operations
		Prototype or building mockups
		Develop models
		Run simulations
		Perform architectural tradeoffs
		Document risk opportunities
Development	System Level Design	Define and refine system requirements
		Define system architecture
		Define interfaces
		Test
	Detailed Design	Document system configuration
		Define quality assurances processes
		Procure/provision for production

B. DESCRIPTIVE METHOD FIGURES

This section elaborates on the phases for each of the identified methods: design thinking, LPD, agile, SBCE, systems thinking, and devops. More specifically, for each, the phases or stages will be expanded upon regarding the activities that occur. Additionally, any governing technical or general principles that apply are identified. These descriptive figures are not inclusive of all activities and do not include differentiating characteristics.

Following a similar paradigm, each method is visually represented like Figure 5 and activities listed in Table 2, based on the literature presented in Chapter II and Chapter III. The processes depicted are representative of proposed processes or derived from activity analysis. Phases or activities that fall outside of the concept and development phase are not considered for further analysis. There are cases in which a phase can be considered as part in scope and part out of scope. These are depicted for visual representation, and only analyzed for activities that fall within scope. This information will be used to align the methods and their activities to the generic life-cycle process as defined in the *INCOSE SE Handbook.*

1. Design Thinking

Figure 6 describes design thinking using the five phases as defined by Mueller-Roterberg (2018), Wrigley, Nusem and Straker (2020), and Pop (2020) to name a few. It further includes the delineation of the problem space and solution space identified by Mueller-Roterberg (2018). Design thinking can be a serial process, though it is meant to be iterative at any point to which the team needs to cycle backwards. For example, this could depend on whether the prototype that was created wholly meets the needs of the target audience. Getting the correct design and product though is unlikely to occur in one iteration through the process, it is more likely that once a protype is created and tested, it will require, at a minimum, modifications based on testing and feedback.

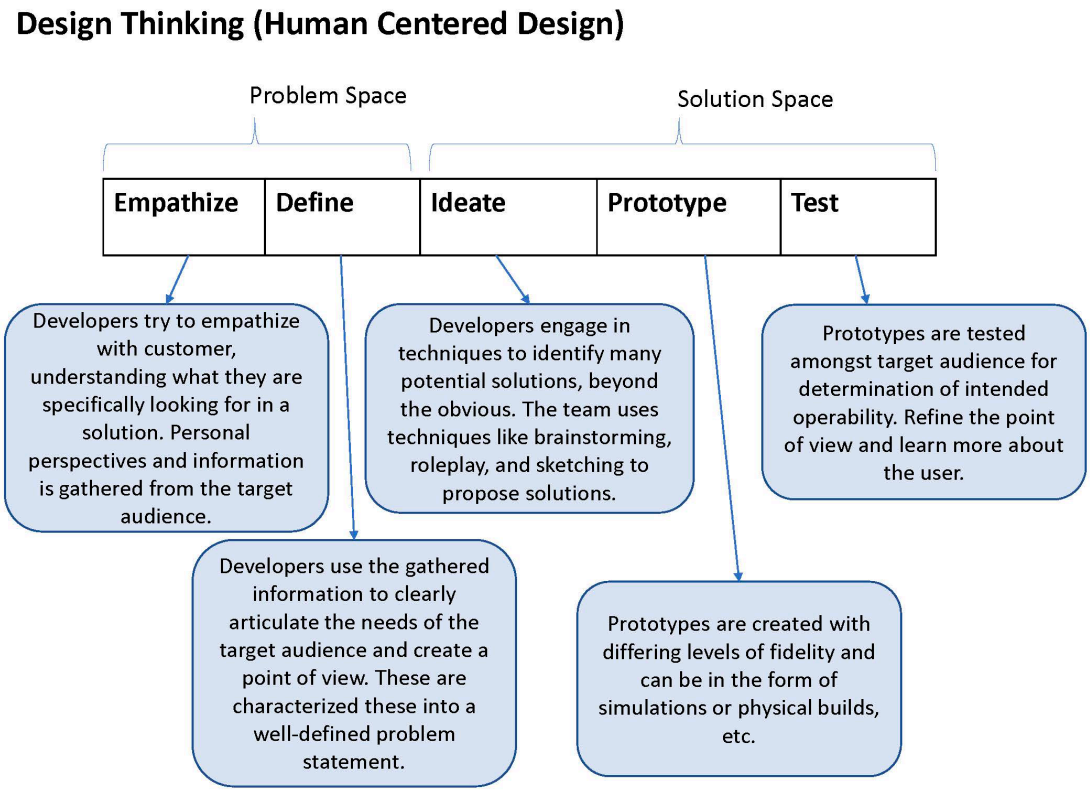

Figure 6. Design Thinking Development Process Summary. Adapted from Lindberg et al. (2010), Mueller-Roterberg (2018), Linke (2017), Pop (2020), Balcaitis (2019), and Hasso Plattner Institute of Design (2019).

Based on the phase description in Figure 6, the activities that occur are extrapolated and organized in Table 3. The activities are identified or reiterated via literature and case study analyses and matched to the phase in which they occur.

Table 3. Design Thinking Phases and Activities. Adapted from Lindberg et al. (2010), Mueller Roterberg (2018), Linke (2017), Pop (2020), Balcaitis (2019), and Hasso Plattner Institute of Design (2019).

Design Thinking	
Phase	Activity
Empathize	Identify stakeholder needs
	Gather inspiration
	Seek stories
	Prepare Research
Define	Define requirements definition
	Define problem statement
	Frame opportunities
	Identify meaning surprises and tensions
	Infer insights
Ideate	Brainstorm radical ideas
	Generate concepts
	Evaluate concepts
	Suspend judgement
	Refine ideas
Prototype	Develop models
	Create low-resolution prototypes
	Roleplay to understand context
	Build prototypes to think and learn
Test	Test amongst target audience
	Gather feedback
	Reflect and generate a new solution

2. Lean Product Development

The descriptive figure for LPD includes two phases: concept and development. Lean product development is considered in the context of applying lean principles to an existing hardware development process. Therefore, the major phases for consideration will be those of the generic life cycle and include concept and development. Figure 7 shows the concept and development phases and includes major activities that would occur when applying lean principles or exercises during the development process. It is important to keep in mind that when lean principles are applied to an existing process, the activities of which occur during both concept and development phases may not change.

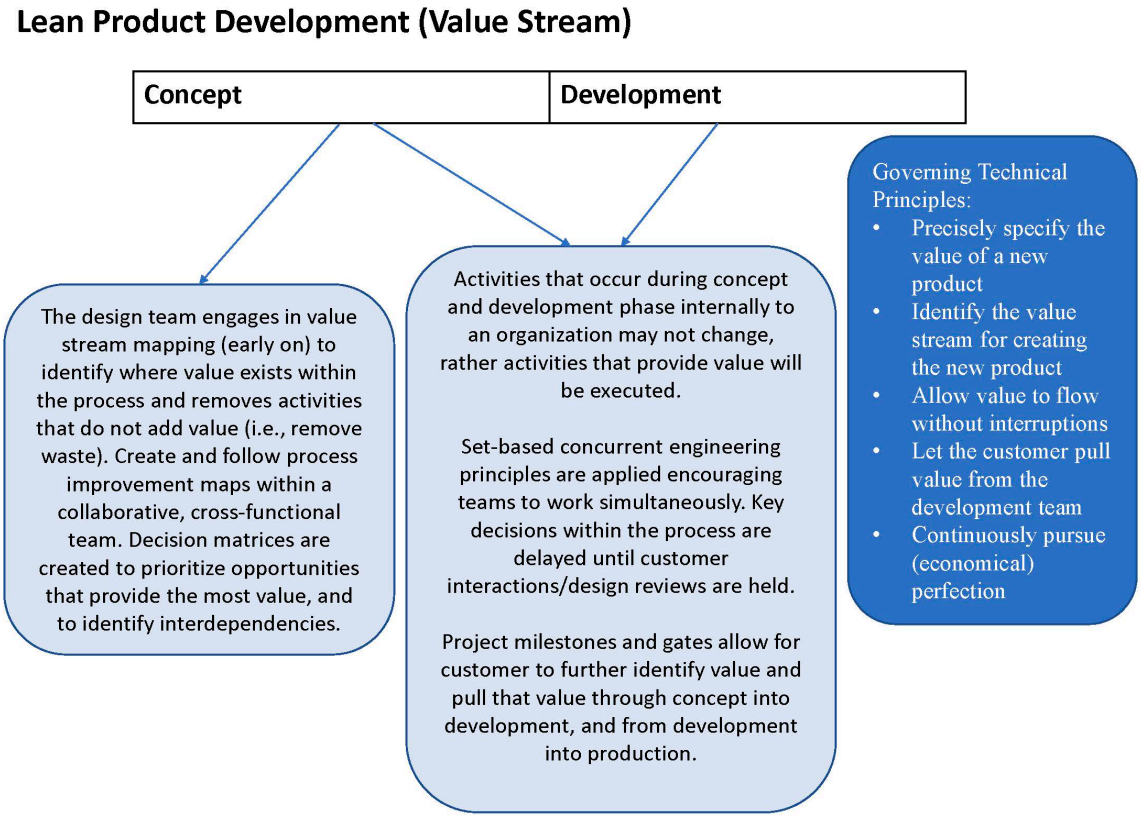

Figure 7. Lean Product Development Process Summary. Adapted from Mynott (2012), Mascitelli (2006), Radeka (2013), and Liker and Morgan (2011).

Based on the process description and technical principles, Table 4 shows the activities which then occur during concept and development. It is assumed that the activities during concept and development of which occur in the proposed lean product development process are at a minimum, the same or similar, to those that occur as part of the generic life-cycle concept and development stages. Therefore, Table 4 reflects specific lean activities in addition to concept and development activities defined by the generic life cycle (Table 2).

Table 4. Lean Product Development Phases and Activities. Adapted from Mynott (2012), Mascitelli (2006), Radeka (2013), and Liker and Morgan (2011).

Lean Product Development	
Phase	Activities
Concept	Identify value stream
	Identify waste
	Eliminate waste
	Define quality assurance processes
	Define requirements
	Develop features in increments
	Generate concepts
	Evaluate concepts
	Prototype or build mockups
Development	Develop models
	Run simulations
	Perform architectural tradeoffs
	Document risk opportunities
	Refine value stream
	Define and refine system requirements
	Define system architecture
	Perform system level test
	Continue feature development
	Iterate for constant feedback
	Identify waste
	Eliminate waste
	Seek continuous improvement

3. Agile

Agile on the other hand, from a literature perspective, is mostly supported in software and IT service applications. With little evidence that it can or has been applied to hardware development, the three stages shown in Figure 8 reflect those identified in its general flow and governed by scrum. The LEGO Group argues that by encouraging agile behavior, an agile process transformation is an easy transition (2019). Activity analysis of agile scrum indicated the release and transition stage falls partially outside of the scope of this work (i.e., outside of development) because the design team may iterate back through sprint execution to complete the tasks within the backlog or as part of a user story. The

main activities through the release and transition phase include releasing a feature or a complete product so it can be transitioned to a production environment or deployed to its end users.

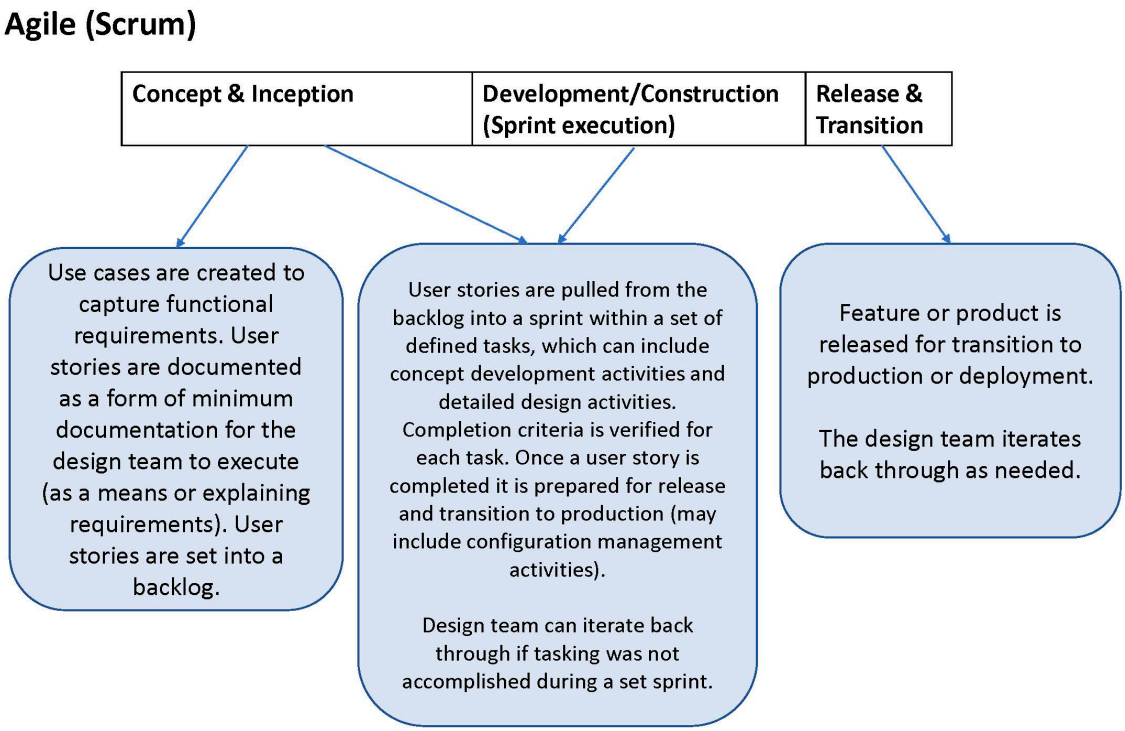

Figure 8. Agile Scrum Development Process Summary. Adapted from Walsh and Mahesh (2015), Pries and Quigley (2010), and Vanderjack (2015).

Like LPD, activities of agile scrum can include any of those that fall within concept or development, or those that fit are defined within user stories. Therefore, Table 5 reflects specific agile scrum activities, lean activities, and concept and development activities defined by the generic life cycle (Table 2).

Table 5. Agile Scrum Phases and Activities. Adapted from Walsh and Mahesh (2015), Pries and Quigley (2010), and Vanderjack (2015).

Agile	
Phase	Activities
Concept and Inception	Create use cases
	Define functional requirements
	Create user stories
	Define minimum documentation
	Create backlog
	Plan sprints for task completion
	Execute sprints
Development/Construction	Generate concepts
	Evaluate concepts
	Prototype or build mockups
	Develop models
	Run simulations
	Perform architectural tradeoffs
	Document risk opportunities
	Define and refine system requirements
	Define system architecture
	Perform system level test
	Define quality assurance processes
	Place product or feature under configuration management control
	Procure necessary resources
Release and Transition	Release feature
	Release product
	Transition to production
	Transition for deployment

4. Set-Based Concurrent Engineering

SBCE is another method in which there is a lack of developed process for general use and specifically as it relates to hardware development. On its own SBCE lacks structure, though literature supports its existence based on its principles being applied in some other method (like LPD). Strictly considering SBCE, it is governed by three principles defined by Sobek et al. (1999) – "map the design space, integrate by intersection, and establish feasibility before commitment" (73). Though, it is Raudberget (2011) that proposes an SBCE model (Figure 3) that includes development phases that align to those

principles identified by Sobek et al. (1999). Figure 9 is based on the process proposed by Sobek et al. (1999), in addition to supporting literature detailing SBCE principles, categories, phases, and activities.

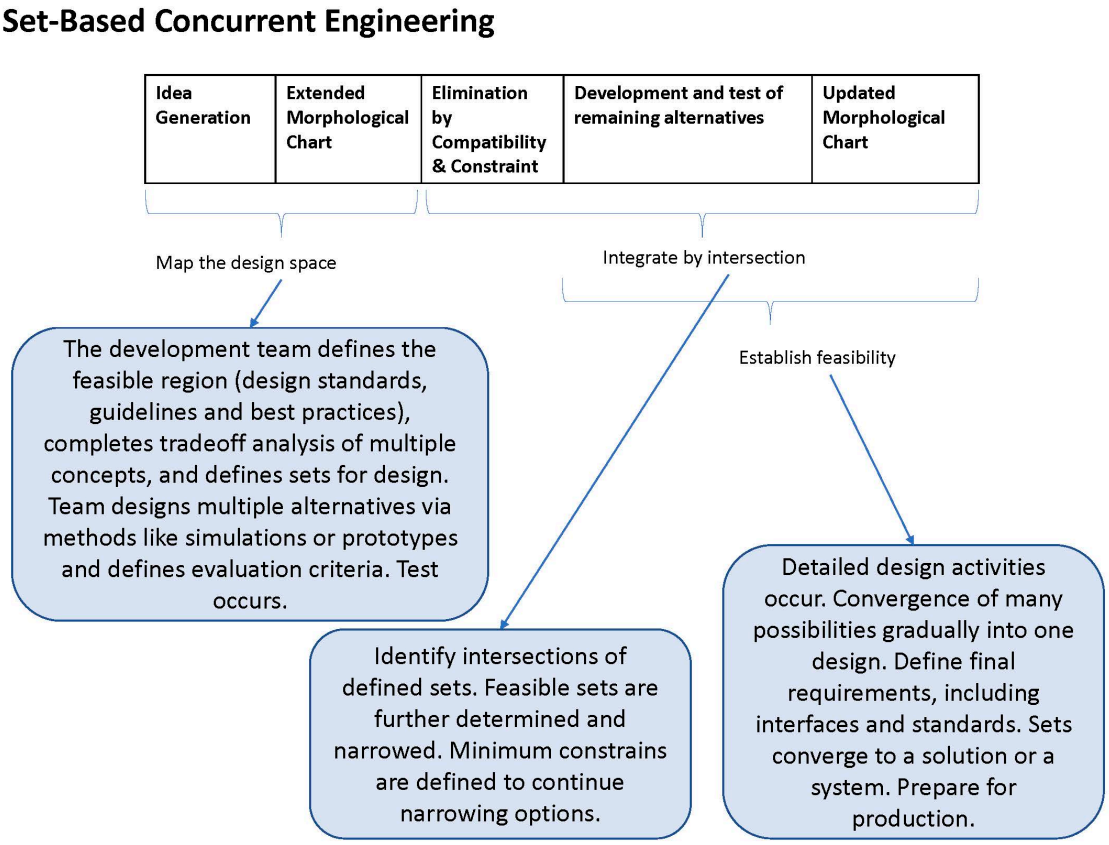

Figure 9. Set-Based Concurrent Engineering Development Process Summary. Adapted from Sobek et al. (1999), and Khan et al. (2011).

Table 6 specifically lists the phases as described by Sobek et al. (1999) and the activities supported by the literature presented in this work. The governing principles defined by Sobek et al. (1999) are not include in Table 6.

Table 6. Set-Based Concurrent Engineering Phases and Activities. Adapted from Sobek et al. (1999), and Khan et al. (2011).

Set-Based Concurrent Engineering	
Phase	Activities
Idea Generation / Extended Morphological Chart	Define design standards and guidelines
	Define requirements
	Perform tradeoff analysis of multiple concepts
	Define sets
	Develop models
	Develop prototypes
	Run simulations
	Identify functions and mechanisms (documented on morphological chart)
	Define evaluation criteria
	Perform evaluation tests
Elimination by Compatibility and Constraint	Identify intersection of sets to determine feasibility
	Set minimum constraints
	Down-select feasible sets
Development and Test of Remaining Alternatives	Increase design details
	Perform tests
	Update morphological chart
	Narrow sets gradually

5. Systems Thinking

Literature supporting systems thinking in terms of product development and hardware development especially is practically non-existent. Much of the study of systems thinking focuses on systems theory and applying those principles within a business and organization (Galli 2018). Though, Rutherford (2019) indicates that systems thinking can be applied more generally to solve existing problems, adapt, find fast solutions, and prevent future problems – where any problem at any time can be categorized by a system archetype giving the system structure. It was Maani and Cavana (2007) who proposed systems dynamics, a systems thinking process that aligns with hard systems thinking where the focus, as described by Lebcir (2006), is modeling the problem throughout, allowing the growth of an initial simple model and structure to something more complex. Though there

are other implementations of systems thinking, Figure 10, reflects the systems thinking process specifically relating to systems dynamics. When implementing systems thinking, there are minimum process requirements as defined by Sweeney and Sterman (2000) (also reflected in Figure 10).

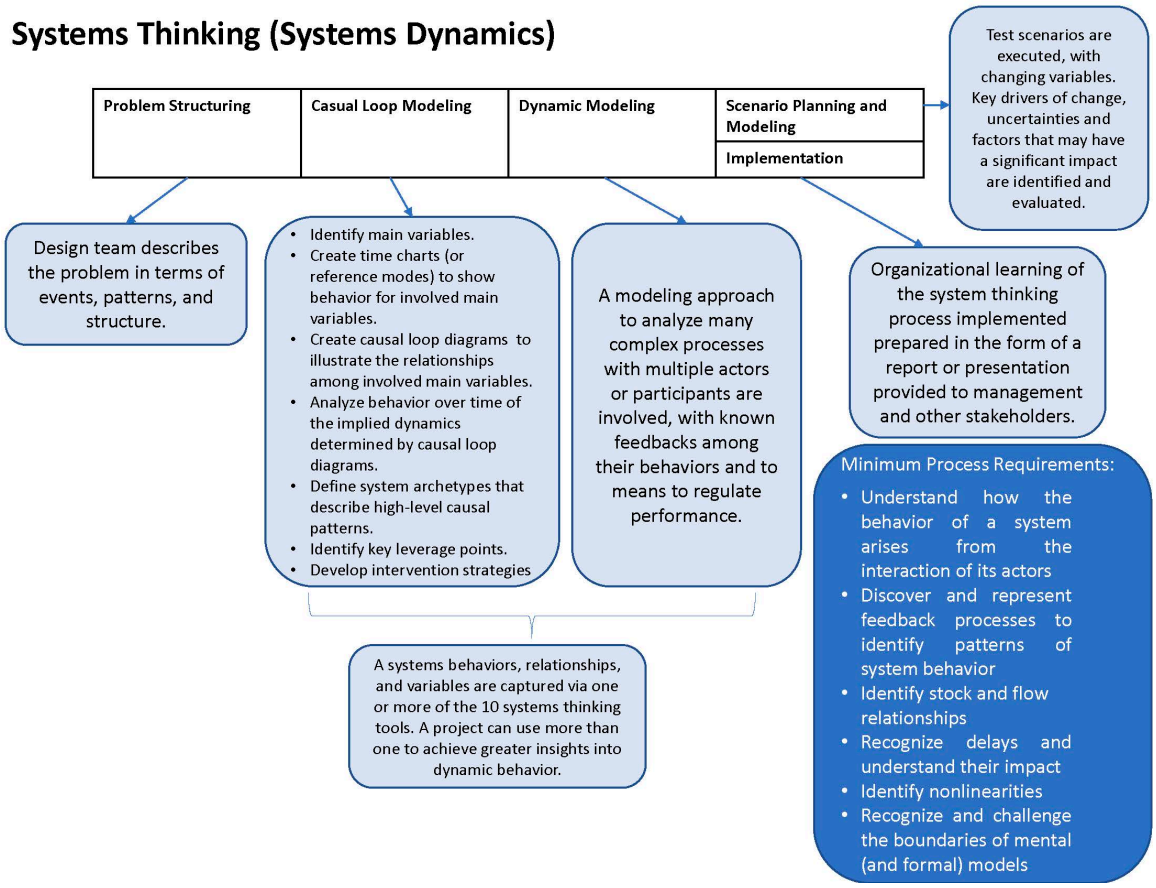

Figure 10. Systems Thinking Systems Dynamics Development Process Summary. Adapted from Maani and Cavana (2007), and Sweeney and Sterman (2000).

Table 7 then reflects the activities specifically relating to the phases as identified. The phases and activities shown in Table 7 do not reflect the minimum process requirements as defined by Sweeney and Sterman (2000). Additionally, Table 7 does not reflect the "Scenario Planning and Modelling" and "Implementation" phases occurring concurrently.

Table 7. Systems Thinking Systems Dynamics Phases and Activities. Adapted from Maani and Cavana (2007), and Sweeney and Sterman (2000).

Systems Thinking	
Phase	Activities
Problem Structuring	Define the problem
	Describe problem in terms of events, patterns, and structures
Casual Loop Modeling	Identify main variables
	Visually represent behavior via time charts (or reference models)
	Develop casual loop diagrams
	Analyze relationships amongst variables
	Analyze behavior overtime
	Identify system archetypes
	Document high-level casual patterns
	Identify key leverage points
	Develop intervention strategies
Dynamics Modeling	Develop models to analyze the system
	Represent actors or participants in model
	Analyze complex processes
	Analyze feedback among behaviors
	Regulate performance
Scenario Planning and Modeling	Execute test scenarios (with changing variables)
	Identify key drivers of change
	Identify key drivers of uncertainty
	Identify factors of significant impact
	Evaluate factors of significant impact
Implementation	Implement organizational learning plan
	Provide learning plan to management
	Provide learning plan to stakeholders

6. Development and Operations

Devops too lacks a specific focus for implementation specifically into hardware development programs. There is literature though that supports devops execution along with other methods (Kim et al. 2016). On its own though, devops drives a collaborative environment in which the designers and the operators work together through the process, ridding of groups previously siloed from one another (Leite et al. 2020). It is a collection

of philosophies tools and practices than a defined process. Therefore, in proposing a process, Figure 11 represents phases described and supported amongst literature and case studies. For example, Ebert et al. 2016 identifies three phases, while Microsoft identifies four phases, in which the only difference is the addition of Microsoft's planning phase.

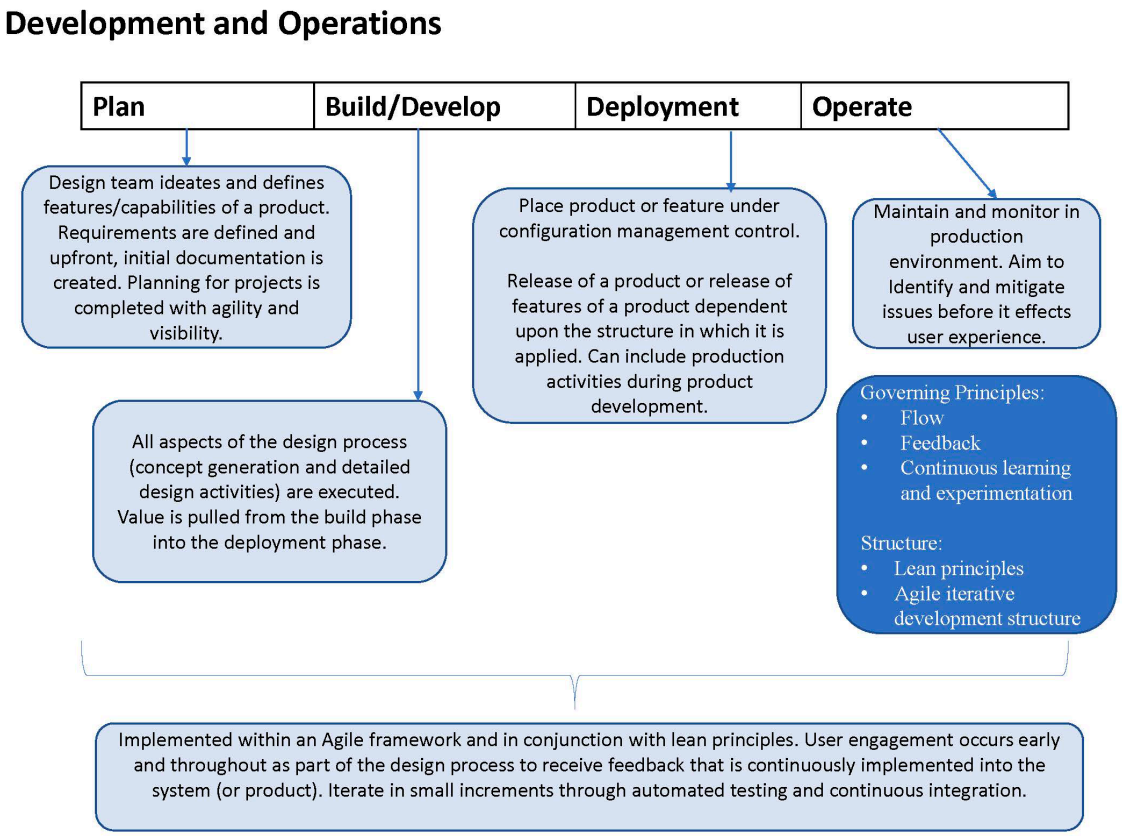

Figure 11. Devops Development Process Summary. Adapted from Kim et al. (2016), Ebert et al. (2016), Dornenburg (2018), and Microsoft (n.d.).

Table 8 documents the activities related to identified phases, though these are not specific to only Microsoft and Ebert et al. (2016) phase descriptions. The governing principles and structure as defined by Kim et al. (2016) are not reflected in Table 8.

Table 8. Devops Phases and Activities. Adapted from Kim et al. (2016), Ebert et al. (2016), Dornenburg (2018), and Microsoft (n.d.).

Development and Operations	
Phase	Activities
Plan	Ideate feature/capabilities or product
	Define features/capabilities of product
	Define requirements
	Create initial documentation
	Plan for remainder of project
Build	Identify value stream
	Generate concepts
	Evaluate concepts
	Prototype or build mockups
	Develop models
	Run simulations
	Perform architectural tradeoffs
	Document risk opportunities
	Define and refine system requirements
	Define system architecture
	Perform system level test
	Define quality assurance processes
Deployment	Place product or feature under configuration management control
	Procure/provision for production
	Release feature
	Release product
	Produce product
Operate	Maintain feature/product
	Monitor feature/product
	Identify risks and issues
	Mitigate risks and issues

C. METHOD ALIGNMENT

Based on the descriptive figures and activity tables presented, each of the methods are aligned the generic life-cycle process. Specifically, their phases, activities, and general execution are analyzed and compared to the generic life-cycle concept and development stages. Figure 12 is a visual representation of how each method aligns to the generic life-cycle concept and development phases, and subsequently each other.

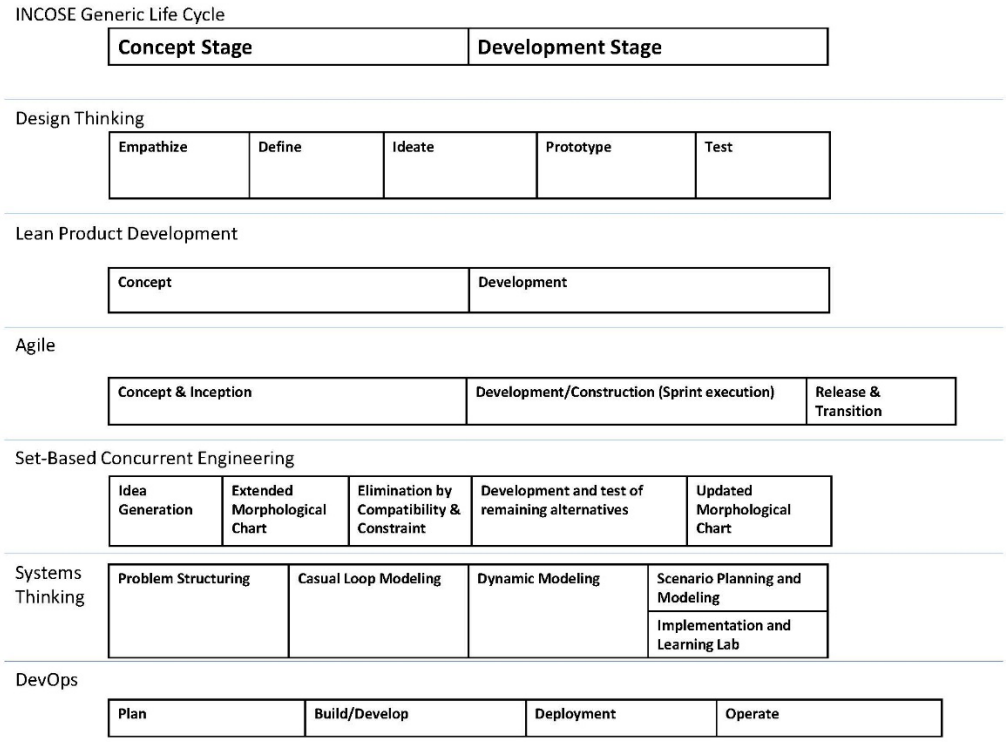

Figure 12. Method Alignment

The Figure 12 proposes that methods execute in a similar fashion–they consist of activities that are comparable in nature though the way in which they are executed is different (i.e., the process they follow). For example, consider the activities that occur in the design thinking empathize and define phases. Figure 12 shows that activities in which occur during these phases align to the activities that occur in the concept stage of the generic life-cycle concept stage. The design thinking ideate phase has activities that align to activities of which occur during the generic life-cycle concept and development stages, so it aligns to both. Specifically, as part of the ideate phase of design thinking the team comes up with many potential solutions and concepts (concept stage) and begins to refine ideas based on stakeholder needs and requirements (development stage). Similarly, all activities that fall within the prototype and test phase align to those or are like those in the development stage of the generic life cycle. For each of the identified methods in Figure 12, the phase alignment based on activities is identified and can be expanded upon.

D. ACTIVITY MAPPING

The alignment in Figure 12 is strongly based on the activities of which occur during a given phase, or phases. So, if the activities are solely considered, then such activities that occur as part of design thinking, LPD, agile, SBCE, systems thinking, and devops can too be aligned, or mapped to those that occur in the concept and development stages of the generic life cycle. The challenge though, is that not all activities are created equal. That is, the activity description may not be a one-for-one match based on literature and case study analysis. Further, a given method may have a different quantity of activities than another method, or even some that are completely different. So rather than mapping the activities of the given methods to the generic life-cycle stages, the generic life-cycle stages will be mapped to the methods presented throughout this work. The mapping will indicate whether the generic life-cycle activity definitively occurs in each method based off the literature and case study research in Chapter II and Chapter III, and the descriptive figure and activity table as documented Figure 5 and Table 2 respectively. This mapping is presented in Figure 13. Where there is an "X" denotes an activity that definitely occurs, and where there is an "●" denotes an activity that may occur.

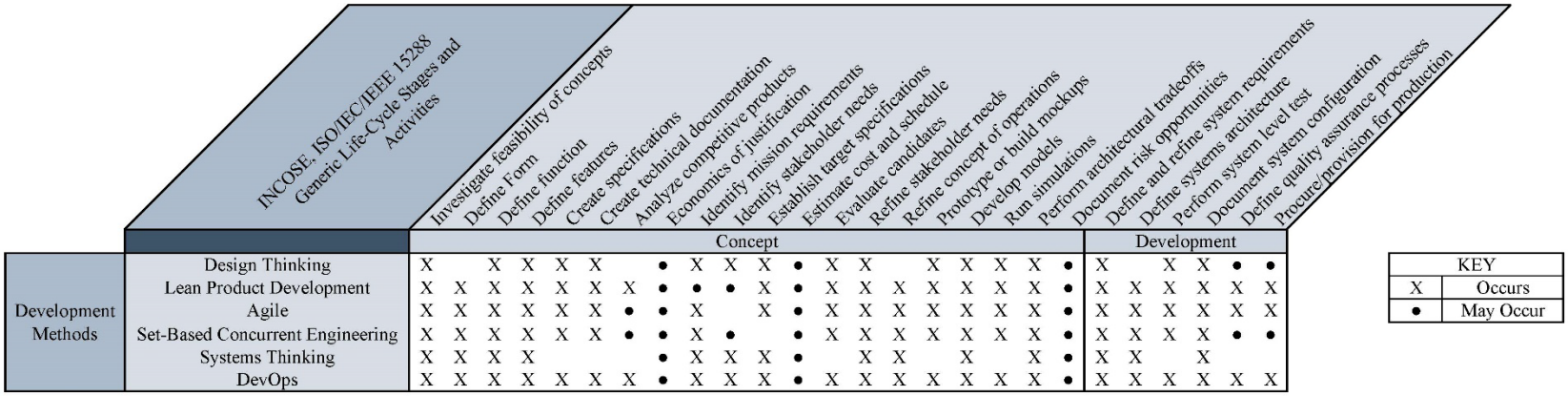

Figure 13. High-Level Development Method Activity Mapping

E. ACTIVITY MAPPING SCENARIO

Take for instance, a development team or program, that chooses to execute a hardware development project using a method other than their standard process-how would they know which method to choose? Or maybe a hardware design team is given the opportunity to implement or attempt hardware development utilizing a different process– how do they know if they choose the right one? While this choice is based on many things including planning, expertise, training, to name a few, one of the areas of focus should be the translation of activities that occur between the organizations standard process to a potential new or different process.

Figure 13 whether a given activity as identified as part of the generic life cycle, occurs in the method its being compared to. The figure assumes that the given methods will not execute that activity in the same manner, rather the intent in completing that activity is the same. If a team needs to identify stakeholder needs, they may go about it differently using based on the method, though the result would still be identification of stakeholder needs.

Take for example a team that needs to develop an electronic component as a technical refresh for a product that has no spares and cannot be repaired. Their typical hardware development process follows or is closely representative of the generic life cycle defined in the *INCOSE SE Handbook* where concept activities and development activities are those as defined in Table 2 and the process is like that shown in Figure 5. As part of the program plan (or similar documentation) describing the deliverables and subsequent activities of the development process, there are several activities that absolutely must be executed:

1. Define form, function, and features
2. Evaluate candidates
3. Develop models; Run simulations; Build prototypes
4. Refine system level requirements
5. Perform tests

6. Document system configuration

7. Procure/provisioning for production

This development team has also been given the opportunity to be apply a different method within the hardware development environment. However, they do not have enough knowledge to make an informed decision as to the method they could implement and the process they should execute. Understanding the activities that they need to execute, is the first step in figuring out whether a given method is feasible.

Given the seven activities this team is required to execute, Figure 13 can be used to determine where those activities occur in a specific method. The first step is to identify the activities that are required for execution, then to identify if the activity occurs in each method (which is done by seeing if an "X" is checked in the matrix). Figure 14 depicts these updates. The activities that require execution are bolded, and for each cell that is populated, it is highlighted green.

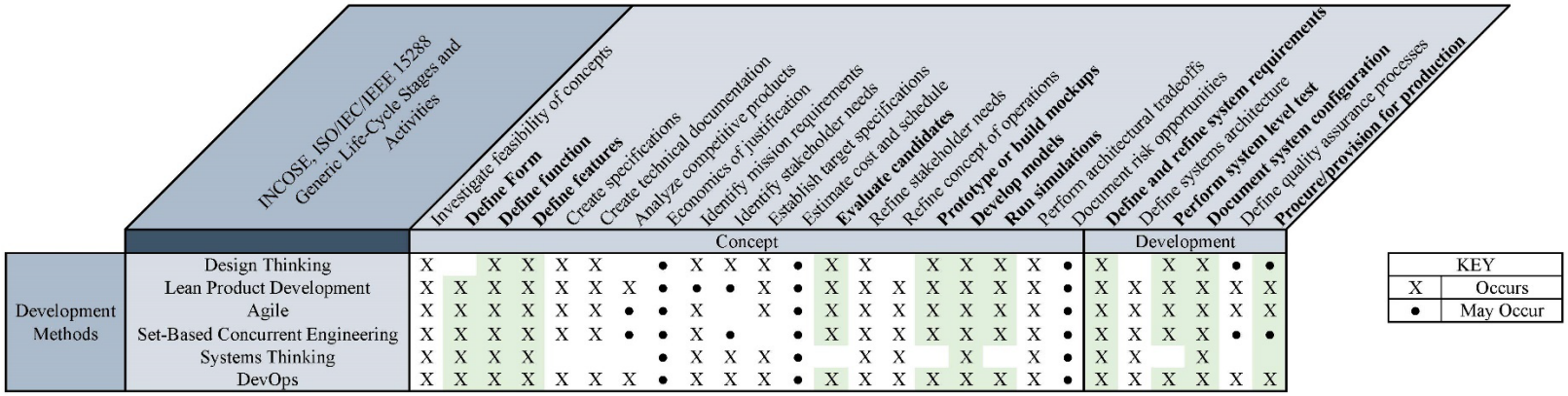

Figure 14. Example Use of Activity Mapping

Next, the team analyzes the chart to determine whether all applicable activities occur in each method. Based on Figure 14 LPD, agile, SBCE, and devops all execute these activities as art of the process. Therefore, the development team could use any one or combination of methods, considering only their activities.

V. CONCLUSIONS, LIMITATIONS, AND RECOMMENDATIONS

This thesis provides a high-level activity mapping of six development methods to the activities that occur in the generic life cycle defined by the *INCOSE SE Handbook,* specific to the concept and development stages for hardware-focused systems. This chapter presents conclusions, suggested follow-on research, and recommendations.

A. CONCLUSIONS

This thesis lays the groundwork for a much bigger effort in determining feasibility of methods for not only hardware development but systems engineering in general. What this work shows is that from an activity perspective, one or more methods can shape hardware development processes to make them more effective and totally tailorable. Using the generic life cycle as identified by the *INCOSE SE Handbook* sets a baseline for development process and activities by reiterating the most basic understanding of what is takes to execute development in any domain. The mapping presented reflects a comparison of the activities that occur as part of the generic life cycle concept and development stages to their existence in the six methods identified which can be used in any systems engineering or development environment within the commercial industry or the DOD. The application of this mapping is unlimited. It is the first step in creating a resource that is inclusive and exhaustive in documenting existing methods, especially as the engineering world continues to aim for superiority in its products and services.

B. LIMITATIONS

The six methods discussed in this work were pulled from human, software, and hardware focused development domains. There are limitations to these methods when applying them from one domain to another, including hardware. The literature and case study reviews completed as part of this work truly lacked a focus in hardware development processes and applications. There are limitations to hardware development that are not experienced in software development. For one, hardware must be acquired to proof out a design or build a mature enough product for testing. Modifications can be costly and impact

schedule, especially if a component is considered long lead. Testing is also not immediate, as functional testing and environmental testing require additional fixturing, test setups, and training for proper use. Due to the nature of hardware development, it is not surprising that many of methods are being applied to development life cycles that may have better access determine how to specifically apply these new methods, or potentially create one that is specific to hardware development. What this work does show, is that some version of the methods described throughout could apply to hardware development programs if they are considered solely on their activities.

Additionally, this work does not touch upon the implementation of more than one method in the hardware development process. Devops for example, has an agile structure and applied LPD principles. Literature also indicates that SBCE principles are foundational to the structure and flow of LPD. Therefore, devops could be an accumulation of two to three different methods. So, if a development team was to choose a method to execute, they would need to do further research as to how these could be integrated into one cohesive process. Additionally, it would be possible to use one method during concept and another during development if the activities aligned properly. Though, by applying two different methods, an organization may find the transition difficult and need to provide proper training to accompany the transformation.

C. RECOMMENDATIONS FOR FUTURE RESEARCH

This work is not exhaustive of all methods, their phases, implementations, and activities, so it would be beneficial to continue building upon the method alignment and the activity mapping presented herein.

Since the mapping presented in this work is specific in mapping activities to methods, additional work is needed to map activities to activities between methods. As in, the activities that occur during the generic life cycle stages of the *INCOSE SE Handbook* could be mapped to the activities that occur as part of agile when applied to development (for example). This would provide an in depth and true comparison of the activities and events that occur during development given a specific method is applied.

Furthermore, rather than solely considering activities, it would be beneficial for an organization to organize the defining characteristics for the methods discussed herein. This would give better insight to specific applications including best use case scenarios, engineering and organizational cultures, and industry implementation. In conjunction with an activity mapping (or mappings), it would help create a comprehensive resource for use in any engineering environment.

Finally, the development of a tool that a user could navigate by inputting or choosing activities and characteristics of a development program, that not only provides a recommended method but may even propose a baseline process would be beneficial to any organization and in any development program.

THIS PAGE INTENTIONALLY LEFT BLANK

LIST OF REFERENCES

Al-Ashaab, Ahmed, Matic Golob, Usama M Attia, Muhammad Khan, Jon Parsons, Alberto Andino, Alejandro Perez et al. 2013. "The Transformation of Product Development Process into Lean Environment Using Set-Based Concurrent Engineering: A Case Study from an Aerospace Industry." *Concurrent Engineering, Research and Applications* 21 (4): 268–85. https://doi.org/10.1177/1063293X13495220.

Arnold, Ross D, and Jon P Wade. 2015. "A Definition of Systems Thinking: A Systems Approach." *Procedia Computer Science* 44: 669–78. https://doi.org/10.1016/j.procs.2015.03.050.

Balcaitis, Ramunas. 2019. "Design Thinking Models. Standard d.school." June 15, 2019. https://empathizeit.com/design-thinking-models-stanford-d-school/

Banica, Logica, Magdalena Radeulescu, Doina Rosca, and Alina Hagui. 2017. "Is DevOps Another Project Management Methodology?" *Informatica Economica* 21 (3/2017): 39–51. https://doi.org/10.12948/issn14531305/21.3.2017.04.

Blanchard, Benjamin S., and W. J. Fabrycky. 2011. *Systems Engineering and Analysis*. 5th ed. Boston: Prentice Hall

Bjögvinsson, Erling, Pelle Ehn, and Per-Anders Hillgren. 2012. "Design Things and Design Thinking: Contemporary Participatory Design Challenges." *Design Issues* 28 (3): 101–16.

Canciglieri, Osiris, João Pedro Buiarskey Kovalchuk, Marcelo Rudek, and Teófilo Miguel de Souza. 2010. "Development of White Goods Parts in a Concurrent Engineering Environment Based on DFM/DFA Concepts." In *New World Situation: New Directions in Concurrent Engineering*, 491–501. London: Springer London. https://doi.org/10.1007/978-0-85729-024-3_47.

Capers, Jones. 2018. "Agile/Scrum Software Development." In *Software Methodologies a Quantitative Guide*, 1st ed., 49–55. CRC Press. https://doi.org/10/1201/9781315314488-5

Chang, YoungJoong, Jaibeom Kim, and Jaewoo Joo. 2013. "An Exploratory Study on the Evolution of Design Thinking: Comparison of Apple and Samsung." *Design Management Journal* 8 (1): 22–34. doi:10.1111/dmj.12001

Cooke, Jaime L. 2012. “A Five-Minute History of Agile” In *Everything You Want to Know About Agile*, 32–. IT Governance Publishing.

Dornenburg, Erik. 2018. “The Path to DevOps.” *IEEE Software* 35 (5): 71–75. https://doi.org/10.1109/MS.2018.290110337.

DiMeo, Andrew. 2018. “Design Thinking and Human Centered Design – What’s the Difference?” December 4, 2018. https://www.trig.com/explore/design-thinking-and-human-centered-design-whats-the-difference

Ebert, Christof, Gorka Gallardo, Josune Hernantes, and Nicolas Serrano. 2016. “DevOps.” *IEEE Software* 33 (3): 94–100. https://doi.org/10.1109/MS.2016.68.

Ferdinando, Lisa. 2018. “DOD Must Be More Agile in Technology Development, Official Says.” DOD News. April 19, 2018. https://www.defense.gov/Explore/News/Article/Article/1497393/

Galli, Brian J. 2018. “Effectively Using Systems Thinking in New Product Development (NPD).” *International Journal of Applied Logistics* 8 (2): 69–85. https://doi.org/10.4018/IJAL.2018070104.

Garamone, Jim. 2017. “DOD restructures acquisition, technology office to improve military lethality, speed.” DOD News Defense Media Activity. August 7, 2017. https://www.army.mil/article/191904/dod_restructures_acquisition_technology_office_to_improve_military_lethality_speed

Goede, Roelien. 2005. “A Framework for the Explicit Use of Specific Systems Thinking Methodologies in Data-Driven Support System Development.” Dissertation, University of Pretoria. http://hdl.handle.net/2263/24606

Hasso Plattner Institute of Design. 2019. *An Introduction to Design Thinking PROCESS GUIDE*. San Jose, California.

IDEO. n.d. “What is Design Thinking?” Accessed April 8, 2021. https://www.ideou.com/blogs/inspiration/what-is-design-thinking.

INCOSE. 2015. Systems Engineering Handbook: A Guide for System Life-cycle Processes and Activities. Hoboken, NJ: John Wiley & Sons.

Khan, Muhammad, Ahmed Al-Ashaab, Athanasia Doultsinou, Essam Shehab, Paul Ewers, and Robert Sulowski. 2011. “Set-Based Concurrent Engineering Process Within the LeanPPD Environment.” In *Improving Complex Systems Today*, 433–40. London: Springer London. https://doi.org/10.1007/978-0-85729-799-0_51.

Khan, Muhammad S, Ahmed Al-Ashaab, Essam Shehab, Badr Haque, Paul Ewers, Mikel Sorli, and Amaia Sopelana. 2013. “Towards Lean Product and Process Development.” *International Journal of Computer Integrated Manufacturing* 26 (12): 1105–16. https://doi.org/10.1080/0951192X.2011.608723.

Kim, Gene, Jez Humble, Patrick Debois, and John Willis. 2016. The DevOps Handbook How to Create World-Class Agility, Reliability, & Security in Technology Organizations. Portland: IT Revolution Pres, LLC.

Lebcir, Mohamed. 2006. *Health Care Management: The Contribution of Systems Thinking.* UHBS 2006:7. Hartfield, UK: The Business School University of Hertfordshire. https://uhra.herts.ac.uk/bitstream/handle/2299/683/S65.pdf?sequence=1

Leite, Leonardo, Carla Rocha, Fabio Kon, Dejan Milojicic, and Paulo Meirelles. 2020. “A Survey of DevOps Concepts and Challenges.” *ACM Computing Surveys* 52 (6): 1–35. https://doi.org/10.1145/3359981.

Liker, Jeffrey K, and James Morgan. 2011. “Lean Product Development as a System: A Case Study of Body and Stamping Development at Ford.” *Engineering Management Journal* 23 (1): 16–28. https://doi.org/10.1080/10429247.2011.11431884.

Lindberg, Tilman, Raja Gumienny, Birgit Jobst, and Christoph Meinel. 2010. “Is There a Need for a Design Process?” In *Proceedings of Design Thinking Research Symposium* 8. 243–254. https://hpi.de/fileadmin/user_upload/fachgebiete/meinel/papers/Design_Thinking/2010_Lindberg_Design.pdf

Linke, Rebecca. 2017. “Design thinking, explained.” September 14, 2017. MIT. https://mitsloan.mit.edu/ideas-made-to-matter/design-thinking-explained#:~:text=Design%20thinking%20is%20an%20innovative%20problem-solving%20process%20rooted,Brown,%20CEO%20and%20president%20of%20design%20company%20IDEO.

Maani, Kambiz E., and Robert Y. Cavana. 2007. “Systems Methodology.” *The Systems Thinker* 18 (8) (October): https://thesystemsthinker.com/wp-content/uploads/pdfs/180801pk.pdf

Mascitelli, Ronald. 2006. Lean Product Development Guidebook: Everything Your Design Team Needs to Improve Efficiency and Slash Time-to-Market. Northridge: Technology Perspectives

Mazzuchetti, Roselis, Elaine Lopes, and Ismael Barbosa. 2019. "Design Thinking in the Development of New Products: A Case Study." *International Journal of Development Research* 9 (05) (May): 27442 – 27444. IJDR

Measey, Peter., Chris. Berridge, Alex. Gray, Lazaro. Wolf, Peter. Measey, Les. Oliver, Barbara. Roberts, Michael. Short, and Darren. Wilmshurst. 2015. *Agile Foundations Principles, Practices and Frameworks*. 1st ed. Swindon: BCS Learning & Development Limited.

Microsoft. n.d. "What is DevOps." Accessed July 5, 2021. https://azure.microsoft.com/en-us/overview/what-is-devops/

Moore, Dale L. 2021. "Government as Lead Systems Integrator (LSI) as a DOD Acquisition Program Management Strategy." Unpublished paper. April 5, 2021.

Mueller-Roterberg, Christian. 2018. *Handbook of Design Thinking.* Kindle Direct Publishing: Washington.

Mynott, Colin. 2012. *Lean Product Development a Manager's Guide* London: Institution of Engineering and Technology.

Petersen, Mira, and Nana F Hempler. 2017. "Development and Testing of a Mobile Application to Support Diabetes Self-Management for People with Newly Diagnosed Type 2 Diabetes: A Design Thinking Case Study." *BMC Medical Informatics and Decision Making* 17 (1): 91–91. https://doi.org/10.1186/s12911-017-0493-6.

Plattner, Hasso, Christoph Meinel, and Larry Leifer. 2011. *Design Thinking Understand – Improve – Apply*. 1st ed. 2011. Berlin, Heidelberg: Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-642-13757-0.

Pop, Marlena. 2020. "Design Thinking in Product Development – Case Study: Leather Library." Paper presented at ICAMS 2020 8th International Conference on Advanced Materials and Systems, Bucharest, Romania.

Pries, Kim H., and Jon M. Quigley. 2010. *Scrum Project Management*. Boca Raton, FL: CRC Press.

Putta, Abheeshta, Maria Paasivaara, and Casper Lassenius. 2018. "Adopting Scaled Agile Framework (SAFe): a Multivocal Literature Review." In *Proceedings of the 19th International Conference on Agile Software Development*, 1–4. ACM. https://doi.org/10.1145/3234152.3234164

Qumer Gill, Asif, Abhishek Loumish, Isha Riyat, and Sungyoup Han. 2018. "DevOps for Information Management Systems." *VINE Journal of Information and Knowledge Management Systems* 48 (1): 122–39. https://doi.org/10.1108/VJIKMS-02-2017-0007.

Radeka, Katherine. 2013. The Mastery of Innovation a Field Guide to Lean Product Development. Boca Raton, Fla: CRC Press.

Ramage, Magnus, and Karen Shipp. 2020. "Peter Checkland." In *Systems Thinkers*, 151–59. London: Springer London. https://doi.org/10.1007/978-1-4471-7475-2_15.

Raudberget, Dag. 2011. "Enabling Set-Based Concurrent Engineering in traditional product development." Paper presented at International Conference on Engineering Design, University of Denmark.

Ries, Eric. 2011. The Lean Startup: How Today's Entrepreneurs Use Continuous Innovation to Create Radically Successful Businesses. 1st ed. New York: Crown Business.

Rutherford, Albert. 2019. *Learn to Think in Systems.* South Carolina: Kindle Direct Publishing. Kindle.

Samiphak, S., A M Agogino, S L Syme, and R H Lamoreaux. 2016. "Design a Systems Thinking in Development Engineering: A Case Study of Liver Fluke Infection in Khon Kean, Thailand." In *Proceedings of International Conference on Engineering Education and Research.* https://www.westernsydney.edu.au/__data/assets/pdf_file/0005/1176746/iCEER2016_Conference_Proceedings_official.pdf

Scaled Agile. 2021. *Achieving Business Agility with SAFe 5*. https://www.scaledagile.com/resources/safe-whitepaper/?_ga=2.53569384.1967315312.1626743188-605840441.1621991766

Schallmo, Daniel, Christopher A. Williams, and Klaus Lang. 2018. "An Integrated Design Thinking Approach-Literature Review, Basic Principles and Roadmap for Design Thinking." Paper presented at ISPIM Innovation Conference – Innovation, the Name of the Game. Stokholm, Sweden.

Senge, Peter M. 2006. *The Fifth Discipline: The Art and Practice of the Learning Organization*. Rev. and updated. New York: Doubleday/Currency.

Sobek, Durward K., Allen C Ward, and Jeffrey K Liker. 1999. "Toyota's Principles of Set-Based Concurrent Engineering." *Sloan Management Review* 40 (2): 67– 83.

Sommer, Anita Friis. 2019. "Agile Transformation at LEGO Group: Implementing Agile Methods in Multiple Departments Changed Not Only Processes but Also Employees' Behavior and Mindset." *Research Technology Management* 62 (5): 20–29. https://doi.org/10.1080/08956308.2019.1638486.

Smartsheet. n.d. "Waterfall." Accessed April 17, 2021. https://www.smartsheet.com/content-center/best-practices/project-management/project-management-guide/waterfall-methodology

Stowell, Frank. 2013. "Peter Checkland Interview." *International Journal of Information Technologies and Systems Approach* 6 (2): 53–60. https://doi.org/10.4018/jitsa.2013070105.

Sweeney, Linda B., and John D. Sterman. 2000. "Bathtub Dynamics: Initial Results of a Systems Thinking Inventory." *System Dynamics Review* 16 (4) (Winter): 249–286. doi: http://dx.doi.org.libproxy.nps.edu/10.1002/sdr.198.

Thomke, Stefan, and Barbara Feinberg. "Design Thinking and Innovation at Apple." Harvard Business School Case 609-066. Boston: Harvard Business School Publishing, 2009.

Tuli, Prashant, and Ravi Shankar. 2015. "Collaborative and Lean New Product Development Approach: A Case Study in the Automotive Product Design." *International Journal of Production Research* 53 (8): 2457–71. https://doi.org/10.1080/00207543.2014.974849.

Tsuchida, Bruce T., and Lawrence E. Jones. 2019. *Systems Dynamics Modeling an Approach to Planning and Developing Strategy in the Changing Electricity Industry*. Boston, MA: The Brattle Group. https://brattlefiles.blob.core.windows.net/files/16049_system_dynamics_modeling.pdf

Ulrich, Karl T., and Steven D. Eppinger. 2016. *Product Design and Development*. 6th ed. New York: McGraw-Hill/Irwin.

Vanderjack, Brian. 2015. *The Agile Edge: Managing Projects Effectively Using Agile Scrum*. 1st ed. New York, New York (222 East 46th Street, New York, NY 10017): Business Expert Press

Walsh, Kenneth R., and Sathiadev Mahesh. 2015. "Agile Scrum." In *Encyclopedia of Information Science and Technology.* 3rd ed. edited by Khosrow-Pour, D.B.A., Mehdi, 7018-7025. Hershey, PA: IGI Global, 2015. http://doi:10.4018/978-1-4666-5888-2.ch691

Wrigley, C., Nusem, E., & Straker, K. (2020. "Implementing Design Thinking: Understanding Organizational Conditions." *California Management Review*, 62 (2), 125–143.

Zhu, Liming, Len Bass, and George Champlin-Scharff. 2016. "DevOps and Its Practices." *IEEE Software* 33 (3): 32–34. https://doi.org/10.1109/MS.2016.81.

THIS PAGE INTENTIONALLY LEFT BLANK

INITIAL DISTRIBUTION LIST

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudley Knox Library
   Naval Postgraduate School
   Monterey, California